Handwritten Optical Music Recognition
PhD Thesis Proposal

Jan Hajič jr.
Faculty of Mathematics and Physics, Charles University
Institute of Formal and Applied Linguistics
hajicj@ufal.mff.cuni.cz

1 Introduction

Optical Music Recognition (OMR) is a field of document analysis that aims to automatically read music scores. Music notation encodes music in a graphical form; OMR backtracks through this process to extract the musical information from this graphical representation.

Common western music notation (CWMN) is an intricate system for visually representing music that has shown considerable resilience in the face of many developments in the musical world, and has thrived beyond its original European cultural sphere, being able to represent sufficiently well much of the world's musical traditions. It has been evolving for about 1000 years, with the current standard having emerged in the 17th century and stabilized around the end of the 18th century. In the latter half of the 20th century, there have been multiple attempts to find new ways of conveying musical thought visually from composer to performer; however, none have yet threatened the dominance of the CWMN system. As long as musicians think of music in terms of notes, and have the need to communicate this kind of music visually, CWMN is – probably – here to stay, or at most keep evolving. Throughout these years, CWMN has settled into an optimum for quickly and accurately decoding the musical information. Unfortunately for OMR, this optimum has proven rather complex for document analysis systems, and there are only a few sub-problems of OMR that have been resolved successfully since OMR has first been attempted over 50 years ago (Denis Pruslin, 1966; David S Prerau, 1971).

The main aim of the proposed thesis is to create an OMR system capable of reading handwritten CWMN music notation. Two tasks will be attempted: offline handwriting recognition, and augmenting the system with audio inputs for multimodal human-in-the-loop recognition. Given the state of the art in OMR, however, it was necessary to first create the supporting infrastructure of evaluation and datasets, including some theoretical work on ground truth design.

In the rest of this introduction, we first briefly present the motivation for OMR and its application domains (Subsec. 1.1). We then describe in more detail the process of reading music that OMR is seeking to emulate (Subsec. 1.2), and from this description we draw conclusions about why OMR is a difficult problem, especially compared to OCR (Subsec. 1.3). We finally describe the individual "flavors" of OMR based on the input modalities and output requirements (Subsec. 1.4).

Next, in Section 2, we describe how OMR solutions are typically structured and review the literature on OMR. We then proceed to present our contributions to OMR so far: in Section 3, we analyze the requirements for designing an OMR benchmark, and present the MUSCIMA++ dataset,[1] and in Section 4, we present a data-driven approach to OMR end-goal evaluation.[2] In Section 5, we then present a plan of experiments towards completing a workable handwritten offline optical music recognition system.

Section 6 summarizes current achievements and concludes the thesis proposal.

1.1 Motivation: Why OMR?

There are three main application domains where OMR can make a mark.

1.1.1 Live and post-hoc digitalization of music scores

OMR can be a useful tool for composers and arrangers, helping to digitize their work quicker; and for orchestral and ensemble leaders, especially early music practitioners, who often need to digitize and transcribe manuscripts. Even though digital note entry tools are available, they tend to restrict creative thought, and despite the best efforts of many developers, using these tools is often tedious and frustrating, which is detrimental for the composition process.[3]

[1] The section on data, 3, is taken from Hajič jr. and Pecina (2017); all the work from that publication used in the thesis proposal is the author's own.

[2] The section on evaluation, 4, is taken from the paper of Hajič jr. et al. (2016); all the work from that publication used in the thesis proposal is the author's own.

[3] We have conducted a survey among current Czech composers at the Janáček Academy of Music and Performing Arts, where 8 of 10 respondents stated they compose primarily by hand, and 9 out of 10 stated they would use OMR software if it was available and good enough.

1.1.2 Making accessible the musical content of large music notation archives

There is also a large amount of music that only exists in its written form, in archives: both the works of contemporary composers, and earlier works, especially from the 17th and 18th centuries. OMR would enable content-based search in these archives, at scale. In conjunction with open-source music analysis tools,[4] OMR has the potential to expand the scope of possible in musicology. In the Czech Republic,there are thousands of early music pieces in archives such as the Kroměříž collection,[5] or the Church of St. Jacob collection.[6] Even more importantly, many 20th and 21st-century composers have produced significant amounts of manuscripts that have not been published or digitized (e.g.: I. Hurník, K. Sklenička, J. Novák, F. G. Emmert, A. Hába)[7] that are, or may well become, a part of the "classical music canon" – given proper exposure. OMR could make such collections accessible at a fraction of the current costs, thus enabling preservation of valuable cultural resources and their active utilization and dissemination worldwide.[8] Besides significantly cutting down on publishing costs, OMR would open this trove of musical documents to digital humanities in musicology.

1.1.3 Integration of written music into other digital music applications

OMR may also be useful as a component of a multimodal system, where visual information from the score is combined with the corresponding audio signal. Score following, where the "active" part of the musical score is tracked according to the audio, is one such established application,[9] but applications in music pedagogy or a "musician's assistant" require more thorough OMR capabilities.

1.2 How music notation works

We have stated that "music notation encodes music in a written form, OMR backtracks through this process" to decode the music from its written form; this is *reading music*. Understanding this process is key to understanding the objectives of OMR. We now analyze the music reading process, in order to better understand the structure of the problem is OMR trying to solve.

The "music" that CWMN encodes can be described as an arrangement of *note* musical objects in time. Each of these objects has four characteristics: *pitch*, *duration*, *strength*, and *timbre*. When performed, these properties translate to fundamental frequency, projection of the note into time, loudness (perceptual, not always straightforwardly correlated to amplitude), and spectral characteristics, both for the harmonic series associated with the note, and for noise outside this row. *Reading music* denotes the process of gradually decoding this arrangement of notes from the written score. The rules of reading music determine *how* to decode notes from music notation. It is a stateful process: according to these rules, in order to correctly decode the next note, one needs to remember some of what has already been read, especially in order to correctly decode note *onsets*: when to start playing a certain note.

Pitch and duration are recorded most carefully. The entirety of music notation is centered around conveying them precisely – and very quickly, to allow for sight-reading: real-time performance, where the music is played just as the musician is reading. Encoding pitch also entails a complex apparatus that is intimately connected to the prevailing musical theory of *harmony* and relationships of pitches within this theoretical framework. The notation apparatus for strength and timbre is less complex, and CWMN does not attempt to encode these two attributes very precisely, merely providing some clues for the musician to interpret.

The primary point of contact between the space of music and the space of music notation is the *notehead*. Overwhelmingly often, this is a bijection: one notehead encodes one note object, and vice versa. Other music notation symbols are then used to encode the four attributes of notes: the aforementioned pitch, duration, strength, and timbre.

Pitch is a categorical variable, with values from a subset of frequencies defined by the scale and tuning used. In the western music tradition, the scale of available pitches usually consists of semitones, using predominantly – since the late 18th century – the *well-tempered* tuning system, where the frequency of the next higher semitone s_{i+1} is defined as $s_i * 2^{\frac{1}{12}}$.

The linear ordering of pitches by frequency, from low to high, is arranged into a repeating pattern of *steps*: A, B[10], C, D, E, F and G. The distance between neighboring steps is one *tone*, except for the step from B to C and from E to F, which is a semitone. One repetition of this pattern is called an *octave*. Octaves are in the English-speaking tradition numbered from 1 to 8, with the well-tempered tuning system defining A4 to have a frequency of 440 Hz.

Potentially, an *accidental* can shift the pitch by one

[4] Such as the `music21` library: `http://web.mit.edu/music21/`

[5] `http://wwwold.nkp.cz/iaml/studie_mahel2006.pdf`

[6] Inventory of musical items of the St. Jacob church in Brno, 1763, De Adventu, no. 4, Moravian Land Museum, copy G 5.035

[7] From personal communication with persons knowledgeable or in charge of the collections.

[8] The fact that OMR is a topic of interest to music archives, but still in prototype stages, is apparent also from a 2016 job posting at Bayerische Staatsbibliothek: `https://vifamusik.wordpress.com/2016/03/10/stellenausschreibung-der-musikabteilung-der-bayerischen-staatsbibliothek/`

[9] A score following track has been held since 2006 within the MIREX framework of music information retrieval competition.

[10] In German tradition, which includes the Czech Republic, B is called H.

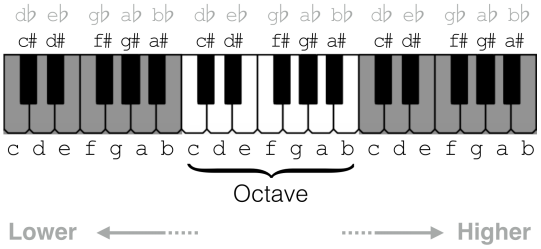


Figure 1: Indexing pitch, illustrated on a piano keyboard. One octave period is indicated.

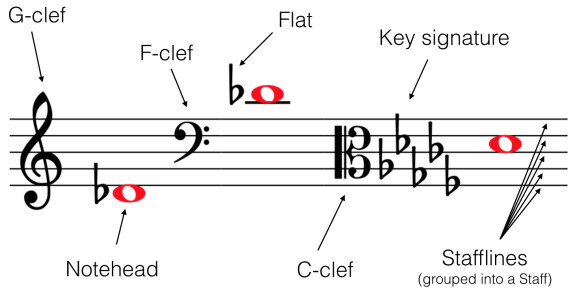


Figure 2: The elements encoding pitch: three notes with the same pitch written in different ways. Noteheads red, for clarity.

or two semitones in either direction: a *sharp* modifies it by one semitone up, a *flat* by one semitone down. This implies that in practice, a note of the same *frequency* can be expressed as different *pitches*: a C modified by a sharp refers – in the well-tempered tuning system – to a sound of the same fundamental frequency as the neighboring D modified by a flat. However, the *meaning* of these pitches is not the same for a musician, even though they may correspond to the same key on a piano: the role of a C sharp is different than that of a D flat within the system of harmony, which is another source of interpreting music notation from the written page to performance. Figure 1 illustrates how pitches are arranged and named on a piano keyboard.

The pitch of a note is encoded by the relationship of the notehead to staffines, clefs, accidentals (whether in the role of a global key signature, or as "inline" modifiers), and ledger lines; sometimes, transposition marks. Decoding pitch introduces long-distance relationships: the clef is generally only written at the start of a staff, and the relationship of the notehead to the clef is two-step: the clef determines how to interpret the stafflines (e.g., the G-clef on the left in Figure 2 tells the reader to interpret noteheads on the second staffline from the bottom as having pitch G4).

Duration is also categorical. It encodes how many *beats* (units of musical time) the given note lasts for. The durations are fractions of a beat, mostly in powers of 2, incl. negative: in terms of beats, 4, 2, 1, 1/2, 1/4, 1/8. The terminology calls a 4-beat note a

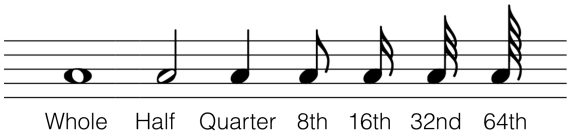


Figure 3: The types of notes according to duration. Two half-notes takes as long as one whole note, one half-note takes the same number of beats as two quarter-notes, etc.

whole note, a 2-beat note is a *half note*, then quarter note, eigth note, etc. (The most common grouping of beats into regular units, called *measures*, is by 4, hence the "whole" moniker for the 4-beat note.) Notes of the common types are depicted in Figure 3. Additionally, *tuple* symbols may encode durations that do not fit into this powers-of-two sequence, such as 1/3 or 2/5.

Duration is encoded through the notehead type (full, empty, and sometimes holdover "breve" rectangular noteheads from earlier, mensural notation), presence or absence of a stem, flags or beams, and can be modified also by *augmentation dots* that multiply the note's duration by 1.5. The elements of music notation that encode duration are illustrated in Figure 4.

Durations also indirectly encode note *onset*: as music is read left-to-right. the next note starts after the previous one has ended. In order to provide more synchronization, music is usually organized into units of a fixed number of beats (usually 3 or 4), which are denoted by *barlines*. The length of a bar unit is denoted by the *time signature*. There are also symbols that encode a certain duration during which no notes are present: these symbols are called *rests*.

A further complication is that consecutive groups of notes with sub-beat duration (8th notes, 16th notes, etc.) may be grouped using the *beam* symbol, in order to indicate their overarching temporal structure. Beamed groups usually group notes that together last 1 or 2 beats. Just like isolated 8th notes are indicated by the presence of one flag and 16th notes by 2 flags (see Fig. 3), 8th notes within beamed groups are indicated by the presence of a single beam, 16th notes by two beams, etc. Rules for constructing beamed groups allow practically an endless variety of beamed group configurations, and these groups are visually the most complex units of music notation, presenting the greatest challenge for OMR.

The projection of a note's duration into time is yet more complex (in fact, it is one of the most complex aspects of performing music, a subject of life-long learning even for professional musicians). There are two components to this projection. First, the relationship of beats to time, which is globally encoded by the *tempo* marking (usually a vague description, such as "quickly", leaving to the performer the exact rate of beats per minute), and local variations of this relation-

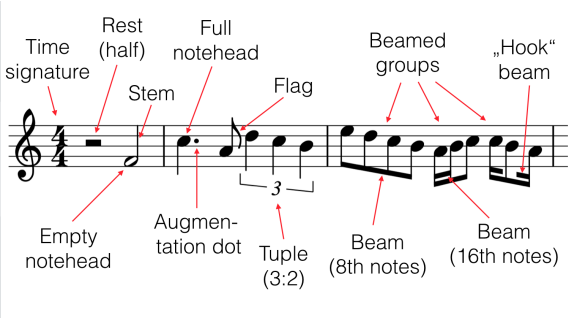


Figure 4: The elements encoding duration.

Figure 5: A somewhat complex beamed group. Note how it changes stem sizes, which are otherwise mostly fixed (at $3.5 * staffspace\_height$).

ship (called the *agogics* of music performance), which is usually not encoded at all. Second, there are further note-level *articulation* instructions that do not change its duration in terms of beats, but instruct the performer to only play the note for a portion of its assigned duration (*staccato*, *tenuto*, ...); there are also symbols that instruct one to lengthen the note (*fermata*). In terms of beats, the onset of the following note is not affected by the presence of articulation marks. Fortunately, beyond recording the presence of articulation marks and tempo-related text, OMR does not have to deal with the relationship of durations and time.

Strength and **timbre** are not as important for the purposes of OMR. Music notation expresses strength – when at all – through *dynamics* instructions, which are (a) textual signs from a pre-defined vocabulary of terms (*pianissimo*, *piano*, *mezzoforte*, etc.), (b) loudness change instructions, either textual (*cresc.*, *decr.*), or ideographical (hairpin symbols). Timbre is primarily manipulated through the selection of musical instruments, exploiting how their timbral characteristics change with respect to the other three parameters, and, if need be, textual instructions: both vague expressive terms that require musical intuition to interpret correctly (*dolce*, *con fuoco*, *maestoso*, ...), and directly controlling how the musical instrument is handled (*con sordinho*, *col legno*, instructions for percussion players about which stickheads to use, pedalization signs for piano, flageolets, etc.). Some of the instrument-specific timbral notation uses specific symbols instead of text.

1.3 Why is OMR difficult?

There is a clear case for describing OMR as "OCR for the music notation writing system". However, OMR remains an open problem and satisfactory solutions only exist for limited sub-problems (Bainbridge and Bell, 2001; Ana Rebelo et al., 2012; Jiří Novotný and Jaroslav Pokorný, 2015). This has two major reasons: OMR has attracted less attention and effort than OCR, and, more importantly, the task is more difficult.

There are multiple reasons why this is the case. First of all, in terms of graphical complexity, CWMN is more complex than practically any other writing system (Donald Byrd and Jakob Grue Simonsen, 2015). Following the classification of writing systems proposed by Sampson (1985), as opposed to *glottographic* writing systems for natural languages, which seek to encode visually what is *spoken* rather than what is *meant*, music is perhaps closer to a *semasiographic* writing system, trying to visually express ideas more directly,[11] or a complex *featural* writing system, which are characterized by using fixed ways of expressing sub-morpheme, sub-phoneme units or *categories* such as place of articulation or voicedness – this we see as analogous to how music notation represents "orthogonal" properties of notes using pre-defined sets of symbols that cannot occur in isolation, but have a certain fixed meaning with respect to notes (see Subsec. 1.2). The featural view of music notation is further supported by variants such as Braille notation, which also factorizes into symbols and sub-symbols according to the separate categories of pitch, duration, etc.

The main reasons why the way CWMN is written makes OMR more difficult than OCR are:

- Both the vertical and horizontal dimensions are salient and used to resolve symbol ambiguity.
- In terms of graphical complexity, the biggest issues are caused by overlapping symbols (including stafflines) (Bainbridge and Bell, 1997b) and composite symbol constructions, esp. beamed groups (see Fig. 5).
- In polyphonic music, there are multiple sequences written, in a sense, "over" each other – as opposed to OCR, where the ordering of the symbols is linear (Fig. 6).
- Recovering pitch and duration requires recovering long-distance relationships (Fig. 7).
- In handwritten music, the variability of handwriting leads to a lack of reliable topological properties overall (Fig. 8), and symbols become easier to confuse.

[11] A semasiographic writing system of natural language exists – or is well-documented to have existed – for Yukaghir, a nearly extinct group of languages in north-east Siberia: https://historyview.blogspot.cz/2011/10/yukaghir-girl-writes-love-letter.html, also taken from (Sampson, 1985).

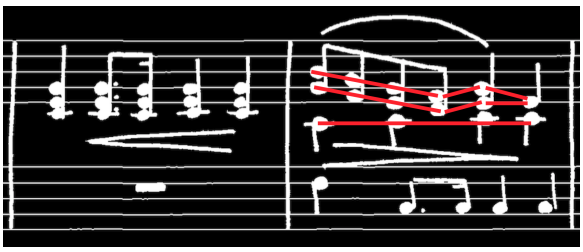


Figure 6: Multiple sequences of notes (voices) sharing symbols, appearing and disappearing.

(a) C-clef on the left influences how stafflines are interpreted with respect to the pitches they denote.

(b) Change of clef and key signature. Also, notice the sharp in the middle: it is valid up to the end of the measure.

Figure 7: Long-distance relationships affecting pitch of the note on the right.

Secondly, the objectives of OMR are also more ambitious than OCR. It aims to recover not just the locations and classification of musical symbols, but also "the music": pitch and duration information of individual notes. This introduces long-distance relationships: the interpretation of one "letter" of music notation may change based on notation events some distance away. These intricacies of music notation has been thoroughly discussed since early attempts at OMR, notably by D. Byrd in (Byrd, 1984; Donald Byrd and Jakob Grue Simonsen, 2015).

Finally, a separate problem that relates more to a lack of resources and effort, although inherent properties of music notation also play a role, is that OMR is seriously lacking in two aspects: a set of practical evaluation metrics that would enable practitioners to report comparable and replicable results, and datasets that would provide ground truth to evaluate performance on (Bainbridge and Bell, 2001; Michael Droettboom and Ichiro Fujinaga, 2004; Szwoch, 2008; Victor Padilla et al., 2014; Chanda et al., 2014; Baoguang Shi et al., 2015; Donald Byrd and Jakob Grue Simonsen, 2015; Hajič jr. et al., 2016; Arnau Baro et al., 2016). Furthermore, especially for handwritten notation, statistical machine learning methods have often been used that require training data for parameter estimation (Stuckelberg and Doermann, 1999; Rebelo et al., 2011b,a; Jorge Calvo-Zaragoza and Jose Oncina, 2014; Cuihong Wen et al., 2015).

1.4 Flavors of OMR

Based on the mode of input, OMR is divided into *offline* and *online*. Offline OMR processes images of music notation, online OMR processes the sequence of pen positions in time. Offline OMR is further divided according to whether the sheet music is printed, or handwritten. Different application domains also have different requirements on OMR system accuracy.

Online OMR is significantly easier, as the sequential nature of the input "untangles" most of symbol overlap. Individual input strokes serve as a good initial oversegmentation, although some writers may connect more symbols into one stroke. In contrast, for offline OMR, the analogous heuristic of connected components is relatively bad, even after stafflines are removed.

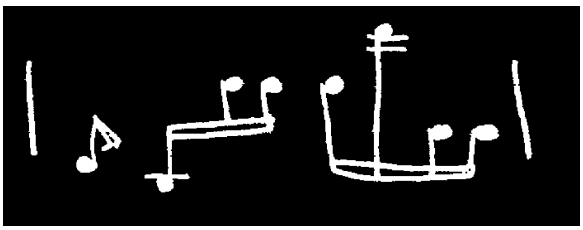
Handwriting is of course much harder to recognize than printed notation, because of its variability. Individual symbols assume radically different shapes, however, especially troublesome is the lack of topological constraints. The rules of music notation specify how individual symbols should indicate their relationships to one another: stems should be connected to the noteheads they pertain to, but none other; augmentation dots should be a certain distance away from the notehead; beams should be parallel to each other. The variability of handwriting styles is illustrated in Fig. 8.

OMR tasks can also be characterized by their *objectives:* What kind of information does the application need?

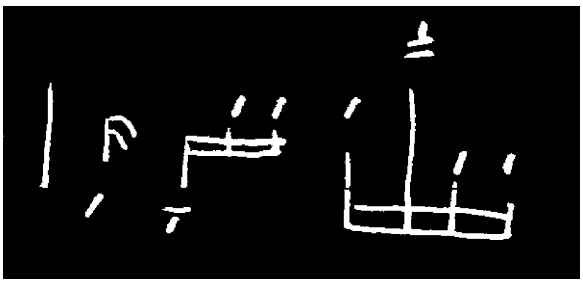
Miyao and Haralick (2000) group OMR applications into two broad groups: those that require replayability, and those that need reprintability. **Replayability** entails recovering pitches and durations of individual notes and organizing them in time by note onset. **Reprintability** means the ability to take OMR results as the input to music typesetting software and obtain a result that encodes this music in the same way as it was encoded in the input sequence. Reprintability implies replayability, but not vice versa, as one musical sequence can be encoded by different musical scores; e.g. MIDI is a good representation for replayability, but not reprintability (see Fig. 9).

This is an important distinction, which – unfortunately – has not been reflected upon further in OMR literature beyond the mention in the original paper (Miyao and Haralick, 2000). It clearly separates OMR applications that care about recovering *what* was expressed, in terms of pitches and durations (replayability), and applications that require knowing *how* it was expressed.

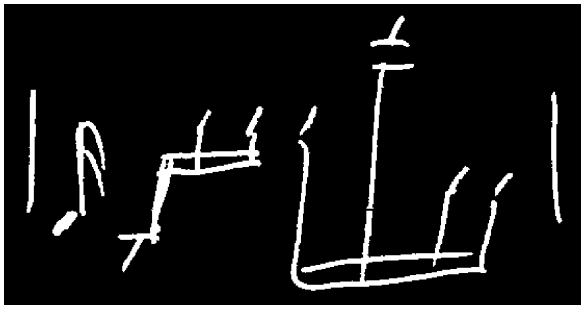
The distinction is somewhat arbitrary, as replayability in terms of the ability to *perform* the music as it was intended requires recovering nearly all of the score anyway. Nevertheless, it is a practical way to talk about OMR applications. For many applications such as query by humming or harmony search in music archives, it is enough to recover the corresponding MIDI, or even just its subset (pitch and duration data, or



(a) Writer 9: nice handwriting.



(b) Writer 49: Disjoint notation primitives



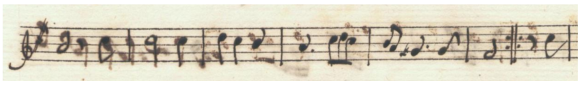
(c) Writer 22: Disjoint primitives and deformed noteheads. Some noteheads will be very hard to distinguish from the stem

Figure 8: Variety of handwriting. Taken from the CVC-MUSCIMA dataset (Fornés et al., 2012).

pitch sequences only). Understood in this way, replayability is less demanding overall, as many details about the musical score can be "forgotten". On the other hand, reprintability is needed whenever a musician is expected to read music based on OMR output. Music notation conveys subtle, yet important hints beyond what MIDI can record (beaming of notes may convey rhythmical nuance, other markings encode structural importance), but it does not necessarily require one to actually recover pitch and duration data, and thus the long-range relationships that are one of the reasons for the difficulty of OMR can be ignored.

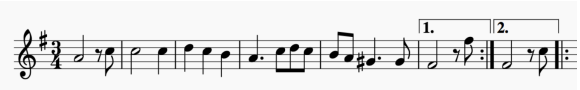
Besides the distinction between replayability and reprintability, various OMR objectives also differ in error tolerance. Full digital transcription of music scores for performance is the most demanding task, with no tolerance to error; each mistake is a detriment to performance. Searching the musical content of archives, on the other hand, may be more tolerant – one does not need every note recognized correctly to retrieve a useful subset of scores (Andrew Hankinson et al., 2012).

Out of these flavors of OMR, we focus on **offline OMR for handwritten CWMN.**

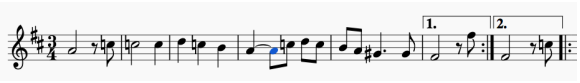


(a) Input: manuscript image.

(b) Replayable output: pitches, durations, onsets. Time is the horizontal axis, pitch is the vertical axis. This visualization is called a *piano roll*.



(c) Reprintable output: re-typesetting.



(d) Reprintable output: same music expressed differently

Figure 9: OMR for replayability and reprintability. The input (a) encodes the sequence of pitches, durations, and onsets (b), which can be expressed in different ways (c, d).

2 Related Work

We review OMR literature in three groups: the methods by which OMR is usually tackled (Subsec. 2.1), the state of OMR evaluation, which is a difficult problem in its own right (Subsec. 2.2), and the available OMR datasets (Subsec. 2.3).

2.1 Methods

OMR systems are usually pipelines with four major stages (Bainbridge and Bell, 2001; Ana Rebelo et al., 2012; Jiří Novotný and Jaroslav Pokorný, 2015; Andrew Hankinson, 2015):

1. Image preprocessing: enhancement, binarization, scale normalization;
2. Music symbol recognition: staffline identification and removal, localization and classification of remaining symbols;
3. Musical notation reconstruction: recovering the logical structure of the score;
4. Final representation construction: depending on the output requirements, usually inferring pitch and duration (MusicXML, MEI, MIDI, LilyPond, etc.).

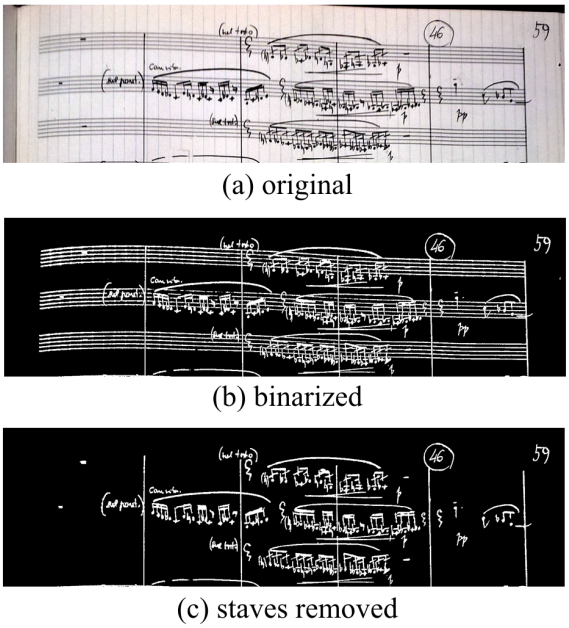
The **image preprocessing** stage is mostly practical: by making the image conform to some assumptions (such as: stems are straight, attempted by de-skewing), the OMR system has less complexity to deal with down the road, while very little to no information is lost. The

problems that this stage needs to handle are mostly related to document quality (degradation, stains, especially bleedthrough) and imperfections in the imaging process (e.g., uneven lighting, deformations of the paper; with mobile phone cameras, limited depth-of-field may lead to out-of-focus segments of the image) (Donald Byrd and Jakob Grue Simonsen, 2015). The most important problem for OMR in this stage is binarization (Ana Rebelo et al., 2012): sorting out which pixels belong to the background, and which actually make up the notation. There is evidence that sheet music has some specifics in this respect (John Ashley et al., 2008), and there have been attempts to tackle binarization for OMR specifically (JaeMyeong Yoo et al., 2008; Pinto et al., 2010, 2011). On the other hand, authors have attempted to bypass binarization, especially before staffline detection (Rebelo and Cardoso, 2013; Calvo Zaragoza et al., 2016), as information may be lost with binarization that could help resolve symbol overlap or other ambiguities later.

The input of **music symbol recognition** is a "cleaned" and usually binarized image. The output of this stage is a list of musical symbols recording their locations on the page, and their types (e.g., c-clef, beam, sharp). Usually, there are three sub-tasks: staffline identification and removal, symbol localization (in binary images, synonymous with foreground segmentation), and symbol classification (Ana Rebelo et al., 2012).

Stafflines are often handled first. Removing them then greatly simplifies the foreground, and in turn the remainder of the task, as it will make connected components a useful (if imperfect) heuristic for pruning the search space of possible segmentations (Fujinaga, 1996; Ana Rebelo, 2012). Staffline detection and removal is a large topic in OMR since its inception (Denis Pruslin, 1966; David S Prerau, 1971; Fujinaga, 1988), and it is the only one where a competition was successfully organized, by Fornés et al. (Fornes et al., 2011).

Staff detection and removal methods make use of the fact that stafflines are by definition long and straight, or at least should be. Straight stafflines have been detected by using horizontal projections since the first works of Denis Pruslin (1966) and David S Prerau (1971), notably also by Fujinaga (1988). For slightly curved and slanted stafflines, such as may happen with imperfect scanning, Fujinaga applies de-skewing (Fujinaga, 1996). Others use a *line-tracking* approach that attempts to identify adjacent vertical black runs that are not too much longer than the detected staffline height (Christoph Dalitz et al., 2008). dos Santos Cardoso et al. (2009), further elaborated on by Ana Rebelo et al. (2013), use the image as a graph of 8-connected pixels, assign costs corresponding to pixel intensity, and search for shortest stable paths from the left edge of the score to the right, assuming that the stafflines are the only extensive horizontal dark object, and remove staff pixels based on vertical run-lengths in binary images (Ana Rebelo, 2012) and what they call strong staff pixels (Rebelo and Cardoso, 2013) in grayscale images. More recently, convolutional networks have been applied by Calvo Zaragoza et al. (2016), achieving robust results beyond the CVC-MUSCIMA dataset used for the competition.



(a) original

(b) binarized

(c) staves removed

Figure 10: OMR pipeline from the original image through image processing, binarization, and staff removal. While staff removal is technically part of symbol recognition, as stafflines are symbols as well, it is considered practical to think of the image after staff removal as the input for recognition of other symbols.

Because errors during staff removal make further recognition complicated, especially by breaking symbols into multiple connected components with over-eager removal algorithms, some authors skip this stage: Sheridan and George instead add extra stafflines to annul differences between notes on stafflines and between stafflines (Sheridan and George, 2004), Pugin interprets the stafflines in a symbol's bounding box to be part of that symbol for the purposes of recognition (Laurent Pugin, 2006). However, even if stafflines are not removed, correctly extracting staffline and staffspace height is critical for scale normalization.

Next, the page is segmented into musical symbols, and the segments are classified by symbol class. While classification of musical symbols has produced near-perfect accuracy for both printed and handwritten musical symbols (Ana Rebelo, 2012; Chanda et al., 2014; Cuihong Wen et al., 2016), with baseline classification algorithms on raw pixel values achieving close to 80 % accuracy (Jorge Calvo-Zaragoza and Jose Oncina, 2014), segmentation of handwritten scores remains elusive. Most segmentation approaches such as projections (Fujinaga, 1988, 1996; Bellini et al., 2001) rely on topological constraints that do not necessarily hold in

handwritten music. Morphological skeletons have been proposed instead (Roach and Tatem, 1988; Ng et al., 1999; Luth, 2002) as a basis for handwritten OMR.

It is worth noting the different definitions of what the "alphabet" of elementary music notation symbols at this stage is. Some OMR researchers decompose notation into individual primitives (notehead, stem, flag) (Coüasnon and Camillerapp, 1994; Bainbridge and Bell, 1997a; Bellini et al., 2001; Bainbridge and Bell, 2003; Fornés, 2005), while others retain the "note" as an elementary visual object. Beamed groups are decomposed into the beam(s) and the remaining notehead+stem "quarter-like notes" (Rebelo et al., 2010; Ana Rebelo, 2012; Pham and Lee, 2015), or not included (Jorge Calvo-Zaragoza and Jose Oncina, 2014; Chanda et al., 2014).

In turn, the list of locations and classes of symbols on the page is the input to the **music notation reconstruction** stage. The outputs are more complex: at this stage, it is necessary to recover the relationships among the individual musical symbols, so that from the resulting representation, the "musical content" (most importantly, pitch and duration information – what to play, and when) can be unambiguously inferred.

Naturally, the set of relationships over a list of elementary music notation graphical elements (symbols) can be represented by a graph, possibly directed and requiring labeled edges. The list of symbols from the previous step can be re-purposed as a list of vertices of the graph, with the symbol classes and locations being the vertex attributes. While topological heuristics are readily available for printed music to clearly indicate how relationships should form, for handwritten music, this is also a non-trivial step. Graph-based assembly of music notation primitives has been used explicitly e.g. in (Reed and Parker, 1996; Miyao and Haralick, 2000; Liang Chen et al., 2015), and grammar-based approaches to notation reconstruction (e.g., (Fujinaga, 1988; Coüasnon and Camillerapp, 1994; Bainbridge and Bell, 2003; Szwoch, 2007; Fornés, 2005)) lend themselves to a graph representation as well, by recording the parse tree(s).

The outputs of symbol recognition and notation reconstruction are illustrated in Fig. 11.

Finally, the extracted infromation about the input score is **encoded in the desired output format.** There is a plethora of music encoding formats: from the text-based formats such as DARMS,[12] `**kern`,[13] LilyPond, ABC,[14] over NIFF,[15] MIDI,[16] to XML-based formats MusicXML[17] and MEI. The individual formats are each suitable for a different purpose: for instance, MIDI is most useful for interfacing different electronic audio devices, MEI is great for editorial work, LilyPond allows for excellent control of music engraving. Many of these have associated software tools that enable rendering the encoded music as a standard musical score, although some – notably MIDI – do not allow for a lossless round-trip. Furthermore, evaluating against the more complex formats is notoriously problematic (Szwoch, 2008; Hajič jr. et al., 2016).

Text-based formats such as LilyPond or ABC are ripe targets for end-to-end OMR, as they reduce the output to a single sequence, which enables the application of OCR, text-spotting, or even image-captioning models. This has been attempted specifically for OMR by Baoguang Shi et al. (2015) using a recurrent-convolutional neural network – although with only modest success on a greatly simplified task. However, even systems that only use stage 4 output and do not use stage 2 and 3 output in an intermediate step have to consider stage 2 and 3 information implicitly: that is simply how music notation conveys meaning.

That being said, stage 4 is – or, with a good stage 3 output, could be – mostly a technical step. The output of the notation reconstruction stage should leave as little ambiguity as possible for this last step to handle. In effect, the combination of outputs of the previous stages should give a potential user enough information to construct the desired representation for *any* task that may come up and does not require more input than the original image: after all, the musical score contains a finite amount of information, and it can be explicitly represented.[18]

2.1.1 Neural Networks for OMR

Fully-connected feedforward neural networks have been used for isolated symbol classification by Rebelo et al. (2010); Jorge Calvo-Zaragoza and Jose Oncina (2014); Cuihong Wen et al. (2015). Recently, deep learning approaches based on convolutional neural networks (CNNs) have been applied. These approaches have also demonstrated their utility for pixel-level labeling of musical score regions (music vs. lyrics vs. other) and robust staff removal (Calvo Zaragoza et al., 2016, 2017). Other authors tried to obviate the need for the binarization and staff removal steps for OMR (Baoguang Shi et al., 2015) and live score following of

[12] http://www.ccarh.org/publications/books/beyondmidi/online/darms/ – under the name Ford-Columbia music representation, used as output already in the DO-RE-MI system of Prerau, 1971 (David S Prerau, 1971).)

[13] http://www.music-cog.ohio-state.edu/Humdrum/representations/kern.html

[14] http://abcnotation.com/

[15] http://www.music-notation.info/en/formats/NIFF.html

[16] https://www.midi.org/

[17] http://www.musicxml.com/

[18] This does not mean, however, that the score contains by itself enough information to *perform* the music. That skill requires years of training, experience, familiarity with tradition, and scholarly expertise, and is not a goal of OMR systems.

the score image (Matthias Dorfer et al., 2016b). Deep learning for end-to-end OMR has so far been restricted to monody (Baoguang Shi et al., 2015), as representing polyphonic scores as outputs for deep learning models – and defining good and differentiable loss functions – is an open problem.

2.1.2 Multimodal OMR

First work on combining OMR with the audio modality was done by Fremerey et al. (2008, 2009) on automatically matching recordings to sheet music images.

Matthias Dorfer et al. (2016b) use multimodal OMR for score following in the image itself. The score is divided into buckets that correspond to sufficiently fine-grained positions. A multimodal deep learning model is then trained on pairs of [spectral window, window of score image]. In this work, both the audio and the graphical representation of the window was synthetically generated from an underlying semantic representation with using MIDI and LilyPond. The spectral window and the score image window were each passed as input, without further preprocessing, to the first layer of an image processing stack (score image) and audio processing stack (spectrum of the audio in the time-frame corresponding to the score image window), and a stack of two joint layers is added on top of the unimodal pillars. The objective function is classification categorical cross-entropy of the model prediction. This approach demonstrated quite convincing results in a live late-breaking demo at ISMIR 2016, including robustness to such perturbations as ornamentation or sudden jumps.

The work of Dorfer et al. was also novel from the Score following perspective. Traditionally (Cont et al., 2007), score following had at its input some structured representation of the musical content such as MIDI, not the score image; this is the definition of the corresponding task of the MIREX score following competition. Similarly, mOMR has the potential to enable using musical score images instead of structured transcriptions for score-informed source separation (Ewert et al., 2014) or score-informed polyphonic transcription (Benetos et al., 2012; Ewert et al., 2016).

2.1.3 User Feedback

"Pure" OMR has proven difficult, and for using OMR to transcribe music, it holds that OMR outputs remain under suspicion until cleared by a qualified reviewer (Christopher Raphael and Jingya Wang, 2011). Therefore, there has been work introducing other information aside from the score image to improve results. New applications are introduced that do not require full-scale automated transcription of sheet music – both as responses to existing needs of OMR users (musicians (Matthias Dorfer et al., 2016b) and musicologists (Andrew Hankinson et al., 2012), music pedagogy (Véronique Sébastien et al., 2012) and audiences and the general public (Dan Ringwalt et al., 2015)), and as stepping stones, partial steps towards a complete OMR system. Hankinson et al. (Andrew Hankinson et al., 2012) argue – very rightly so, we believe – that OMR should expand beyond the transcription application for individual users and towards retrieval from large collections, but individual transcription for performance purposes will remain a major target application of OMR for musicians and composers, so the need for user feedback is here to stay. The question is: how to efficiently incorporate user feedback?

Fujinaga (1996) proposed an adaptive system that learned from user feedback over time. Maura Church and Michael Scott Cuthbert (2014) created an interface to let users correct misrecognized rhythmic patterns using correct measures elsewhere in the score. In contrast to these post-editing approaches, Liang Chen et al. (2016) incorporate human guidance directly into the recognition process. Jorge Calvo-Zaragoza et al. (2016) combine the musical score image with the signal from pen-based tracing of the symbols, merging the offline and online modalities of OMR.

What has *not* been attempted yet is Interactive OMR guided by audio input. Closest is the work of Matthias Dorfer et al. (2016b,a), even though playing the music in question seems to be the fastest and most natural way of providing user feedback: after all, musical instruments are exactly the interfaces intended for the interpretation of the musical score.

2.2 Evaluation

The problem of evaluating OMR and creating a standard benchmark has been discussed before (Michael Droettboom and Ichiro Fujinaga, 2004; Pierfrancesco Bellini et al., 2007; Szwoch, 2008; Graham Jones et al., 2008; Donald Byrd and Jakob Grue Simonsen, 2015) and it has been argued that evaluating OMR is a problem as difficult as OMR itself (Michael Droettboom and Ichiro Fujinaga, 2004). Graham Jones et al. (2008) suggest that in order to automatically measure and evaluate the performance of OMR systems, we need (a) a standard dataset and standard terminology, (b) a definition of a set of rules and metrics, and (c) definitions of different ratios for each kind of errors. The authors noted that distributors of commercial OMR software often claim the accuracy of their system is about 90 %, but provide no information about how that value was estimated.

Pierfrancesco Bellini et al. (2007) manually assess results of OMR systems at two levels of symbol recognition: low-level, where only the presence and positioning of a symbol is assessed, and high-level, where the semantic aspects such as pitch and duration are evaluated as well. At the former level, mistaking a beamed group of 32nds for 16ths is a minor error; at the latter it is much more serious. They defined a detailed set of rules for counting symbols as recognized, missed and confused symbols. The symbol set used

by Pierfrancesco Bellini et al. (2007) is relatively rich: 56 symbols. They also define *recognition gain*, based on the idea that an OMR system is at its best when it minimizes the time needed for correction as opposed to transcribing from scratch, and stress *verification cost*: how much it takes to verify whether an OMR output is correct.

An extensive theoretical contribution towards benchmarking OMR has been made recently by Donald Byrd and Jakob Grue Simonsen (2015). They review existing work on evaluating OMR systems and clearly formulate the main issues related to evaluation. They argue that the complexity of CWMN is the main reason why OMR is inevitably problematic, and suggest the following stratification into levels of difficulty:

1. Music on one staff, strictly monophonic,
2. Music on one staff, polyphonic,
3. Music on multiple staves, but each strictly monophonic, with no interaction between them,
4. "Pianoform": music on multiple staves, one or more having multiple voices, and with significant interaction between and/or within staves.

They provide 34 pages of sheet music that cover the various sources of difficulty. However, the data does not include handwritten music and no ground truth for this corpus is provided.

Automatically evaluating MusicXML has been attempted most significantly by Szwoch (2008), who proposes a metric based on a top-down MusicXML node matching algorithm and reports agreement with human annotators, but how agreement was assessed is not made clear, no implementation of the metric is provided and the description of the evaluation metric itself is quite minimal. Due to the complex nature of MusicXML (e.g., the same score can be correctly represented by different MusicXML files), Szwoch also suggests a different representation may be better than comparing two MusicXML files directly.

More recently, evaluating OMR with MusicXML outputs has been done by Victor Padilla et al. (2014). While they provide an implementation, there is no comparison against gold-standard data. (This is understandable, as the work of Victor Padilla et al. (2014) is focused on recognition, not evaluation.) Aligning MusicXML files has also been explored by Knopke and Byrd (2007) in a similar system-combination setting, although not for the purposes of evaluation. They however make an important observation: stems are often mistaken for barlines, so the obvious simplification of first aligning measures is not straightforward to make.

In an attempt to tie errors in semantics to units that can be counted, instead of a notation reconstruction stage, authors define two levels of symbols: *low-level* primitives that cannot by themselves express musical semantics, and *high-level* symbols that already do have some semantic interpretation (Michael Droettboom and Ichiro Fujinaga, 2004; Pierfrancesco Bellini et al., 2007; Donald Byrd and Jakob Grue Simonsen, 2015). For instance, the letter `p` is just a letter from the low-level point of view, but a dynamics sign from the high-level perspective. From the perspective of a notation graph (see 2.1, stage 3 of the OMR pipeline), the high-level symbols can also belong to the symbol set of stage 2, and the two levels of description can be explicitly linked. The fact that correctly interpreting whether the `p` is a dynamics sign, or part of a text (e.g., `presto`) requires knowing the positions and classes of other symbols, simply hints that it may be a good idea to solve stages 2 and 3 jointly.

To summarize, OMR evaluation wrestles with the two-step nature of OMR: on the one hand, OMR just recognizes symbols on the page, on the other hand, it is only useful when it also recovers the musical information, as analyzed in 1.2. Furthermore, when using OMR to transcribe scores, it is not clear how individual errors relate to the effort it takes to correct them, as the toolchain used for post-editing and the individual editors factor in.

In Hajič jr. et al. (2016), we introduced a data-driven approach to automated cost-to-correct evaluation for end-to-end OMR that attempts to take these concerns into account. This work of ours is described in detail in sec. 4.

2.3 Datasets

There are already some OMR datasets available.

For **staff removal** in handwritten music, the premier dataset is CVC-MUSCIMA of Fornés et al. (2012), consisting of 1000 handwritten scores (20 pages of music, each copied by hand by 50 musicians). The state-of-the-art for staff removal has been established with a competition using CVC-MUSCIMA (Fornes et al., 2011). For each input image, CVC-MUSCIMA has three binary images: a "full image" mask, which contains all foreground pixels; a "ground truth" mask of all pixels that belong to a staffline and at the same time to no other symbol, and a "symbols" mask that complementarily contains only pixels that belong to some *other* symbol than a staffline. The dataset was collected by giving a set of 20 pages of sheet music to 50 musicians. Each was asked to rewrite the same 20 pages by hand, using their natural handwriting style. A standardized paper and pen was provided for all the writers, so that binarization and staff removal was done automatically with very high accuracy, and the results were manually checked and corrected. In vocal pieces, lyrics were not transcribed. The dataset also provides a *variety* of data:the 20 pages include scores of various levels of complexity and a wide array of music notation symbols (including tremolos, glissandi, grace notes, or trills), and handwriting style varies greatly among the 50 writers, including topological inconsistencies, as illustrated in Fig. 8. Importantly, CVC-MUSCIMA is freely available for download under a CC-BY-NC-SA

Table 1: CVC-MUSCIMA notation complexity

Complexity level	CVC-MUSCIMA pages
Single-staff, single-voice	1, 2, 4, 6, 13, 15,
Single-staff, multi-voice	7, 9 (?), 11 (?), 19
Multi-staff, single-voice	3, 5, 12, 16,
Multi-staff, multi-voice	14 (?), 17, 18, 20
Pianoform	8, 10

4.0 license.[19]

Handwritten **symbol classification** systems can be trained and evaluated on the HOMUS dataset of Jorge Calvo-Zaragoza and Jose Oncina (2014), which provides 15200 handwritten musical symbols (100 writers, 32 symbol classes, and 4 versions of a symbol per writer per class, with 8 for note-type symbols). HOMUS is also interesting in that the data is captured from a touchscreen: it is availabe in *online* form, with x, y coordinates for the pen at each time slice of 16 ms, and for offline recognition (ie. from a scan), images of music symbols can be generated from the pen trajectory. Together with potential data from a recent multimodal recognition (offline + online) experiment (Jorge Calvo-Zaragoza et al., 2016), these datasets might enable trajectory reconstruction from offline inputs. Since online recognition has been shown to perform better than offline on the dataset (Jorge Calvo-Zaragoza and Jose Oncina, 2014), such a component – if performing well – could lead to better OMR accuracy.

Rui Miguel Filipe da Silva (2013) collects a similar dataset of 12600 symbols from 50 writers (84 different symbols, each drawn 3 times), but this dataset is not made available.

However, these datasets only contains isolated symbols, not their positions on a page. While it might be possible to synthesize handwritten music pages from the HOMUS symbols, such a synthetic dataset will be rather limited, as HOMUS does not contain beamed groups and chords.[20] For **symbol localization**, we are only aware of a dataset of 3222 handwritten symbols by the group of Rebelo et al. (Rebelo et al., 2010; Ana Rebelo, 2012).

For **notation reconstruction**, we are not aware of a dataset that explicitly marks the relationships among handwritten musical symbols. While Pierfrancesco Bellini et al. (2007) do perform evaluation of OMR systems with respect to pitch and duration on a limited dataset of 7 pages, the evaluation was done manually, without creating a ground truth for this data.

Musical content reconstruction is usually understood to entail deriving pitch and relative duration of the written notes. It is possible to mine early music manuscripts in the IMSLP database[21] and pair them against their open-source editions, which are sometimes provided on the website as well, or look for matching encoded data in large repostories such as Mutopia[22] or KernScores[23]; however, we are not aware of such a paired collection for OMR, or any other available dataset for pitch and duration reconstruction.

3 The MUSCIMA++ Dataset

As has been stated, OMR is in dire need of benchmarking. One part of the equation are evaluation metrics: how do we measure the performance of a system? The other part is a benchmark dataset: what do we measure OMR performance *on?*

In this section,[24] we present MUSCIMA++[25], our new extensive dataset of handwritten musical notation for OMR. We use the term *dataset* in the following sense: $\mathcal{D} = \langle (x_i, y_i)\ \forall i = 1 \dots n \rangle$. Given a set of inputs x_i (in our case, images of sheet music), the dataset $\mathcal{D}$ records the desired outputs y_i – ground truth. The quality of OMR systems can then be measured by how closely they approximate the ground truth, although defining this approximation for the variety of representations of music is very much an open problem (see 2.2, 4).

For MUSCIMA++, we proposed and applied a principled ground truth definition that bridges the gap between the graphical expression of music and musical semantics (described in Subsec. 1.2), enabling evaluation of multiple OMR sub-tasks up to inferring pitch and duration both in isolation and jointly, and provide open-source tools for processing the data, visualizing it and annotating more. In the rest of this section, we describe how the dataset is designed

3.1 Designing a benchmark dataset

To build a dataset of handwritten music, we need to decide:

- What should the desired output y_i be for an image x_i? *(Ground truth.)*
- What sheet music do we choose as data points? *(Choice of data.)*

[19] http://www.cvc.uab.es/cvcmuscima/index_database.html

[20] It should also be noted that HOMUS is not available under an open license, so copyright restrictions apply. Not even every EU country has the appropriate "fair use"-like exception for academic research, even though it is mandated by the EU Directive 2001/29/EC, Article 5(3) (http://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX:32001L0029).

[21] https://www.imslp.org

[22] http://www.mutopiaproject.org

[23] http://humdrum.ccarh.org

[24] Corresponding to the author's work described in (Hajič jr. and Pecina, 2017).

[25] Standing for MUsic SCore IMAges, credit for abbreviation to (Fornés et al., 2012)

Table 2: OMR Pipeline as inputs and outputs

Sub-task	Input	Output
Image Processing	Score image	"Cleaned" image
Binarization	"Cleaned" image	Binary image
Staff ID & removal	Binary image	Stafflines list
Symbol localization	(Staff-less) image	Symbol regions
Symbol classification	Symbol regions	Symbol labels
Notation assembly	Symbol regs. & labels	Notation graph
Infer pitch/duration	Notation graph	Pitch/duration attrs.
Output conversion	Notation graph + attrs.	MusicXML, MIDI, ...

3.1.1 Ground truth

The definition of ground truth must **reflect what OMR *does*** (see 1.2, 1.4). The dataset should capture enough information to be useful for OMR aimed both at replayability and reprintability.

As the ground truth over a dataset is the desired output of a system solving a task, we need to understand how the OMR pipeline can be expressed in terms of inputs and outputs. This is summarized in Table 2. The key problems of OMR reside in stages 2 and 3: finding individual musical symbols on the page, and recovering their relationships.

The input of **music symbol recognition** is a "cleaned" and usually binarized image. The output of this stage is a list of musical symbols recording their locations on the page, and their types (e.g., c-clef, beam, sharp). Usually, there are three sub-tasks: staffline identification and removal, symbol localization (in binary images, synonymous with foreground segmentation), and symbol classification (Ana Rebelo et al., 2012). Stafflines are typically handled separately: they have distinct graphical properties that enable solving staffline identification and removal in isolation, and removing them then greatly simplifies the foreground, as it will make connected components a useful (if imperfect) heuristic for pruning the search space of possible segmentations (Fujinaga, 1996; Ana Rebelo, 2012).

In turn, the list of locations and classes of symbols on the page is the input to the **music notation reconstruction** stage. At this stage, it is necessary to recover the relationships among the individual musical symbols. These relationships open the gateway towards inferring the "musical content" (most importantly, pitch and duration information – what to play, and when): there is a 1:1 relationship between a notehead notation primitive and a note musical object, of which pitch and duration are properties,and the other symbols that relate – directly or indirectly – to a notehead, such as stems, stafflines, beams, accidentals, or clefs, inform the reader's decision to assign the pitch and duration.

The result of OMR stage 3 naturally forms a graph. The symbols from the previous stage become vertices of the graph, with the symbol classes and locations being the vertex attributes, and the relationships between symbols assume the role of edges. Graph-based assembly of music notation primitives has been used explicitly e.g. by Reed and Parker (1996); Liang Chen et al. (2015), and grammar-based approaches (e.g., (Fujinaga, 1988; Coüasnon and Camillerapp, 1994; Bainbridge and Bell, 2003; Szwoch, 2007)) lend themselves to a graph representation as well, by recording the parse tree(s).

An example of symbol recognition and notation reconstruction output over the same snippet of a musical score is given in Figure 11.

The notation graph is a good ground truth for an OMR dataset. We contend that its information content is both necessary and sufficient for both replayability and reprintability.

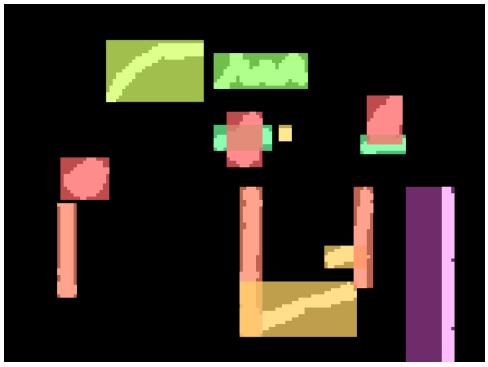
Necessary. Before the notation graph is constructed in stage 3, there is not enough information extracted for the output to be either replayable or reprintable. Recognizing a note is not enough to determine its pitch: one needs it to relate to the stafflines, accidentals, etc. The non-locality of pitch, duration, and onset information implies that whatever the design of the vocabulary of (contiguous) musical symbols, there will always be a configuration of notes in time that, when correctly encoded into CWMN, will be impossible to represent using such a vocabulary. Recovering relationships between symbols is necessary for replayability.

Sufficient. The nature of reading musical scores (see Subsec. 1.2) is such that stage 3 output is the point where the OMR system has extracted *all* the useful information – signal – from the input image, resolving all ambiguities; the system is therefore properly "free" to forget about the input image. All that remains to project the written page to the corresponding point in the space of musical note[26] configurations in time is to follow the rules for reading music, which can be expressed in terms of querying the graph to infer additional properties of the nodes representing noteheads – essentially, a graph transformation. This implies that all it takes to create the desired representation in stage 4 is a technical task: implementing conversion to the desired output format (which can nevertheless still be a very complex piece of software).[27]

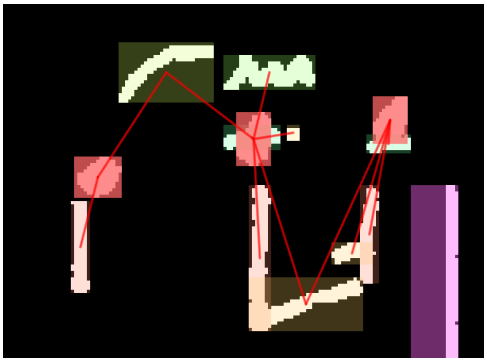
A notable and desirable property of the notation graph is that once inter-symbol relationships are fully recovered (including precedence, simultaneity and high-level relationships between staves), *symbol posi-*

[26] A musical note object, as opposed to the written note, is characterized in music theory by four attributes: pitch, duration, loudness, and timbre, of which OMR needs to recover pitch and duration; the musical score additionally encodes the onsets of notes in musical time.

[27] Note that while the dataset will record its ground truth in some representation, it is not necessarily *the* representation that should be used for experiments. Rather, experiment-specific output representations (such as pitch sequences for end-to-end experiments by Baoguang Shi et al. (2015), or indeed a MIDI file for replayability-only experiments) is *unambiguously obtainable* from the notation graph. This choice of dataset representation is made to make clear its "information content".



(a) Notation symbols: noteheads, stems, beams, ledger lines, a duration dot, slur, and ornament sign; part of a barline on the lower right. Vertices of the notation graph.



(b) Notation graph, highlighting noteheads as "roots" of subtrees. Noteheads share the beam and slur symbols.

Figure 11: Visualizing the list of symbols and the notation graph over staff removal output. Colors of symbol bounding boxes correspond to symbol classes (noteheads in red, stems in orange, ledger lines in green, etc.). The notation graph in (b) allows unambiguously inferring pitch and duration (stafflines removed for clarity, although for encoding pitch, we would need to establish the relationship of the noteheads to stafflines).

tions cease to be informative: they serve primarily as features that help music readers infer these relationships. If we wanted to, we could forget the input image after stage 3.

3.1.2 Choice of data

Second, the selection of musical score images in the dataset should **cover the "dimensions of difficulty"**, to allow for assessing OMR systems with respect to increasingly complex inputs.

As we are focusing on musical manuscripts, we collect a dataset of **handwritten notation.** This is also the economical choice: for printed music notation, the lack of datasets can be bypassed by generating music in representations such as LilyPond[28] or MEI,[29] and capturing intermediate steps of the rendering process, but for handwritten music, no satisfactory synthetic data generator exists so far, and thus an extensive annotation effort cannot be avoided.

What is the challenge space of OMR, besides printed vs. handwritten inputs? In their state-of-the-art analysis of the difficulties of OMR, Byrd and Simonsen (Donald Byrd and Jakob Grue Simonsen, 2015) identify three axes along which musical score images become less or more challenging inputs for an OMR system:

- Notation complexity
- Image quality
- Tightness of spacing (adherence to topological standards)

The dataset should also contain a wide variety of musical *symbols*, including less frequent items such as tremolos or glissandi, to enable differentiating systems also according to the breath of their vocabulary.

The axis of **notation complexity** is structured by Donald Byrd and Jakob Grue Simonsen (2015) into four levels:

1. Single-staff, monophonic music (one note at a time),
2. Single-staff, multi-voice music (chords or multiple simultaneous voices),
3. Multiple staves, monophonic music on each
4. Multiple staves, "pianoform" music.

The category of pianoform music is defined as multi-staff, polyphonic, with interaction between staves, and with no restrictions on how voices appear, disappear, and interact throughout.[30]

Each of these levels brings a new challenge. Level 1, single-staff single-voice music, tests an "OMR minimum": the recognition of individual symbols for a single sequence of notes. Level 2, single-staff multi-voice music, tests the ability to deal with multiple sequences of notes in parallel, so e.g. rhythmical constraints based on time signatures (Ana Rebelo et al., 2013) are harder to use (but still applicable (Rong Jin and Christopher Raphael, 2012)). Level 3, multi-staff single-voice music, tests high-level segmentation into systems and staffs; this is arguably easier than dealing with the polyphony of level 2 (Pau Riba et al., 2015), as the voices on the staves are not allowed to interact. Level 4, pianoform music, then presents the most complex, combined challenge, as piano music has exploited

[28] http://www.lilypond.org
[29] http://www.music-encoding.org

[30] Technically, we should also add another level between multi-staff monophonic and pianoform music: multi-staff music with multi-voice staves, but without notation complexities specific to the piano, as described in (Donald Byrd and Jakob Grue Simonsen, 2015) appendix C.

the rules of CWMN to their fullest (Donald Byrd and Jakob Grue Simonsen, 2015) and sometimes beyond.[31] The dataset should contain a choice of musical scores representing all these levels.

On the other hand, difficulties relating to **image quality** – deformations, noise, and document degradations – do not have to be represented in the dataset. Their descriptions in (Donald Byrd and Jakob Grue Simonsen, 2015) essentially define how to simulate them: increasing salt-and-pepper noise, random pertrubations of object borders, and distortions such as kanungo noise or localized thickening/thinning operations. Many morphological distortions have already been implemented for staff removal data (Christoph Dalitz et al., 2008; Fornés et al., 2012). Also bearing in mind that the dataset is supposed to address stages 2 and 3 of the OMR pipeline, this leads us to believe that **it is a reasonable choice to annotate binary images,** also with respect to the economy of the annotation process.

The **tightness of spacing** in (Donald Byrd and Jakob Grue Simonsen, 2015) refers to default horizontal and vertical distances between symbols.[32] As spacing tightens, assumptions about relative notation spacing may cease to hold: Byrd and Simonsen give an example where the augmentation dot of a preceding note can be easily confused with a staccato dot of its following note (see (Donald Byrd and Jakob Grue Simonsen, 2015), Fig. 21). This leads to increasingly ambiguous inputs to the primitive assembly stage. In handwritten music, variability in spacing is superseded by the variability of handwriting itself. Handwritten music gives no topological guarantees: by definition straight lines, such as stems, become curved, noteheads and stems do not touch, accidentals and noteheads *do* touch, etc. The various styles of handwriting, and the ensuing challenges, also have to be represented in the dataset, as broadly as possible. As there is currently no model available to synthesize realistic handwritten music scores, we need to cover this spectrum directly through choice of data.

Based on this analysis, we designed the MUSCIMA++ dataset ground truth, selected music to annotate, and collected the data. In the following sections, we describe the specifics of the resulting data.

3.2 Inside MUSCIMA++ 0.9

Our main source of musical score images is the CVC-MUSCIMA dataset described in Subsection 2.3. The CVC-MUSCIMA dataset fulfills the requirements for a good choice of data: the 20 pages include scores of all 4 levels of complexity, as summarized in Table 1, and a wide array of music notation symbols (including tremolos, glissandi, grace notes, or trills), and handwriting style varies greatly among the 50 writers, including topological inconsistencies.

The goal for the first round of MUSCIMA++ annotation was for each of our annotators to mark one of the 50 versions for each of the 20 pages. With 7 available annotators, this amounted to 140 annotated pages of music. Furthermore, we assigned the 140 out of 1000 pages of CVC-MUSCIMA so that all of the 50 writers are represented as equally as possible: 2 or 3 pages are annotated from each writer.[33]

There is a total of 91255 symbols marked in the 140 annotated pages of music, of 107 distinct symbol classes. There are 82261 relationships between pairs of symbols. The total number of *notes* encoded in the dataset is 23352. The set of symbol classes consists of both notation primitives, such as noteheads or beams, and higher-level notation objects, such as key signatures or time signatures. The specific choices of symbols and ground truth policies is described in Subsec. 3.4.

The frequencies of the most significant symbols are described in Table 3. We can draw two lessons immediately from the table. First, even when lyrics are stripped away (Fornés et al., 2012), texts make up a significant portion of music notation – nearly 5 % of symbols are letters. Some utilization of handwritten OCR, or at least identifying and removing texts, therefore seems reasonably necessary. Second, at least among 18th to 20th century music, some 90 % of notes occur as part of a beamed group, so works that do not tackle beamed groups are in general greatly restricted (possibly with the exception of choral music such as hymnals, where isolated notes are still the standard).

How does MUSCIMA++ compare to existing datasets? Given that individual notes are split into primitives and other ground truth policies, to obtain a fair comparison, we should subtract the stem count, letters and texts, and the `measure_separator` symbols. Some sharps and flats are also part of key signatures, and numerals are part of time signatures, which again leads to two symbols where other datasets may only annotate one. These subtractions bring us to a more directly comparable symbol count of about 57000.

A note on naming conventions: CVC-MUSCIMA refers to the set of binary images described in **??**. MUSCIMA++ 1.0 refers to the symbol-level ground truth of the selected 140 pages. (Future versions of MUSCIMA++ may contain more types of data, such as MEI-encoded musical information, or other representations of the encoded musical semantics.) The term "MUSCIMA++ images" refers to those 140 undistorted images from CVC-MUSCIMA that have been

[31] The first author of (Donald Byrd and Jakob Grue Simonsen, 2015), Donald Byrd, has been maintaining a list of interesting music notation events. See: `http://homes.soic.indiana.edu/donbyrd/InterestingMusicNotation.html`

[32] We find *adherence to topological standards* to be a more general term that describes this particular class of difficulties.

[33] With the exception of writer 49, who is represented in 4 images due to a mistake in the distribution workflow that was only discovered after the image was already annotated.

Table 3: Symbol frequencies in MUSCIMA++

Symbol	Count	Symbol (cont.)	Count
stem	21416	16th_flag	495
notehead-full	21356	16th_rest	436
ledger_line	6847	g-clef	401
beam	6587	grace-notehead-full	348
thin_barline	3332	f-clef	285
measure_separator	2854	other_text	271
slur	2601	hairpin-decr.	268
8th_flag	2198	repeat-dot	263
duration-dot	2074	tuple	244
sharp	2071	hairpin-cresc.	233
notehead-empty	1648	half_rest	216
staccato-dot	1388	accent	201
8th_rest	1134	other-dot	197
flat	1112	time_signature	192
natural	1089	staff_grouping	191
quarter_rest	804	c-clef	190
tie	704	trill	179
key_signature	695	*All letters*	*4072*
dynamics_text	681	*All numerals*	*594*

annotated so far.

3.3 Designated test sets

To give some basic guidelines on comparing trainable systems over the dataset, we designate some images to serve as a test set. One can always use a different train-test split; however, we believe our choice is balanced well. Similar to Jorge Calvo-Zaragoza and Jose Oncina (2014), we provide a *user-independent* test set, and a *user-dependent* one. Each of these contains 20 images, one for each CVC-MUSCIMA page. However, they differ in how each of these 20 is chosen from the 7 available versions, with respect to the individual writers.

The **user-independent** test set evaluates how the system handles data form previously unseen writers. The images are split so that the 2 or 3 MUSCIMA++ images from any particular writer are either all in the training portion, or all in the test portion of the data.

The **user-dependent** test set, to the contrary, contains at most one image from each writer in its set of the 20 CVC-MUSCIMA pages. For each writer in the user-dependent test set, there is also at least one image in the training data. This allows experimenting with at least some amount of user adaptation.

Furthermore, both test sets are chosen so that the annotators are represented as uniformly as possible, so that the evaluation is not biased towards the idiosyncracies of a particular annotator.[34]

[34]The choice of test set images is provided as supplementary data, together with the dataset itself.

3.4 MUSCIMA++ ground truth

How does MUSCIMA++ implement the notation graph?

We define a fine-grained vocabulary of musical symbols as the graph vertices, and we define relationships between symbols to be expressed as unlabeled directed edges. A "proto-grammar" then defines which ordered pairs of symbols are allowed to participate in a relationship.[35]

Each vertex (symbol) furthermore has a set of attributes. These are a superset of the primitive attributes in (Miyao and Haralick, 2000) – for a symbol, we encode:

- its bounding box with respect to the page,
- its label (`notehead`, `sharp`, `g-clef`, etc.),
- its mask: exactly which pixels in the bounding box belong to this symbol?

The mask is important especially for beams, as they are often slanted and so their bounding box overlaps with other symbols (esp. stems and parallel beams). Slurs also often have this problem. Annotating the mask enables us to build an accurate model of actual symbol shapes.

The symbol set includes what Michael Droettboom and Ichiro Fujinaga (2004), Pierfrancesco Bellini et al. (2007) and Donald Byrd and Jakob Grue Simonsen (2015) would describe as a mix of low-level symbols as well as high-level symbols, without explicitly labeling the symbols as either. Instead of trying to categorize symbols according to whether they carry semantics or not, we chose to express the high- vs. low-level dichotomy through the rules for forming relationships. This leads to "layered" annotation. For instance, a 3/4 time signature is annotated using three symbols: a `numeral_3`, `numeral_4`, and a `time_signature` symbol that has outgoing relationships to each of the numerals involved. In the current version of MUSCIMA++ (0.9, beta) we do *not* annnotate invisible symbols (e.g. implicit tuplets). Each symbol has to have at least one foreground pixel.

The policy for making decisions that are arbitrary with respect to the information content, as discussed in Subsec. 3.1, was set to stay as close as possible to the *written page*, rather than the semantics. If this principle was in conflict with the requirement for both replayability and reprintability introduced in 1, a symbol class was added to the vocabulary to capture the requisite meaning. Examples are the `key_signature`, `time_signature`, `tuple`, or `measure_separator`. These second-layer symbols are often composite, but not necessarily so: for instance, a single `sharp` can also form

[35]The most complete annotation guidelines detailing what the symbol set is and how to deal with individual notations are available online: `http://muscimarker.readthedocs.io/en/develop/instructions.html`

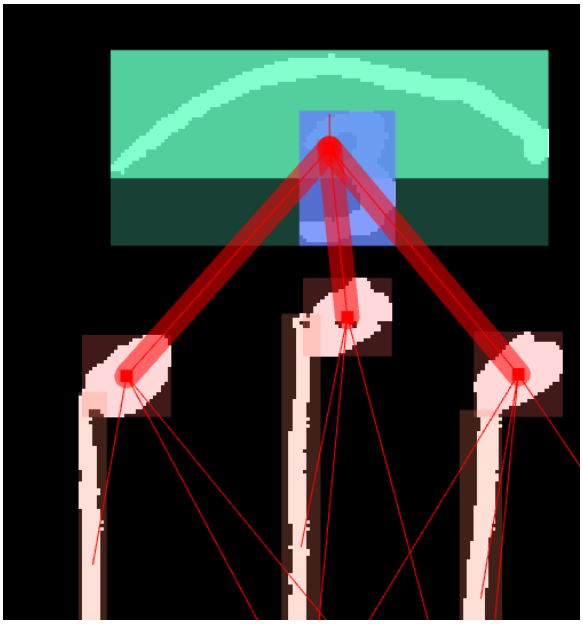


Figure 12: Two-layer annotation of a triplet. Highlighted symbols are `numeral_3`, `tuple_bracket/line`, and the three noteheads that form the triplet. The `tuple` symbol itself, to which the noteheads are connected, is the darker rectangle encompassing its two components; it has relationships leading to both of them (not highlighted).

a `key_signature`, or a `measure_separator` is expressed by a single `thin_barline`. An example of this structure for is given in Figure 12.

While we take care to define relationships so that the result is a Directed Acyclic Graph (DAG), there is no hard limit on the maximum oriented path length. However, in practice, it very rarely goes deeper than 3, as the path in the `tuple` example leading from a `notehead` to the `tuple` and then to its constitutent `numeral_3` or `tuple_bracket/line`, and in most cases, this depth is at most 2.

On the other hand, we do not break down symbols that consist of multiple connected components, unless it is possible that these components can be seen in valid music notation in various configurations (e.g., an empty notehead may show up with a stem, without one, with multiple stems when two voices share pitch,[36] or may share stem with others). A bass clef dot, for instance, cannot show up without the rest of the bass clef, and vice versa; on the other hand, a single repeat spanning multiple staves may have a variable number of repeat dots. This is a different policy from Miyao & Haralick (Miyao and Haralick, 2000), who split e.g. the `repeat_measure` "percent sign" into three primitives.

We do not define a `note` symbol. Notes are hard to pin down on the written page: in the traditional understanding of what is a "note" symbol (Ana Rebelo, 2012; Jorge Calvo-Zaragoza and Jose Oncina, 2014; Michael Droettboom and Ichiro Fujinaga, 2004), they consist of multiple primitives (notehead and stem and beams or flags), but at the same time, multiple notes can *share* these primitives, including noteheads – the relationship between high- and low-level symbols has in general an `m:n` cardinality. Another high-level symbol may be a key signature, which can consist of multiple sharps or flats. It is not clear how to annotate notes. If we follow the "semantics" criterion for distinguishing between the low- and high-level symbol description of the written page, should e.g. an accidental be considered a part of the note, because it directly influences its pitch?[37]

The policy for relationships was to make noteheads independent. That is: as much as possible of the semantics of a note corresponding to a notehead can be inferred based on the explicit relationships that pertain to this particular notehead. However, this ideal is not fully implemented in the current version MUSCIMA++ (0.9, beta) , and possibly cannot be reasonably reached, given the rules of music notation. While it is possible to infer duration from the current annotation (with the exception of implicit tuples), not so much pitch. First of all, one would need to add staffline and staff objects and link the noteheads to the staff. This is not explicitly done in MUSCIMA++ (0.9, beta) , but given the staff removal ground truth included with the annotated CVC-MUSCIMA images, it should be merely a technical step that does not require much manual annotation. The other missing piece of the pitch puzzle are assignment of notes to measures, and precedence relationships. Precedence relationships need to include (a) notes, to capture effects of accidentals at the measure scope; (b) clefs and key signatures, which can change within one measure,[38] so it is not sufficient to attach them to measures.

3.5 Available tools

In order to make using the dataset easier, we provide two software tools under an open license.

First, the `musicma` Python 3 package[39] implements the MUSCIMA++ data model, which can parse the dataset and enables manipulating the data further (such as assembling the related primitives into notes, to provide a comparison to the existing datasets with different symbol sets).

Second, we provide the `MUSCIMarker` application.[40] This is the annotation interface, and can visualize the data.

[36] As seen in page 20 of CVC-MUSCIMA.

[37] We would go as far as to say that it is inadequate to try marking "note" graphical objects in the musical score. A note is a basic unit of *music*, but it is not a unit of music *notation*. Music notation *encodes* notes, it does not *contain* them.

[38] Even though this does not occur in CVC-MUSCIMA, it does happen, as illustrated e.g. by Fig. 8 in (Donald Byrd and Jakob Grue Simonsen, 2015).

[39] https://github.com/hajicj/muscima

[40] https://github.com/hajicj/MUSCIMarker

Figure 13: MUSCIMarker 1.1 interface. Tool selection on the left; file controls on the right. Highlighted relationships have been selected. (Last bar of from MUSCIMA++ image `W-35_N-08`.)

3.6 Annotation process

Annotations were done using custom-made MUSCIMarker open-source software, version 1.1.[41] The annotators worked on symbols-only CVC-MUSCIMA images, which allowed for more efficient annotation. The interface used to add symbols consisted of two tools: a background-ignoring lasso selection, and connected component selection.[42] A screenshot of the annotation interface is in Fig. 13.

As annotation was under way, due to the prevalence of first-try inaccuracies, we added editing tools that enabled annotators to "fine-tune" the symbol shape by adding or removing arbitrary foreground regions to the symbol's mask. Adding symbol relationships was done by another lasso selection tool. A significant speedup was also achieved by providing a rich set of keyboard shortcuts. The effect was most acutely felt in the keyboard interface for assigning classes to annotated symbols, as the class list is rather extensive.

There were seven annotators working on MUSCIMA++. Four of these are professional musicians, three are experienced amateur musicians. The annotators were asked to complete three progressively more complex training examples. There were no IT skills required (we walked them through MUSCIMarker installation and usage, the odd troubleshooting was done through the TeamViewer[43] remote interface). Two of these were single measures, for basic familiarization with the user interface; the third was a full page, to ensure understanding of the majority of notation situations. Based on this "training round", we have also further refined the ground truth definitions.

As noted above, each annotator completed one image of each of the 20 CVC-MUSCIMA pages. The work was dispatched to annotators in four packages of 5 images each, one package at a time. After each package submission by an annotator, we checked for correctness. Automated validation was implemented to check for "impossible" or missing relationships (e.g., a stem and a beam should not be connected, a grace notehead has to be connected to a "normal" notehead). However, there was still need for manual checks and manually correcting mistakes found in auto-validation, as the validation itself was an "advisory voice" to highlight questionably annotated symbols, not an authoritative response.

Manually checking each submitted file also allowed for continuous annotator training and filling in "blind spots" in the annotation guidelines (such as specifying how to deal with *volta* signs, or tuples). Some notation events were simply not anticipated (e.g., mistakes that CVC-MUSCIMA writers made in transcription or non-standard symbols). Feedback was provided individually after each package was submitted and checked. This feedback was the main mechanism for continual training. Requests for clarifications of guidelines for situations that proved problematic were also disseminated to the whole group.

At first, the quality of annotations was inconsistent: some annotators performed well, some poorly. Some required extensive feedback. Thanks to the continuous communication and training, the annotators improved, and the third and fourth packages required relatively minor corrections. Overall, however, only one annotator submitted work at a quality that required practically no further changes in quality control. Differences in annotator speed did not equalize as much as annotation correctness.

Finally, after collecting annotations for all 140 images, we performed a second quality control round, this time with further automated checks. We checked for disconnected symbols, and for symbols with suspiciously sparse masks (a symbol was deemed suspicious if more than 0.07 of the foreground pixels in its bounding box were not marked as part of any symbol at all). This second round of quality control uncovered yet more errors, and we also fixed clearly inaccurate markings (e.g., if a significant amount of stem-only pixels was marked as part of a beam).

On average throughout the annotations, the fastest annotator managed to mark about 6 symbols per minute, or one per 10 seconds; the next fastest came at 4.6 symbols per minute. Two slowest annotators clocked in around 3.4 symbols per minute. The average speed overall was 4.3 symbols per minute, or one per 14 seconds. The "upper bound" on annotation speed was established by the author (who is intimately familiar with best ways of using the MUSCIMarker annotation tool) to be 8.9 objects per minute (one in 6.75 seconds). These numbers are computed over the whole time spent annotating, so they include

[41] https://lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-1850, ongoing development at https://github.com/hajicj/MUSCIMarker

[42] Basic usage is described in the MUSCIMarker tutorial: http://muscimarker.readthedocs.io/en/develop/tutorial.html

[43] https://www.teamviewer.com

the periods during which annotators were marking relationships and checking their work: in other words, if you plan to extend the dataset with a comparable annotator workforce, you can expect an average page of about 600 symbols to take about 2.5 hours.

Annotating the dataset using the process detailed above took roughly 400 hours of work; the "quality control" correctness checks took an additional 100 – 150. The second, more complete round of quality control took roughly 60 – 80 hours, or some 0.5 hours per image.[44]

3.7 Inter-annotator agreement

In order to assess (a) whether the annotation guidelines are well-defined, and (b) the extent to which we can trust annotators, we conducted a test: all seven annotators were given the same image to annotate, and we measured inter-annotator agreement. Inter-annotator agreement does explicitly not decouple the factors (a) and (b). However, given that the overall expected level of ambiguity is relatively low, and given the learning curve along which the annotators were moving throughout their work, which would as be hard to decouple from genuine (a)-type disagreement, we opted to not expend resources on annotators re-annotating something which they had already done, and therefore cannot provide exact intra-annotator agreement data.

Another use of inter-annotator agreement is to provide an upper bound on system performance. If a system performs better than average inter-annotator agreement, it may be overfitting the test set. (On the other hand, it may have merely learned to compensate for anntoator mistakes – more analysis is needed before concluding that the system overfits. But it is a useful warning: one should investigate unexpectedly high performance numbers.)

3.7.1 Computing agreement

In order to evaluate the extent to which two annotators agree on how a given image should be annotated, we perform two steps:

- Align the annotated object sets against each other,
- Compute the macro-averaged f-score over the aligned object pairs.

Objects that have no counterpart contribute 0 to both precision and recall.

Alignment was done in a greedy fashion. For symbol sets S, T, we first align each $t \in T$ to the $s \in S$ with the highest pairwise f-score $F(s,t)$, then vice versa align each $s \in S$ to the $t \in T$ with the highest pairwise f-score. Taking the intersection, we then get symbol pairs s, t such that they are each other's "best friends" in terms of f-score. The symbols that do not have such a clear counterpart are left out of the alignment. Furthermore, symbol pairs that are not labeled with the same symbol class are removed from the alignment as well.

When breaking ties in the pairwise matchings (from both directions), symbol classes $c(s), c(t)$ are used. If $F(s,t_1) = F(s,t_2)$, but $c(s) = c(t_1)$ while $c(s) \neq c(t_2)$, then (s,t_1) is taken as an alignment candidate instead of (s,t_2). (If both t_1 and t_2 have the same class as s, then then the tie is broken randomly. In practice, this would be extremely rare and would not influence agreement scores very much.)

[44] For the sake of completeness, implementing MUSCIMarker took about 600 hours, including the learning curve for the GUI framework.

Table 4: Inter-annotator agreement

Setting	macro-avg. f-score
noQC-noQC (inter-annot.)	0.89
noQC-withQC (self)	0.93
withQC-withQC (inter-annot.)	0.97

3.7.2 Agreement results

The resulting f-scores are summarized in Table 4. We measured inter-annotator agreement both before and after quality control (noQC-noQC and withQC-withQC), and we also measured the extent to which quality control changed the originally submitted annotations (noQC-withQC). Tentatively, the post-QC measurements reflect the level of genuine disagreement among the annotators about how to lead the boundaries of objects in intersections and the inconsistency of QC, while the pre-QC measurements also measures the extent of actual mistakes that were fixed in QC.

Ideally, the task of annotating music notation symbols is relatively unambiguous. Legitimate sources of disagreement lie in two factors: unclear symbol boundaries in intersections, and illegible handwriting. For relationships, ambiguity is mainly in polyphonic scores, where annotators had to decide how to attach noteheads from multiple voices to crescendo and decrescendo hairpin symbols. However, after quality control, there were 689 – 691 objects in the image and 613 – 637 relationships, depending on which annotator we asked. This highlights the limits of both the annotation guidelines and QC: the ground truth is probably not entirely unambiguous, so various configurations passed QC, and additionally the QC process itself allows for human error. (If we could really automate infallible QC, we would also have solved OMR!)

At the same time, as seen in Table 4, the two-round quality control process apparently removed nearly four fifths of all disagreements, bringing the withQC inter-annotator f-score of 0.97 from a noQC f-score of 0.89. On average, quality control introduced *less* change than was originally between individual annotators. This statistic seems to suggest that the withQC results are somewhere in the "center" of the space of submitted annotations, and therefore the quality control process really leads to more accurate annotation instead of merely

distorting the results in its own way.

We can conclude that the annotations, using the quality control process, is quite reliable, even though slight mistakes may remain.

3.8 Known limitations

The MUSCIMA++ dataset is not perfect, as is always the case with extensive human-annotated datasets. In the interest of full disclosure and managing expectations, we list the known issues.

Annotators also might have made mistakes that slipped both through automated validation and manual quality control. In automated validation, there is a tradeoff between catching errors and false alarms: events like multiple stems per notehead happen even in the limited set of 20 pages of MUSCIMA++. In the same vein, although we did implement automated checks for highly inaccurate annotations, they only catch some of the problems as well, and our manual quality control procedure also relies on inherently imperfect human judgment. All in all, the data is not perfect. With limited man-hours, there is always a tradeoff between quality and scale.

The CVC-MUSCIMA dataset has had staff lines removed automatically with very high accuracy, based on a precise writing and scanning setup (using a standard notation paper and a specific pen across all 50 writers). However, there are still some errors in staff removal: sometimes, the staff removal algorithm took with it some pixels that were also legitimate part of a symbol. This manifests itself most frequently with stems.

The relationship model is rather basic. Precedence and simultaneity relationships are not annotated, and stafflines and staves are not among the annotated symbols, so notehead-to-staff assignment is not explicit. Similarly, notehead-to-measure assignment is also not explicitly marked. This is a limitation that so far does not enable inferring pitch from the ground truth. However, much of this should be obtainable automatically, from the annotation that is available and the CVC-MUSCIMA staff removal ground truth.

There are also some outstanding technical issues in the details of how the bridge from graphical expression to interpretation ground truth is designed. For example, there is no good way to encode a 12/8 time signature. The "1" and "2" would currently be annotated as separate numerals, and the fact that they belong together to encode the number 12 is not represented explicitly: one would have to infer that from knowledge of time signatures. A potential fix is to introduce a "numeric_text" symbol as an intermediary between "time_signature" and "numeral_X" symbols, similarly to various "(some)_text" symbols that group "letter_X" symbols. Another technical problem is that the mask of empty noteheads that lie on a ledger line includes the part of the ledger line that lies within the notehead.

Finally, there is no good policy on symbols broken into two at line breaks. They are currently handled as two separate symbols.

4 Data-driven evaluation

There is presently no (publicly available) way of comparing various OMR tools and assessing their performance. While it has been argued that OMR can go far even in the absence of such standards (Donald Byrd and Jakob Grue Simonsen, 2015), the lack of benchmarks and difficulty of evaluation has been noted on multiple occasions (Michael Droettboom and Ichiro Fujinaga, 2004; Victor Padilla et al., 2014; Baoguang Shi et al., 2015). The need for end-to-end system evaluation (at the final level of OMR when musical content is reconstructed and made available for further processing), is most pressing when comparing against commercial systems such as PhotoScore,[45] SmartScore[46] or SharpEye[47]: these typically perform as "black boxes", so evaluating on the level of individual symbols requires a large amount of human effort for assessing symbols and their locations, as done by Bellini et al. (Pierfrancesco Bellini et al., 2007) or Sapp (Sapp, 2013). Manually counting errors, however, is too time-consuming for iterative experimentation. OMR would greatly benefit from an automated evaluation metric that could guide development and reduce the price of conducting evaluations. However, an automated metric for OMR evaluation needs itself to be evaluated: does it really rank as better systems that *should* be ranked better?

In this section,[48] we describe our robust, cumulative, data-driven methodology for *creating* one. We collect human preference data that can serve as a gold standard for comparing MusicXML automated evaluation metrics, mirroring how the BLEU metric and its derivatives has been established as an evaluation metric for the similarly elusive task of assessing machine translation based on agreement with human judgements (Papineni et al., 2002). This "evaluating evaluation" approach is inspired by machine translation (MT), a field where comparing outputs is also notoriously difficult The Workshop of Statistical Machine Translation competition (WMT) even has a separate Metrics track (Callison Burch et al., 2010; Bojar et al., 2011; Matouš Macháček and Ondřej Bojar, 2014), where automated MT metrics are evaluated against human-collected evaluation results, and there is ongoing research (Bojar et al., 2011; Matouš Macháček and Ondřej Bojar, 2015) to design a better metric than the current standards such as BLEU (Papineni et al., 2002) or Meteor (Lavie and Agarwal, 2007).

To collect cost-to-correct estimates for various notation errors, we generate a set of synthetically dis-

[45] http://www.neuratron.com/photoscore.htm

[46] http://www.musitek.com/index.html

[47] http://www.visiv.co.uk

[48] This section is taken from our publication (Hajič jr. et al., 2016); all the segments of (Hajič jr. et al., 2016) used here describe work done solely by the author.

torted "recognition outputs" from a set of equally synthetic "true scores". Then, annotators are shown examples consisting of a true score and a pair of the distorted scores, and they are asked to choose the simulated recognition output that would take them less time to correct. Assuming that the judgment of (qualified) annotators is considered the gold standard, the following methodology then can be used to assess an automated metric:

1. Collect a corpus of annotator judgments to define the expected gold-standard behavior,
2. Measure the agreement between a proposed metric and this gold standard.

This methodology is nothing surprising; in principle, one could machine-learn a metric given enough gold-standard data. However: how to best design the gold-standard data and collection procedure, so that it encompasses what we in the end want our application (OMR) to do? How to measure the quality of such a corpus – given a collection of human judgments, how much of a gold standard is it?

In the rest of this section, we describe the corpus of collected human judgments, estimate the extent to which the corpus can be trusted, and propose and assess some automated end-to-end evaluation metrics.

4.1 Test case corpus

We collect a corpus C of *test cases*. Each test case $c_1 \dots c_N$ is a triplet of music scores: an "ideal" score I_i and two "mangled" versions, $P_i^{(1)}$ and $P_i^{(2)}$, which we call *system outputs*. We asked our K annotators $a_1 \dots a_K$ to choose the less mangled version, formalized as assigning $r_a(c_i) = -1$ if they preferred $P_i^{(1)}$ over $P_i^{(2)}$, and $+1$ for the opposite preference. The term we use is to "rank" the predictions. When assessing an evaluation metric against this corpus, the test case rankings then constrain the space of well-behaved metrics.[49]

The exact formulation of the question follows the "cost-to-correct" model of evaluation of (Pierfrancesco Bellini et al., 2007):

"Which of the two system outputs would take you less effort to change to the ideal score?"

4.1.1 What is in the test case corpus?

We created 8 ideal scores and derived 34 "system outputs" from them by introducing a variety of mistakes in a notation editor. Creating the system outputs manually instead of using OMR outputs has the obvious disadvantage that the distribution of error types does not reflect the current OMR state-of-the-art. On the other hand, once OMR systems change, the distribution of corpus errors becomes obsolete anyway. Also, we create errors for which we can assume the annotators have a reasonably accurate estimate of their own correction speed, as opposed to OMR outputs that often contain strange and syntactically incorrect notation, such as isolated stems. Nevertheless, when more annotation manpower becomes available, the corpus should be extended with a set of actual OMR outputs.

The ideal scores (and thus the derived system outputs) range from a single whole note to a "pianoform" fragment or a multi-staff example. The distortions were crafted to cover errors on individual notes (wrong pitch, extra accidental, key signature or clef error, etc.: micro-errors on the semantic level in the sense of (Michael Droettboom and Ichiro Fujinaga, 2004; Pierfrancesco Bellini et al., 2007)), systematic errors within the context of a full musical fragment (wrong beaming, swapping slurs for ties, confusing staccato dots for noteheads, etc.), short two-part examples to measure the tradeoff between large-scale layout mistakes and localized mistakes (e.g., a four-bar two-part segment, as a perfect concatenation of the two parts into one vs. in two parts, but with wrong notes) and longer examples that constrain the metric to behave sensibly at larger scales.

Each pair of system outputs derived from the same ideal score forms a test case; there are 82 in total. We also include 18 control examples, where one of the system outputs is identical to the ideal score. A total of 15 annotators participated in the annotation, of whom 13 completed all 100 examples; however, as the annotations were voluntary, only 2 completed the task twice for measuring intra-annotator agreement.

4.1.2 Collection Strategy

While Pierfrancesco Bellini et al. (2007) define how to count individual errors at the level of musical symbols, assign some cost to each kind of error (miss, add, fault, etc.) and define the overall cost as composed of those individual costs, our methodology does not assume that the same type of error has the same cost in a different *context*, or that the overall cost can be computed from the individual costs: for instance, a sequence of notes shifted by one step can be in most editors corrected simultaneously (so, e.g., clef errors might not be too bad, because the entire part can be transposed together).

Two design decisions of the annotation task merit further explanation: why we ask annotators to compare examples instead of rating difficulty, and why we disallow equality.

Ranking. The practice of ranking or picking the best from a set of possible examples is inspired by machine translation: Callison-Burch et al. have shown that people are better able to agree on which proposed translation is better than on how good or bad individual translations are (Callison Burch et al., 2007). Furthermore, ranking does not require introducing a cost metric in the first place. Even a simple 1-2-3-4-5 scale has this

[49] We borrow the term "test case" from the software development practice of unit testing: test cases verify that the program (in our case the evaluation metric) behaves as expected on a set of inputs chosen to cover various standard and corner cases.

problem: how much effort is a "1" on that scale? How long should the scale be? What would the relationship be between short and long examples?

Furthermore, this annotation scheme is fast-paced. The annotators were able to do all the 100 available comparisons within 1 hour. Rankings also make it straightforward to compare automated evaluation metrics that output values from different ranges: just count how often the metric agrees with gold-standard ranks using some measure of monotonicity, such as Spearman's rank correlation coefficient.

No equality. It is also not always clear which output would take less time to edit; some errors genuinely are equally bad (sharp vs. flat). These are also important constraints on evaluation metrics: the costs associated with each should not be too different from each other. However, allowing annotators to explicitly mark equality risks overuse, and annotators using *underqualified* judgment. For this first experiment, therefore, we elected not to grant that option; we then interpret disagreement as a sign of uncertainty and annotator uncertainty as a symptom of this genuine tie.

4.2 How gold is the standard?

We need to assess the quality of the test case rankings we have gathered. Similarly to our analysis of the MUSCIMA++ dataset quality, we turn to inter-annotator agreement.

All annotators ranked the control cases correctly, except for one instance. However, this only accounts for elementary annotator failure and does not give us a better idea of systematic error present in the experimental setup. In other words, we want to ask the question: if all annotators are performing to the best of their ability, **what level of uncertainty should be expected under the given annotation scheme?** (For the following measurements, the control cases have been excluded.)

Normally, inter-annotator agreement is measured: if the task is well-defined, i.e., if a gold standard *can* exist, the annotators will tend to agree with each other towards that standard. However, usual agreement metrics such as Cohen's κ or Krippendorf's α require computing *expected agreement*, which is difficult when we do have a subset of examples on which we do *not* expect annotators to agree but cannot *a priori* identify them. We therefore start by defining a simple agreement metric L. Recall:

- C stands for the corpus, which consists of N examples $c_1 \dots c_N$,
- A is the set of K annotators $a_1 \dots a_K$, $a, b \in A$;
- r_a is the *ranking function* of an annotator a that assigns +1 or -1 to each example in c,

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

This is simply the proportion of cases on which a and b agree: if they disagree, $r_a(c) + r_b(c) = 0$. However, we expect the annotators to disagree on the genuinely uncertain cases, so some disagreements are not as serious as others. To take the existence of legitimate disagreement into account, we modify $L(a,b)$ to weigh the examples according to how certain the other annotators $A \setminus \{a,b\}$ are about the given example. We define weighed agreement $L_w(a,b)$:

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

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

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

This way, it does not matter if a and b disagree on cases where no one else agrees either, but if they disagree on an example where there is strong consensus, it should bring the overall agreement down. Note that while maximum achievable $L(a,b)$ is 1 for perfectly agreeing annotators (i.e., all the sum terms equal to 1), because $w(c) \leq 1$, the maximum achievable $L_w(a,b)$ will be less than 1, and furthermore depends on the choice of a and b: if we take notoriously disagreeing annotators away from the picture, the weights will increase overall. Therefore, we finally adjust $L_w(a,b)$ to the proportion of maximum achievable $L_w(a,b)$ for the given (a,b) pair, which is almost the same as $L_w(a,a)$ with the exception that b *must also be excluded from computing the weights*. We denote this maximum as $L^*_w(a,b)$, and the adjusted metric $\hat{L}_w$ is then:

$$\hat{L}_w(a,b) = L_w(a,b) / L^*_w(a,b)$$

This metric says: "What proportion of achievable weighed agreement has been actually achieved?" The upper bound of $\hat{L}_w$ is therefore 1.0 again; the lower bound is agreement between two randomly generated annotators, with the humans providing the consensus.

The resulting pairwise agreements, with the lower bound established by averaging over 10 random annotators, are visualized in Fig. 14. The baseline agreement $\hat{L}_w$ between random annotators weighed by the full human consensus was close to 0.5, as expected. There seems to be one group of annotators relatively in agreement (green and above, which means adjusted agreement over 0.8), and then several individuals who disagree with everyone – including among themselves (lines 6, 7, 8, 11, 12, 14).

Interestingly, most of these "lone wolves" reported significant experience with notation editors, while the group more in agreement not as much. We suspect this is because with increasing notation editor experience, users develop a personal editing style that makes certain actions easier than others by learning a *subset* of the "tricks" available with the given editing tools – but each user learns a different subset, so agreement on the

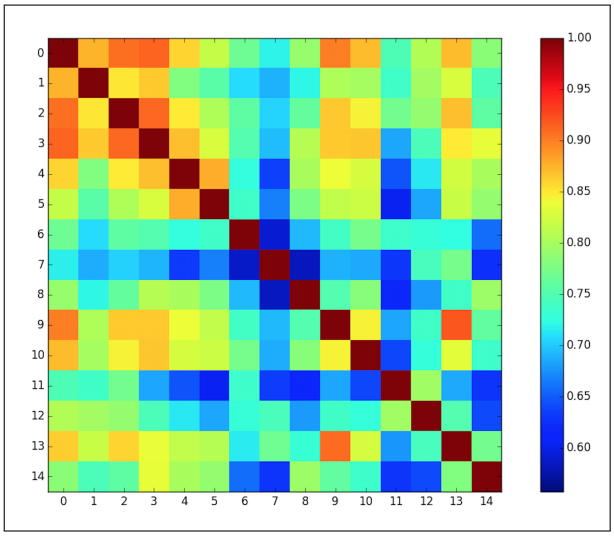


Figure 14: Weighed pairwise agreement. The cell $[a, b]$ represents $\hat{L}_w(a, b)$. The scale goes from the average random agreement (ca. 0.55) up to 1.

relative editing cost suffers. To the contrary, inexperienced users might not have spent enough time with the editor to develop these habits.

4.3 Assessing some metrics

We illustrate how the test case ranking methodology helps analyze these rather trivial automated MusicXML evaluation metrics:

1. Levenshtein distance of XML canonization (**c14n**),
2. Tree edit distance (**TED**),
3. Tree edit distance with `<note>` flattening (**TEDn**),
4. Convert to LilyPond + Levenshtein distance (**Ly**).

c14n. Canonize the MusicXML file formatting and measure Levenshtein distance. This is used as a trivial baseline.

TED. Measure Tree Edit Distance on the MusicXML nodes. Some nodes that control auxiliary and MIDI information (`work`, `defaults`, `credit`, and `duration`) are ignored. Replacement, insertion, and deletion all have a cost of 1.

TEDn. Tree Edit Distance with special handling of `note` elements. We noticed that many errors of TED are due to the fact that while deleting a note is easy in an editor, the edit distance is higher because the `note` element has many sub-nodes. We therefore encode the notes into strings consisting of one position per `pitch`, `stem`, `voice`, and `type`. Deletion cost is fixed at 1, insertion cost is 1 for non-note nodes, and 1 + length of code for notes. Replacement cost between notes is the edit distance between their codes; replacement between a note and non-note costs 1 + length of code; between non-notes costs 1.

Metric	r_s	$\hat{r}_s$	ρ	$\hat{\rho}$	τ	$\hat{\tau}$
c14n	0.33	0.41	0.40	0.49	0.25	0.36
TED	0.46	0.58	0.40	0.50	0.35	0.51
TEDn	**0.57**	**0.70**	0.40	0.49	**0.43**	**0.63**
Ly	0.41	0.51	0.29	0.36	0.30	0.44

Table 5: Measures of agreement for some proposed evaluation metrics.

Ly. The LilyPond[50] file format is another possible representation of a musical score. It encodes music scores in its own LaTeX-like language. The first bar of the "Twinkle, twinkle" melody would be represented as `d'8[ d'8] a'8[ a'8] b'8[ b'8] a'4 |` This representation is much more amenable to string edit distance. The **Ly** metric is Levenshtein distance on the LilyPond import of the MusicXML system output files, with all whitespace normalized.

For comparing the metrics against our gold-standard data, we use nonparametric approaches such as Spearman's r_s and Kendall's τ, as these evaluate monotonicity without assuming anything about mapping values of the evaluation metric to the $[-1, 1]$ range of preferences . To reflect the "small-difference-for-uncertain-cases" requirement, however, we use Pearson's ρ as well (Matouš Macháček and Ondřej Bojar, 2014). For each way of assessing a metric, its maximum achievable with the given data should be also estimated, by computing how the metric evaluates the consensus of one group of annotators against another. We randomly choose 100 splits of 8 vs 7 annotators, compute the average preferences for the two groups in a split and measure the correlations between the average preferences. The expected upper bounds and standard deviations estimated this way are:

- $r_s^* = 0.814$, with standard dev. 0.040
- $\rho^* = 0.816$, with standard dev. 0.040
- $\tau^* = 0.69$, with standard dev. 0.045

We then define $\hat{r_s}$ as $\frac{r_s}{r_s^*}$, etc. Given a cost metric $\mathcal{L}$, we get for each example $c_i = (I_i, P_i^{(1)}, P_i^{(2)})$ the cost difference $\ell(c_i) = \mathcal{L}(I_i, P_i^{(1)}) - \mathcal{L}(I_i, P_i^{(2)})$ and pair it with the gold-standard consensus $r(c_i)$ to get pairwise inputs for the agreement metrics.

The agreement of the individual metrics is summarized in Table 5. When developing the metrics, we did *not* use the gold-standard data against which metric performance is measured here; we used only our own intuition about how the test cases should come out.

4.4 Lessons Learned

Our results weakly suggest that the central assumption of a single ground truth for preferences among a set of system outputs is weaker with increasing annotator experience. This is not an encouraging result for automating cost-to-correct evaluation..

[50] http://www.lilypond.org

To make the methodology more robust, we recommend:

- Explicitly control for experience level; do not assume that more annotator experience is necessarily desirable.
- Measure actual cost-to-correct (in time and interface operations) through a notation editor, to verify how much human *estimation* of this cost can be relied on.
- Develop models for computing expected agreement for data where the annotations may legitimately be randomized (the "equally bad" cases). Once expected agreement can be computed, we can use more standard agreement metrics.

The usefulness of the test case corpus for developing automated evaluation metrics was clear: the TEDn metric that outperformed the others by a large margin was developed through analyzing the shortcomings of the TED metric on individual test cases (before the gold-standard data had been collected). As Szwoch (2008) suggested, modifying the representation helped.

However, if enough human judgments are collected, it should even be possible to sidestep the difficulties of hand-crafting an evaluation metric through machine learning; we can for instance try learning the insertion, deletion, and replacement costs for individual MusicXML node types.

5 Plan of Recognition Experiments

We plan two complementary lines of experimentation: notation syntax-aware offline handwritten OMR, and augmenting it by including the audio modality.

5.1 Syntax-aware OMR

For offline handwritten OMR, the key problem is segmentation. While staff removal is the logical first step, As illustrated by Fig. 8, post-staff removal connected components – while potentially a useful heuristic for initialization – are not a reliable indicator of segmentation.

However, we can exploit music notation syntax to jointly solve segmentation, symbol classification, and reconstruction. Once we are certain that a symbol exists at some location, the uncertainty in its surroundings is greatly reduced. For instance, once we find a notehead, it is quite certain that a vertical line starting in its close vicinity will be a stem, not a barline. The challenge is then to design a model that can take the syntax of music notation into account, while working with handwritten images: a parser for music notation.

This is by no means a new idea. However, very few syntactic models of music notation have so far been actually developed, and with the exception of the fuzzy system of Rossant and Bloch (2007), we are not aware of a system that would not rely on topological heuristics, such as "a notehead is connected to a stem", which not hold in handwritten music (although at least in early music, the scribes are generally more consistent than in contemporary musical handwriting). Additionally, systems that do take musical syntax or other higher-order information (such as regular grouping of beats into measures) into account are often using it as a separate step after recognizing the individual symbols, so the syntactic constraints are not leveraged for improving the symbol recognition step (Ana Rebelo et al., 2013; Victor Padilla et al., 2014).

The project will focus on designing a joint model for musical symbol recognition and notation graph construction: in other words, a **parser of handwritten music notation.**

After establishing non-joint baselines for object detection such as Faster R-CNN (Shaoqing Ren et al., 2015) and post-hoc notation graph building, we will explore parser models that output a sequence of parser actions, with a sequential input reading mechanism.

The hurdles for this approach are:

- designing a tractable parsing algorithm that can deal with the notation graph,
- designing an appropriate input sequentialization mechanism,
- developing an oracle which unrolls training data into parser action sequences, so that the sequence-to-sequence relationship is learnable by available algorithms.

While there are many ways to design such a graph, the design implemented in the MUSCIMA++ dataset which (a) is a directed acyclic graph, (b) allows decomposition into many individual trees (rooted at noteheads), which only share leaves (e.g., a beam which multiple noteheads relate to). Therefore, the graph parser for music notation does not have to be too general, and there is hope it should be possible to generalize transition-based dependency parsing methods used in natural language processing for individual trees. The *swap* system of **?** for limited non-projective structures is especially interesting, as the swap operation mimics the Bag data structure of Bainbridge's notation construction engine (Bainbridge and Bell, 2003). However, there will be work on adapting these parsers to 2-dimensional input.

For transition-based parsers, a sequential reading mechanism for the input image needs to be designed. Here, we will draw inspiration from the skeleton-based approaches. Subsampling the input image to skeleton pixels simplifies the image greatly, somewhat mimics the sequential nature of the input (as the author wrote the score), and we have observed that there is almost no reconstruction error when labels assigned to skeleton pixels are dilated back to "fill in" the original foreground shapes. Furthermore, many more skeleton pixels can be discarded and skeleton segments can be classified as a whole, even though care needs to be taken for some handwriting styles not to be too zealous. Even so, a mechanism for walking through the chosen points of interest that chooses an ordering which does not hinder

the parser operations will still need to be designed.

The oracle for converting MUSCIMA++-like notation graphs to gold-standard parser action sequence will follow from the sequentialization approach.

5.2 Multimodal OMR

We see the potential of mOMR in the complementarity of the strong and weak points of the input modalities. The audio signal by itself does not carry sufficient information to be converted to a musical score – for instance, it cannot resolve the time signature (e.g.: 6/8 vs. 3/4), key signature, voice leading, clef changes, and other major readability issues. However, these components of a musical score are quite straightforward for image-only OMR systems, as they are isolated, well-defined symbols that are not involved in beamed groups or chords, and their locations are usually quite predictable. However, image-only systems face a greater challenge with the core notation symbols (noteheads, stems, beamed groups): ascertaining pitch and duration of notes, which is exactly the information that is present in the audio signal: frequency information for pitch, and energy information (onsets) to resolve individual notes from non-note symbols. Deep learning multimodal models have already demonstrated that they can, to some extent, capture and combine information from both modalities (Matthias Dorfer et al., 2016b,a).

We formulate the following incremental steps:

- The first step is to adapt multimodal models of Matthias Dorfer et al. (2016b) from live score following to multi-pitch transcription of polyphonic music. We could, of course, simply use existing transcription solutions. However, the visual signal can inform the transcription - for instance, the system can get a better estimate of how many pitches are probably sounding at a given point in time based on the number of noteheads in the corresponding segment of the score. An audio-only transcription model will be used as a baseline. This setting will bypass the problem with designing a differentiable objective function for reprintabile OMR.

- The second step is to also have the model align the piano-roll representation to the symbol data. In this step, we attempt to make the hidden trained relationships between the image and the pitch estimation explicitly into output. The most straightforward approach is incorporating visual attention.

- The third step is to modify the model to concurrently provide symbol location and classification explicitly. This involves incorporating a region-of-interest and symbol classifier above the attention model, and incorporating this additional learning signal into the model.

5.3 MUSCIMA++

Future work on the MUSCIMA++ dataset should include enlarging it with data from different sources than CVC-MUSCIMA that are closer to OMR application domains, if perhaps not as varied in terms of handwriting styles: expecially early manuscripts and manuscripts of contemporary composers.[51] The established annotation methodology, tooling, and especially the annotator team should make extending the dataset a relatively smooth operation.

A separate concern is extending the notation graph annotation to printed datasets. This is a technical issue: there are open-source tools for rendering music notation (LilyPond, MuseScore, LibMEI) that at some point have to explicitly record the information that the notation graph stores. "Hijacking" this process to remember this data and convert it to a notation graph is a serious software undertaking, but nevertheless not a *research* problem.

5.4 Evaluation

The work on evaluation (done prior to work on datasets) is inconclusive so far. The test case corpus needs enlargement, and new annotations are needed – both from newcomers, and from those who have participated in order to provide robust self-agreement statistics. However, as became clear during the investigation of appropriate ground truth formats, *cost-to-correct* evaluation of outputs in an interchange format such as MusicXML can be handled separately from *OMR accuracy* in recovering the notation graph, even though the latter is more relevant to applications.

Future work on evaluation will therefore focus first on evaluting the accuracy of how the notation graph is recovered. For lower-level metrics on symbol recognition, the ground truth defined for MUSCIMA++ enables using straightforward metrics once an alignment of the predicted objects to the ground truth is found. The notehead-centric structure of the graph then enables simple recovery of semantic information about the encoded notes, and allows for easy backtracking of errors in pitch and duration recovery to the original symbols.

The critical issue is searching for this alignment, as graph alignment with respect to an arbitrary scoring functon f is a computationally hard problem (NP-hard, in fact). However, the structure of the notation graph and some constraints on aligned symbol locations should help prune the search space to a manageable size, and the evaluation algorithm does not necessarily need to find an *optimal* alignment with respect to the evaluation metric – an approximation of the optimum should be sufficient, as long as the evaluated OMR system inputs are sufficiently distinct from each other, so that the approximations made during alignment cannot be responsible for differences in the re-

[51] We are in contact with several composers' estates, negotiating the licensing scheme for the contemporary data.

sulting evaluation metric values. (Given a good enough approximate graph alignment, we can also argue that if the ranking of two systems *could* be affected by the approximation, for all intents and purposes, the systems perform the same.) Some research into approximate graph alignment algorithms is needed, beyond tree edit distances mentioned in 4.3. An alternate approach is to improve the ad-hoc alignment used to assess inter-annotator agreement for MUSCIMA++ using min-cost max-flow algorithms for bipartite graphs. However, it is not clear how to incorporate relationships and local neighborhood similarity into the cost function.

6 Conclusions

Optical Music Recognition is a complex open problem that will have significant impact on musicology, preservation and dissemination musical heritage, the economics of music composition, and may open new avenues for music learning. It is also of personal interest to the author, who is an active musician and has studied composition.

The work described in this thesis proposal does not implement OMR solutions. On the other hand, considerable contributions have been made to OMR as a field, by providing new and much needed benchmarking infrastructure: evaluation procedures and, even more importantly, data. The ground truth design process led to establishing a theoretical result in OMR objectives: the sufficiency and necessity of the notation graph for reprintability and replayability. While the idea of a notation graph is not new, to the best of our knowledge, ours is the first body of work to implement it in full: define the graph thoroughly for the wide range of music notation symbols and symbol configurations, consistently with the information content of music notation that makes it straightforward to infer musical content conveyed by the notation graph, and provide a substantial dataset.[52]

These results should now enable developing differentiable objective functions that enable solving OMR globally with respect to the notation graph, such as those described in sec. 5 of this proposal, leveraging the syntactic properties of music notation.

References

Ana Rebelo. 2012. *Robust Optical Recognition of Handwritten Musical Scores based on Domain Knowledge*. Ph.D. thesis. http://www.inescporto.pt/arebelo/arebeloThesis.pdf.

Ana Rebelo, Andre Marcal, and Jaime S. Cardoso. 2013. Global constraints for syntactic consistency in OMR: an ongoing approach. In *Proceedings of the International Conference on Image Analysis and Recognition (ICIAR)*.

Ana Rebelo, 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. *Int J Multimed Info Retr* 1(3):173–190. https://doi.org/10.1007/s13735-012-0004-6.

Andrew Hankinson. 2015. *Optical music recognition infrastructure for large-scale music document analysis*. Ph.D. thesis.

Andrew Hankinson, John Ashley Burgoyne, Gabriel Vigliensoni, Alastair Porter, Jessica Thompson, Wendy Liu, Remi Chiu, and Ichiro Fujinaga. 2012. Digital Document Image Retrieval Using Optical Music Recognition. In Fabien Gouyon, Perfecto Herrera, Luis Gustavo Martins, and Meinard Müller, editors, *Proceedings of the 13th International Society for Music Information Retrieval Conference, ISMIR 2012, Mosteiro S.Bento Da Vitória, Porto, Portugal, October 8-12, 2012*. FEUP Edições, pages 577–582. http://ismir2012.ismir.net/event/papers/577-ismir-2012.pdf.

Arnau Baro, Pau Riba, and Alicia Fornés. 2016. Towards the Recognition of Compound Music Notes in Handwritten Music Scores. In *15th International Conference on Frontiers in Handwriting Recognition, ICFHR 2016, Shenzhen, China, October 23-26, 2016*. IEEE Computer Society, pages 465–470. https://doi.org/10.1109/ICFHR.2016.0092.

D. Bainbridge and T. Bell. 1997a. An extensible Optical Music Recognition system. In *Proceedings of the nineteenth Australasian computer science conference*. page 308317.

D. Bainbridge and T. Bell. 2001. The challenge of optical music recognition. *Computers and the Humanities* 35:95–121. https://doi.org/10.1023/A:1002485918032.

David Bainbridge and Tim Bell. 1997b. Dealing with superimposed objects in optical music recognition. *Proceedings of 6th International Conference on Image Processing and its Applications* (443):756–760. https://doi.org/10.1049/cp:19970997.

David Bainbridge and Tim Bell. 2003. A music notation construction engine for optical music recognition. *Software - Practice and Experience* 33(2):173–200. https://doi.org/10.1002/spe.502.

Baoguang Shi, Xiang Bai, and Cong Yao. 2015. An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition. *CoRR* abs/1507.05717. http://arxiv.org/abs/1507.05717.

P. Bellini, I. Bruno, and P. Nesi. 2001. Optical music sheet segmentation. In *Proceedings First International Conference on WEB Delivering of Music. WEDELMUSIC 2001*. Institute of Electrical

[52] This work has also been received well by the international OMR community, leading to ongoing collaboration with OMR researchers in Alicante and Barcelona, and at JKU Linz.

& Electronics Engineers (IEEE), pages 183–190. https://doi.org/10.1109/wdm.2001.990175.

Emmanouil Benetos, Anssi Klapuri, and Simon Dixon. 2012. Score-informed transcription for automatic piano tutoring. In *Proceedings of the 20th European Signal Processing Conference, EUSIPCO 2012, Bucharest, Romania, August 27-31, 2012*. IEEE, pages 2153–2157.

Ondřej Bojar, Miloš Ercegovčević, Martin Popel, and Omar F. Zaidan. 2011. A Grain of Salt for the WMT Manual Evaluation. *Proceedings of the Sixth Workshop on Statistical Machine Translation* pages 1–11. http://dl.acm.org/citation.cfm?id=2132960.2132962.

Donald Byrd. 1984. *Music Notation by Computer*. Ph.D. thesis.

Chris Callison Burch, Cameron Fordyce, Philipp Koehn, Christof Monz, and Josh Schroeder. 2007. (Meta-) Evaluation of Machine Translation. *Proceedings of the Second Workshop on Statistical Machine Translation* pages 136–158. http://dl.acm.org/citation.cfm?id=1626355.1626373.

Chris Callison Burch, Philipp Koehn, Christof Monz, Kay Peterson, Mark Przybocki, and Omar F. Zaidan. 2010. Findings of the 2010 Joint Workshop on Statistical Machine Translation and Metrics for Machine Translation. *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR* pages 17–53. http://dl.acm.org/citation.cfm?id=1868850.1868853.

Jorge Calvo Zaragoza, Gabriel Vigliensoni, and Ichiro Fujinaga. 2016. Document Analysis for Music Scores via Machine Learning. In *Proceedings of the 3rd International Workshop on Digital Libraries for Musicology*. ACM, New York, NY, USA, DLfM 2016, pages 37–40. https://doi.org/10.1145/2970044.2970047.

Jorge Calvo Zaragoza, Gabriel Vigliensoni, and Ichiro Fujinaga. 2017. A machine learning framework for the categorization of elements in images of musical documents. In *Third International Conference on Technologies for Music Notation and Representation*. University of A Coruña, A Coruña. http://grfia.dlsi.ua.es/repositori/grfia/pubs/360/tenor-unified-categorization.pdf.

Sukalpa Chanda, Debleena Das, Umapada Pal, and Fumitaka Kimura. 2014. Offline Hand-Written Musical Symbol Recognition. *2014 14th International Conference on Frontiers in Handwriting Recognition* pages 405–410. https://doi.org/10.1109/ICFHR.2014.74.

Christoph Dalitz, Michael Droettboom, Bastian Pranzas, and Ichiro Fujinaga. 2008. A Comparative Study of Staff Removal Algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30(5):753–766. https://doi.org/10.1109/tpami.2007.70749.

Christopher Raphael and Jingya Wang. 2011. New Approaches to Optical Music Recognition. In Anssi Klapuri and Colby Leider, editors, *Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR 2011, Miami, Florida, USA, October 24-28, 2011*. University of Miami, pages 305–310. http://ismir2011.ismir.net/papers/OS3-3.pdf.

Arshia Cont, Diemo Schwarz, Norbert Schnell, and Christopher Raphael. 2007. Evaluation of Real-Time Audio-to-Score Alignment. In *International Symposium on Music Information Retrieval (ISMIR)*. Vienna, Austria. https://hal.inria.fr/hal-00839068.

Bertrand Coüasnon and Jean Camillerapp. 1994. Using Grammars To Segment and Recognize Music Scores. *Pattern Recognition* pages 15–27.

Cuihong Wen, Ana Rebelo, Jing Zhang, and Jaime Cardoso. 2015. A new optical music recognition system based on combined neural network. *Pattern Recognition Letters* 58:1 – 7. https://doi.org/10.1016/j.patrec.2015.02.002.

Cuihong Wen, Jing Zhang, Ana Rebelo, and Fanyong Cheng. 2016. A Directed Acyclic Graph-Large Margin Distribution Machine Model for Music Symbol Classification. *PLOS ONE* 11(3):e0149688. https://doi.org/10.1371/journal.pone.0149688.

Dan Ringwalt, Roger B. Dannenberg, and Andrew Russell. 2015. Optical music recognition for interactive score display. In Edgar Berdahl and Jesse T. Allison, editors, *15th International Conference on New Interfaces for Musical Expression, NIME 2015, Baton Rouge, Louisiana, USA, May 31 - June 3, 2015.*. nime.org, pages 95–98.

David S Prerau. 1971. Computer pattern recognition of printed music. *AFIP Conference proceedings, Vol.39 1971 fall joint computer conference* pages 153–162.

Denis Pruslin. 1966. *Automatic recognition of sheet music*. Ph.D. thesis, Cambridge, Massachusetts, USA.

Donald Byrd 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):169–195. https://doi.org/10.1080/09298215.2015.1045424.

J. dos Santos Cardoso, A. Capela, A. Rebelo, C. Guedes, and J. Pinto da Costa. 2009. Staff Detection with Stable Paths. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31(6):1134–1139. https://doi.org/10.1109/tpami.2009.34.

Sebastian Ewert, Bryan Pardo, Meinard Müller, and Mark D. Plumbley. 2014. Score-informed source separation for musical audio recordings: An overview. *IEEE Signal Process. Mag.* 31(3):116–124. https://doi.org/10.1109/MSP.2013.2296076.

Sebastian Ewert, Siying Wang, Meinard Müller, and Mark B. Sandler. 2016. Score-informed identification of missing and extra notes in piano recordings. In Michael I. Mandel, Johanna Devaney, Douglas Turnbull, and George Tzanetakis, editors, *Proceedings of the 17th International Society for Music Information Retrieval Conference, ISMIR 2016, New York City, United States, August 7-11, 2016*. pages 30–36. https://wp.nyu.edu/ismir2016/wp-content/uploads/sites/2294/2016/07/123_Paper.pdf.

Alicia Fornés. 2005. *Analysis of Old Handwritten Musical Scores*. Ph.D. thesis.

Alicia Fornes, Anjan Dutta, Albert Gordo, and Josep Llados. 2011. The ICDAR 2011 music scores competition: Staff removal and writer identification. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*. IEEE, pages 1511–1515.

Alicia Fornés, Anjan Dutta, Albert Gordo, and Josep Llads. 2012. CVC-MUSCIMA: a ground truth of handwritten music score images for writer identification and staff removal. *International Journal on Document Analysis and Recognition (IJDAR)* 15(3):243–251. https://doi.org/10.1007/s10032-011-0168-2.

C Fremerey, D Damm, F Kurth, and M Clausen. 2009. Handling Scanned Sheet Music and Audio Recordings in Digital Music Libraries pages 1–2.

Christian Fremerey, Meinard Müller, Frank Kurth, and Michael Clausen. 2008. Automatic mapping of scanned sheet music to audio recordings. *Proceedings of the International Conference on Music Information Retrieval* pages 413–8.

Ichiro Fujinaga. 1996. *Adaptive Optical Music Recognition*. Ph.D. thesis.

Ichlro Fujinaga. 1988. *Optical Music Recognition using Projections*. Master's thesis.

Graham Jones, Bee Ong, Ivan Bruno, and Kia Ng. 2008. Optical Music Imaging: Music Document Digitisation, Recognition, Evaluation, and Restoration. *Interactive Multimedia Music Technologies* pages 50–79. https://doi.org/10.4018/978-1-59904-150-6.ch004.

Jan Hajič jr. and Pavel Pecina. 2017. In Search of a Dataset for Handwritten Optical Music Recognition: Introducing MUSCIMA++. *ArXiv e-prints* .

Jan Hajič jr., Jiří Novotný, Pavel Pecina, and Jaroslav Pokorný. 2016. Further Steps towards a Standard Testbed for Optical Music Recognition. In Michael Mandel, Johanna Devaney, Douglas Turnbull, and George Tzanetakis, editors, *Proceedings of the 17th International Society for Music Information Retrieval Conference*. New York University, New York University, New York, USA, pages 157–163. https://18798-presscdn-pagely.netdna-ssl.com/ismir2016/wp-content/uploads/sites/2294/2016/07/289_Paper.pdf.

JaeMyeong Yoo, Nguyen Dinh Toan, DeokJai Choi, HyukRo Park, and Gueesang Lee. 2008. Advanced Binarization Method for Music Score Recognition Using Local Thresholds. *2008 IEEE 8th International Conference on Computer and Information Technology Workshops* pages 417–420. https://doi.org/10.1109/CIT.2008.Workshops.101.

Jiří Novotný and Jaroslav Pokorný. 2015. Introduction to Optical Music Recognition: Overview and Practical Challenges. *DATESO 2015 Proceedings of the 15th annual international workshop* .

John Ashley, Burgoyne Laurent, Pugin Greg, and Eustace Ichiro Fujinaga. 2008. A Comparative Survey of Image Binarisation Algorithms for Optical Recognition on Degraded Musical Sources. *ACM SIGSOFT Software Engineering Notes* 24(1985):1994–1997.

Jorge Calvo-Zaragoza, David Rizo, and José Manuel Iñesta Quereda. 2016. Two (Note) Heads Are Better Than One: Pen-Based Multimodal Interaction with Music Scores. In Michael I. Mandel, Johanna Devaney, Douglas Turnbull, and George Tzanetakis, editors, *Proceedings of the 17th International Society for Music Information Retrieval Conference, ISMIR 2016, New York City, United States, August 7-11, 2016*. pages 509–514. https://wp.nyu.edu/ismir2016/wp-content/uploads/sites/2294/2016/07/006_Paper.pdf.

Jorge Calvo-Zaragoza and Jose Oncina. 2014. Recognition of Pen-Based Music Notation: The HOMUS Dataset. *22nd International Conference on Pattern Recognition* https://doi.org/10.1109/icpr.2014.524.

Ian Knopke and Donald Byrd. 2007. Towards Musicdiff : A Foundation for Improved Optical Music Recognition Using Multiple Recognizers. *International Society for Music Information Retrieval Conference* .

Laurent Pugin. 2006. Optical Music Recognitoin of Early Typographic Prints using Hidden Markov Models. In *ISMIR 2006, 7th International Conference on Music Information Retrieval, Victoria, Canada, 8-12 October 2006, Proceedings*. pages 53–56.

Alon Lavie and Abhaya Agarwal. 2007. Meteor: An Automatic Metric for MT Evaluation with High Levels of Correlation with Human Judgments. *Proceedings of the Second Workshop on Statistical Machine Translation* pages 228–231. http://dl.acm.org/citation.cfm?id=1626355.1626389.

Liang Chen, Erik Stolterman, and Christopher Raphael. 2016. Human-Interactive Optical Music Recognition. In Michael I. Mandel, Johanna Devaney, Douglas Turnbull, and George Tzanetakis, editors, *Proceedings of the 17th International Society for Music Information Retrieval Conference, ISMIR 2016, New York City, United States, August 7-11, 2016*. pages 647–653. https://wp.nyu.edu/ismir2016/wp-content/uploads/sites/2294/2016/07/106_Paper.pdf.

Liang Chen, Rong Jin, and Christopher Raphael. 2015. Renotation from Optical Music Recognition. In *Mathematics and Computation in Music*, Springer Science + Business Media, pages 16–26. https://doi.org/10.1007/978-3-319-20603-5_2.

Nailja Luth. 2002. Automatic Identification of Music Notations. In *Proceedings of the Second International Conference on WEB Delivering of Music (WDELMUSIC'02)*.

Matouš Macháček and Ondřej Bojar. 2014. Results of the WMT14 Metrics Shared Task. *Proceedings of the Ninth Workshop on Statistical Machine Translation* pages 293–301.

Matouš Macháček and Ondřej Bojar. 2015. Evaluating Machine Translation Quality Using Short Segments Annotations. *The Prague Bulletin of Mathematical Linguistics* 103(1). https://doi.org/10.1515/pralin-2015-0005.

Matthias Dorfer, Andreas Arzt, and Gerhard Widmer. 2016a. Towards End-to-End Audio-Sheet-Music Retrieval. *CoRR* abs/1612.05070. http://arxiv.org/abs/1612.05070.

Matthias Dorfer, Andreas Arzt, and Gerhard Widmer. 2016b. Towards Score Following In Sheet Music Images. In Michael I. Mandel, Johanna Devaney, Douglas Turnbull, and George Tzanetakis, editors, *Proceedings of the 17th International Society for Music Information Retrieval Conference, ISMIR 2016, New York City, United States, August 7-11, 2016*. pages 789–795. https://wp.nyu.edu/ismir2016/wp-content/uploads/sites/2294/2016/07/027_Paper.pdf.

Maura Church and Michael Scott Cuthbert. 2014. Improving Rhythmic Transcriptions via Probability Models Applied Post-OMR. In Hsin-Min Wang, Yi-Hsuan Yang, and Jin Ha Lee, editors, *Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR 2014, Taipei, Taiwan, October 27-31, 2014*. pages 643–648.

Michael Droettboom and Ichiro Fujinaga. 2004. Symbol-level groundtruthing environment for OMR. *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR 2004)* pages 497–500.

Hidetoshi Miyao and Robert M Haralick. 2000. Format of Ground Truth Data Used in the Evaluation of the Results of an Optical Music Recognition System. In *IAPR workshop on document analysis systems*. page 497506.

Kia Ng, David Cooper, Ewan Stefani, Roger Boyle, and Nick Bailey. 1999. Embracing the Composer: Optical Recognition of Handwritten Manuscripts.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics* pages 311–318. https://doi.org/10.3115/1073083.1073135.

Pau Riba, Alicia Fornés, and Josep Lladós. 2015. Towards the Alignment of Handwritten Music Scores. In Bart Lamiroy and Rafael Dueire Lins, editors, *Graphic Recognition. Current Trends and Challenges - 11th International Workshop, GREC 2015, Nancy, France, August 22-23, 2015, Revised Selected Papers*. Springer, volume 9657 of *Lecture Notes in Computer Science*, pages 103–116. https://doi.org/10.1007/978-3-319-52159-6_8.

Van Khien Pham and Gueesang Lee. 2015. Music Score Recognition Based on a Collaborative Model. *International Journal of Multimedia and Ubiquitous Engineering* 10(8):379–390.

Pierfrancesco Bellini, Ivan Bruno, and Paolo Nesi. 2007. Assessing Optical Music Recognition Tools. *Computer Music Journal* 31(1):68–93. https://doi.org/10.1162/comj.2007.31.1.68.

Telmo Pinto, Ana Rebelo, Gilson Giraldi, and Jaime S Cardoso. 2010. Content Aware Music Score Binarization. Technical report.

Telmo Pinto, Ana Rebelo, Gilson Giraldi, and Jaime S Cardoso. 2011. Music Score Binarization Based on Domain Knowledge. In *Iberian Conference on Pattern Recognition and Image Analysis*. Springer Verlag, Berlin Heidelberg, volume 6669, pages 700–708.

Ana Rebelo, G. Capela, and Jaime S. Cardoso. 2010. Optical recognition of music symbols. *International Journal on Document Analysis and Recognition* 13:19–31. https://doi.org/10.1007/s10032-009-0100-1.

Ana Rebelo and Jaime S. Cardoso. 2013. Staff line detection and removal in the grayscale domain. *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR* pages 57–61. https://doi.org/10.1109/ICDAR.2013.20.

Ana Rebelo, Filipe Paszkiewicz, Carlos Guedes, Andre R S Marcal, and Jaime S Cardoso. 2011a. A Method for Music Symbols Extraction based on Musical Rules. *Proceedings of BRIDGES* (1):81–88.

Ana Rebelo, Jakub Tkaczuk, Ricardo Sousa, and Jaime S. Cardoso. 2011b. Metric learning for music symbol recognition. *Proceedings - 10th International Conference on Machine Learning and Applications, ICMLA 2011* 2:106–111. https://doi.org/10.1109/ICMLA.2011.94.

K. T. Reed and J. R. Parker. 1996. Automatic computer recognition of printed music. *Proceedings - International Conference on Pattern Recognition* 3:803–807. https://doi.org/10.1109/ICPR.1996.547279.

JW W Roach and J E Tatem. 1988. Using domain knowledge in low-level visual processing to interpret handwritten music: an experiment. *Pattern Recognition* 21(1):33–44. https://doi.org/10.1016/0031-3203(88)90069-6.

Rong Jin and Christopher Raphael. 2012. Interpreting Rhythm in Optical Music Recognition. In Fabien Gouyon, Perfecto Herrera, Luis Gustavo Martins, and Meinard Müller, editors, *Proceedings of the 13th International Society for Music Information Retrieval Conference, ISMIR 2012, Mosteiro S.Bento Da Vitória, Porto, Portugal, October 8-12, 2012*. FEUP Edições, pages 151–156. http://ismir2012.ismir.net/event/papers/151-ismir-2012.pdf.

Florence Rossant and Isabelle Bloch. 2007. Robust and Adaptive OMR System Including Fuzzy Modeling, Fusion of Musical Rules, and Possible Error Detection. *EURASIP Journal on Advances in Signal Processing* 2007(1):081541. https://doi.org/10.1155/2007/81541.

Rui Miguel Filipe da Silva. 2013. *Mobile framework for recognition of musical characters*. Master's thesis. https://repositorio-aberto.up.pt/bitstream/10216/68500/2/26777.pdf.

Geoffrey Sampson. 1985. *Writing Systems: A Linguistic Introduction*. Stanford University Press. https://books.google.cz/books?id=tVcdNRvwoDkC.

Craig Sapp. 2013. OMR Comparison of SmartScore and SharpEye. `https://ccrma.stanford.edu/{~}craig/mro-compare-beethoven` https://ccrma.stanford.edu/ craig/mro-compare-beethoven.

Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. 2015. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *CoRR* abs/1506.01497. http://arxiv.org/abs/1506.01497.

Scott Sheridan and Susan E George. 2004. Defacing Music Scores for Improved Recognition. *Proceedings of the Second Australian Undergraduate Students Computing Conference* (i):1–7.

M.V. V Stuckelberg and David Doermann. 1999. On musical score recognition using probabilistic reasoning. *Proceedings of the Fifth International Conference on Document Analysis and Recognition. ICDAR '99 (Cat. No.PR00318)* (Did):115–118. https://doi.org/10.1109/ICDAR.1999.791738.

Mariusz Szwoch. 2007. Guido: A musical score recognition system. *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR* 2(3):809–813. https://doi.org/10.1109/ICDAR.2007.4377027.

Mariusz Szwoch. 2008. Using MusicXML to Evaluate Accuracy of OMR Systems. *Proceedings of the 5th International Conference on Diagrammatic Representation and Inference* pages 419–422. https://doi.org/10.1007/978-3-540-87730-1_53.

Véronique Sébastien, Henri Ralambondrainy, Olivier Sébastien, and Noël Conruyt. 2012. Score Analyzer: Automatically Determining Scores Difficulty Level for Instrumental e-Learning. In Fabien Gouyon, Perfecto Herrera, Luis Gustavo Martins, and Meinard Müller, editors, *Proceedings of the 13th International Society for Music Information Retrieval Conference, ISMIR 2012, Mosteiro S.Bento Da Vitória, Porto, Portugal, October 8-12, 2012*. FEUP Edições, pages 571–576. http://ismir2012.ismir.net/event/papers/571-ismir-2012.pdf.

Victor Padilla, Alan Marsden, Alex McLean, and Kia Ng. 2014. Improving OMR for Digital Music Libraries with Multiple Recognisers and Multiple Sources. *Proceedings of the 1st International Workshop on Digital Libraries for Musicology - DLfM '14* pages 1–8. https://doi.org/10.1145/2660168.2660175.