The MUSCIMA++ Dataset for Handwritten Optical Music Recognition

Jan Hajič jr.
Institute of Formal and Applied Linguistics
Charles University
Email: hajicj@ufal.mff.cuni.cz

Pavel Pecina
Institute of Formal and Applied Linguistics
Charles University
Email: pecina@ufal.mff.cuni.cz

***Abstract*—Optical Music Recognition (OMR) promises to make accessible the content of large amounts of musical documents, an important component of cultural heritage. However, the field does not have an adequate dataset and ground truth for benchmarking OMR systems, which has been a major obstacle to measurable progress. Furthermore, machine learning methods for OMR require training data. We design and collect MUSCIMA++, a new dataset for OMR. Ground truth in MUSCIMA++ is a *notation graph*, which our analysis shows to be a necessary and sufficient representation of music notation. Building on the CVC-MUSCIMA dataset for staffline removal, the MUSCIMA++ dataset v1.0 consists of 140 pages of handwritten music, with 91255 manually annotated notation symbols and 82261 explicitly marked relationships between symbol pairs. The dataset allows training and directly evaluating models for symbol classification, symbol localization, and notation graph assembly, and musical content extraction, both in isolation and jointly. Open-source tools are provided for manipulating the dataset, visualizing the data and annotating more, and the data is made available under an open license.**

I. Introduction: what dataset?

Optical Music Recognition (OMR) is a field of document analysis that aims to automatically read music. Music notation encodes musical information in a graphical form; OMR backtracks through this process to extract the musical information from its graphical representation. OMR can be likened to OCR for the music notation writing system; however, it is more difficult [1], and remains an open problem [2], [3]. The intricacies of Common western music notation (CWMN[1]) have been thoroughly discussed since early attempts at OMR, notably by Byrd [4], [5].

One of the most persistent hindrances to OMR progress is a **lack of datasets**. These are necessary to provide ground truth for evaluating OMR systems [1], [5]–[8], to enable fair, replicable comparison among academic and commercial systems. Furthermore, especially for handwritten notation, supervised machine learning methods have often been used that require training data [9]–[12].

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 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 [2], [5]–[7], [13].

For printed music notation, the lack of datasets can be bypassed by generating music in representations such as LilyPond[2] or MEI,[3] and capturing intermediate steps of the rendering process. However, for handwritten music, no satisfactory synthetic data generator exists so far, and an extensive annotation effort cannot be avoided. Therefore, to best utilize our resources available for creating a dataset, we create a dataset of **handwritten notation**.

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

- What should the ground truth y_i be for an image x_i?
- What sheet music do we choose as data points?

The definition of ground truth must **reflect what OMR *does*.** Miyao and Haralick [14] 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* is 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. 1).

The selection of musical score images in the dataset should **cover the known "dimensions of difficulty"** [5], to allow for assessing OMR systems with respect to increasingly complex inputs.

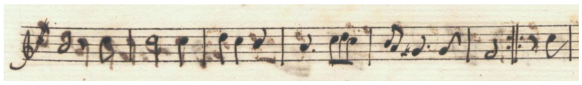
In the rest of the article, we reason what the ground truth for OMR should be (II-A) and what kinds of musical score images the dataset should contain (II-B); we scavenge existing OMR datasets for work already done that would satisfy these design choices (III); finally, we describe the MUSCIMA++ dataset (IV), establish simple baselines (V); and provide some concluding remarks (VI).

The main contributions of this work are:

[1]We assume the reader is familiar with CWMN. In case a refresher is needed, we recommend chapter 2 of "Music Notation by Computer" [4], by Donald Byrd. A comprehensive list of music notation terminology is maintained on Wikipedia: https://en.wikipedia.org/wiki/List_of_musical_symbols

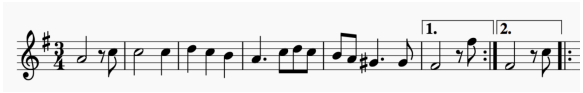
[2]http://www.lilypond.org

[3]http://www.music-encoding.org

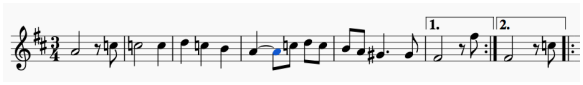


(a) Input: manuscript image.

(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

Fig. 1: 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).

- MUSCIMA++[4] – an extensive dataset of handwritten musical symbols and their relationships,[5]
- A *notation graph* ground truth definition and implementation that de-couples the graphical expression of music and musical semantics, while recording sufficient information to bridge this gap, and also helps understanding the problem space of OMR;
- Open-source tools for processing the data including inferring pitches and durations, visualizing it, and annotating more.

MUSCIMA++ enables training and evaluating models for symbol localization, classification, and arguably its most innovative aspect for OMR is that it enables directly solving music notation reconstruction, in a way that explicitly considers the need to infer musical semantics.

II. Designing a dataset for OMR

In this section, we discuss the key design concerns introduced above: an appropriate ground truth for OMR, and the choice of data.

A. Ground Truth

The ground truth over a dataset is the desired output of a system solving a task. Therefore, in order to design the ground truth for the dataset, we need to understand how OMR can be expressed in terms of inputs and outputs. OMR solutions are usually pipelines with four major stages [1], [2]:

1) Image preprocessing: enhancement, binarization, scaling;

[4]Standing for MUsic SCore IMAges, credit for abbreviation to [15]

[5]Available from: http://hdl.handle.net/11372/LRT-2372

TABLE I: 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, ...

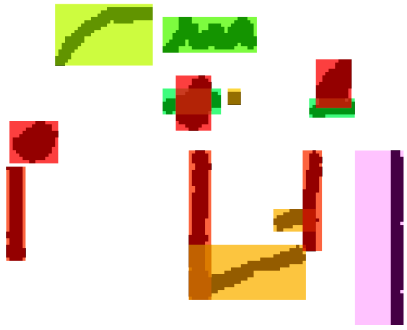
2) Music symbol recognition: staffline identification and removal, localization and classification of other symbols;
3) Musical notation reconstruction: recovering the logical structure of the score;
4) Final representation construction: depending on the output requirements, usually inferring pitch and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

The key problems of OMR reside in stages 2 and 3: finding individual musical symbols on the page, and recovering their relationships. The inputs and outputs of the individual pipeline stages and sub-tasks is summarized in Table I. While end-to-end OMR that bypasses some sections of this pipeline is an attractive option (see [16]), these should still be compared against more orthodox solutions.

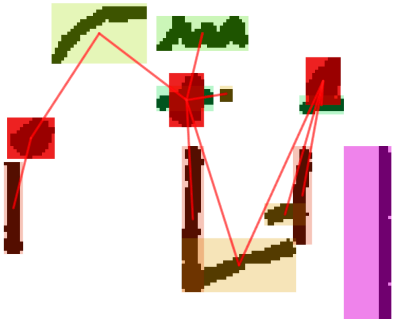
The input of **music symbol recognition** is a "cleaned" and usually binarized image. The output of this stage is a list of musical symbols recording their locations on the page, and their types (e.g., c-clef, beam, sharp). Usually, there are three sub-tasks: staffline identification and removal, symbol localization (in binary images, synonymous with foreground segmentation), and symbol classification [2]. Stafflines are typically handled as a separate step [17], due to them being rather a layout element than a character-like symbol.

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 enable 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. Graphs have been explicitly used for assembly of music notation primitives e.g. by [18], [19], and grammar-based approaches (e.g., [20]–[23]) lend themselves to a graph representation as well, by recording the parse tree(s). An example of symbol recognition and notation reconstruction



(a) Notation symbols, color-coded: 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 with edges, highlighting noteheads as "roots" of subtrees. Noteheads share the beam and slur symbols.

Fig. 2: Visualizing the list of symbols and the notation graph over staff removal output. 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).

output over the same snippet of a musical score is given in Figure 2.

A key observation for ground truth design is that **the notation graph records information both necessary and sufficient for both replayability and reprintability,** and thus makes a good ground truth for an OMR dataset.

1) 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. No finite alphabet can be designed so that its symbols could be interpreted in isolation: recognizing a note is not enough to determine its pitch: one needs it to relate to the stafflines, clefs, key signatures, etc.

2) Sufficient: The process of reading musical scores 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 in order to project the written page to the corresponding point in the space of musical note[6] 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 creating the desired representation in stage 4 is only a technical task: implementing conversion to the desired output format (which can nevertheless still be a very complex piece of software).[7] This observation also implies that an OMR system that can recover the notation graph does *not* have have to explicitly recover pitch and duration.

[6]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.

[7]The representation used to record the dataset is not necessarily best for experiments – but experiment-specific output representations (such as a MIDI file for replayability-only experiments) are *unambiguously obtainable* from the notation graph.

B. Choice of data

The dataset should enable evaluating handwritten OMR with respect to the "challenge space" of OMR. In their state-of-the-art analysis of the difficulties of OMR, Byrd and Simonsen [5] identify three axes along which musical score images become less or more challenging inputs for an OMR system: *Notation complexity*, *Image quality*, and *Tightness of spacing*.

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 [5] into four levels. 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 [24] are harder to use. Level 3, multi-staff single-voice music, tests high-level segmentation into systems and staffs. Level 4, pianoform music, then presents the most complex, combined challenge, as piano music has exploited the rules of CWMN to their fullest [5] and sometimes beyond. 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 [5] essentially define how to simulate them; many morphological distortions have already been implemented for staff removal data [15], [25].

The **tightness of spacing** in [5] refers to default horizontal and vertical distances between symbols.[8] 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 [5], Fig. 21). 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. – see Fig. 3. The various styles of handwriting should be represented in the dataset as broadly as possible.

III. Existing datasets

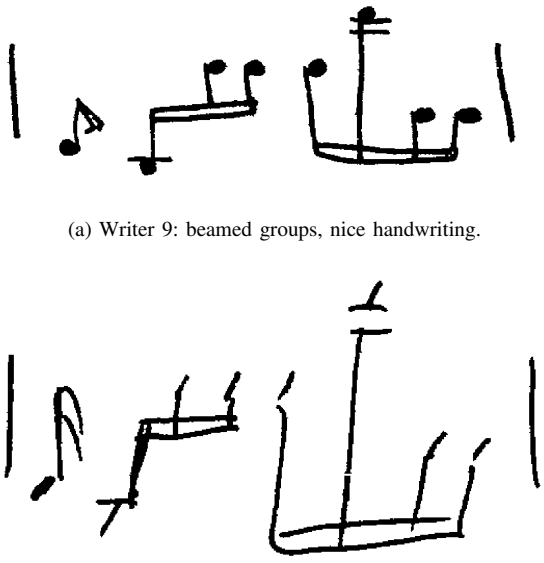
We describe the available datasets and discuss how they correspond to the requirements of Section II. Reviewing Table I, the subtasks at stages 2 and 3 of the OMR pipeline are (a) staffline removal, (b) symbol localization, (c) symbol classification, and (d) symbol assembly.

For **staff removal** in handwritten music, the premier dataset is CVC-MUSCIMA [15], 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 [17]. The dataset fulfills the requirements for a good choice of data: the 20 pages include scores of all 4 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. 3. Importantly, CVC-MUSCIMA is freely available for download under a CC-BY-NC-SA 4.0 license.[9]

The most extensive dataset for handwritten **symbol classification** is the HOMUS dataset of Calvo-Zaragoza and Oncina [11], 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 data is recorded from a touchscreen device, so it can be used for online as well as offline recognition. However, the dataset 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. For **symbol localization**, we are only aware of a dataset of 3222 handwritten symbols by Rebelo et al. [26], and for **notation reconstruction**, we are not aware of a dataset that provides ground truth for recovering the relationships among handwritten musical symbols.

IV. The MUSCIMA++ dataset

Our main source of musical score images is the CVC-MUSCIMA dataset, described in subsection III. The annotator team consisted of three professional and four advanced amateur musicians. Each annotator marked one of the 50 versions for each of the 20 CVC-MUSCIMA pages. We selected 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, thus fulfilling the same choice-of-data requirements (notation complexity, handwriting style) as CVC-MUSCIMA itself.



(a) Writer 9: beamed groups, nice handwriting.

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

Fig. 3: Variety of handwriting styles in CVC-MUSCIMA.

There is a total of 91255 symbols (excluding staff objects, which are already given in the CVC-MUSCIMA ground truth) 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. (Given the decomposition of notes into primitives, the equivalent number in terms of HOMUS symbols would be $\tilde{5}7\ 000$.) The choice of symbols and relationship policies is described in subsec. IV-A. The frequencies of the most significant symbols are described in Table II.

A. MUSCIMA++ ground truth

Our ground truth is a **graph of musical symbols and their relationships,** with unlabeled directed edges. [10] For each vertex (symbol), we annotated:

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

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

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

[10] The complete annotation guidelines detailing what the symbol set is and how to deal with individual notations are available online: https://muscimarker.readthedocs.io/en/latest/instructions.html

TABLE II: 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*

These are a superset of the primitive attributes in [14]. Annotating the mask enables us to build an accurate model of actual symbol shapes.

We do not define a `note` symbol. The concept of a note on paper [6], [11], [26] is ambiguous: 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. Furthermore, it is not clear what primitives constitute a note. If we follow musical semantics, should e.g. an accidental be considered a part of the note, because it directly influences its pitch? It is more elegant to annotate *how the "note" musical objects are expressed*, and if need be, use the relationships among the primitives to construct the somewhat arbitrary "note" written symbols when necessary.

Instead of trying to categorize symbols as low- or high-level [5], [6], [13] according to whether they carry semantics or not (which is a dubious proposition: musical semantics arise from *configurations* of symbols, as music notation is mostly a featural writing system, where the individual symbols encode separate well-defined aspects of musical semantics but make very limited sense in isolation), we express the dichotomy through the rules for forming relationships. This leads to "layered" annotation. E.g., 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 both numerals. An example of this structure for is given in Figure 4. We take care to define relationships so that the result is a Directed Acyclic Graph (DAG). There is no theoretical limit on the maximum oriented path length, but in practice, it is rarely longer than 3. We break down symbols that consist of multiple connected components when these components can be used in syntactically valid music notation in different configurations to encode distinct musical semantics: an empty notehead may show up with a stem, without one, with multiple stems when two voices share pitch,[11] or it may share stem with others, so we define these as separate symbols. An f-clef dot should not exist without the rest of the clef, and vice versa, so we define the `f-clef` as a single symbol; however, a single repeat may have a variable number of repeat dots, based on how many staves it is spanning, so we define a `repeat-dot` separately.

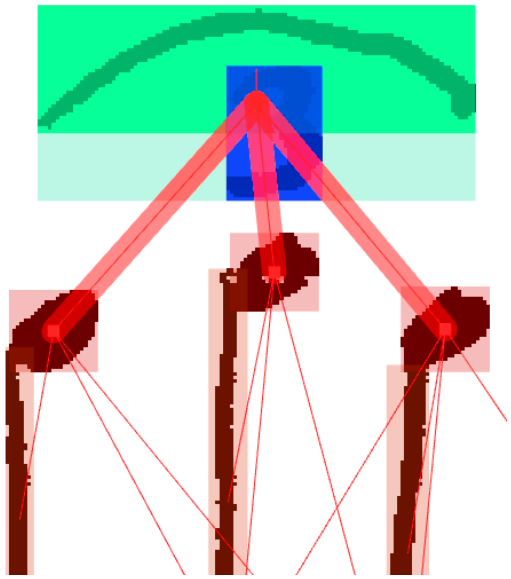


Fig. 4: Two-layer annotation of a triplet. The symbols `numeral_3` (in blue), `tuple_bracket/line`, and the three noteheads that form the triplet are highlighted. The `tuple` symbol itself, to which the noteheads are connected, is the lighter rectangle encompassing its two components; it has relationships leading to both of them (not highlighted).

B. MUSCIMA++ software tools

In order to make using the dataset easier, we provide two open-source software tools. The `musicma` Python 3 package[12] 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), and implements extracting pitch, duration and onset data from the notation graph, thus enabling exporting MIDI and thus multimodal OMR experiments, even if so far only on synthesized audio. Second, we provide the `MUSCIMarker` application[13] used for creating the dataset, which can also visualize the data.

C. Annotation process and quality control

The annotators worked on symbols-only CVC-MUSCIMA images, which allowed for more efficient annotation. The

[11] As seen in page 20 of CVC-MUSCIMA.
[12] https://github.com/hajicj/muscima
[13] https://github.com/hajicj/MUSCIMarker

interface used to add symbols consists of two tools: foreground lasso selection, and connected component selection, and our MUSCIMarker software also supports editing the objects' masks in-place.

After an annotator completed an image, we checked for correctness. Automated validation of the submitted relationships was implemented in MUSCIMarker, however, manual checks and manually correcting mistakes found in auto-validation was still needed, as the validation was just an advisory voice to highlight questionably annotated symbols. 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 7 % of the foreground pixels in its bounding box were not marked as part of any symbol at all). We also fixed other clearly wrong markings (e.g., if a significant amount of stem-only pixels was marked as part of a beam).

The average speed overall was 4.3 symbols per minute, or one per 14 seconds: an average page of about 650 symbols took about $2\frac{3}{4}$ hours. Annotating the dataset using the process detailed above took roughly 400 hours of work; the "quality control" correctness checks and managing the annotation process took an additional 150. The second, more complete round of quality control took roughly 80 hours.

D. Inter-annotator agreement

In order to assess the trustworthiness of the annotations, all annotators were given the same image to annotate, and we measured inter-annotator agreement both before and after quality control (QC) was applied, and we also measured how many changes were made in QC. Given that the expected level of true ambiguity in our ground truth is relatively low, we can interpret disagreement between annotators as evidence of inaccuracies. At the same time, a comparison of annotations *after* quality control gives the upper limit on achievable per-pixel accuracy.

1) Computing agreement: To compute agreement, we align the annotated object sets against each other, and compute the macro-averaged per-pixel f-score over the aligned object pairs. 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 have no such a 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 there are multiple such "best friend" candidates, we prefer aligning those that have the same symbol class. Objects that have no counterpart contribute 0 to both precision and recall.

2) Agreement results: The resulting f-scores are summarized in Table III. We measured inter-annotator agreement both before quality control (noQC-noQC) and after (withQC-withQC), and we also measured the extent to which quality control changed the originally submitted annotations (noQC-withQC), averaged over the 7 annotators. Ideally, 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.

TABLE III: 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**

Legitimate sources of disagreement lie in unclear symbol boundaries in intersections, and illegible handwriting. However, even 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 annotations of the same notation passed QC, and the QC process itself is not free from human error. At the same time, as seen in Table III, the two-round quality control process apparently removed nearly 4/5 of all disagreements, bringing the withQC inter-annotator f-score of 0.97 from a noQC f-score of 0.89. On average, QC introduced *less* change than what the original differences between individual annotators were. This suggests that the withQC results are somewhere in the "center" of the space of submitted annotations, and therefore the quality control process probably really leads to more accurate annotation instead of merely distorting the results in its own way.

V. Baseline experiments

MUSCIMA++ allows developing and evaluating OMR systems on symbol recognition and notation reconstruction subtasks, both in isolation and jointly:

- **Symbol classification:** use the bounding boxes and symbol masks as inputs, symbol labels as outputs. Use primitive relationships to generate a ground truth of composite symbols, for compatibility with datasets of [11] or [2].
- **Symbol localization:** use the pages (or sub-regions) as inputs; the corresponding list of bounding boxes (and optionally, masks) is the output.
- **Primitives assembly:** use the bounding boxes/masks and labels as inputs, adjacency matrix as output.

Convincing baselines for handwritten musical symbol classification have already been established in [11]. We therefore focus on musical symbol localization and primitives assembly, for which MUSCIMA++ is a key contribution.

A. Symbol localization/segmentation

We examine a basic heuristics: skeleton graphs (SGs). Although we do not expect this baseline to be particularly strong, it could prove useful as an *oversegmentation* step, an initialization of other segmentation algorithms, and it

should illuminate what are the serious challenges posed by handwritten notation.

The **skeleton graph** (SG) G is derived from the morphological skeleton S of the binary image. Each endpoint (skeleton pixel with at most one 8-connected neighbor in S) and junction (set of neighboring skeleton pixels with more than 2 neighbors in S) forms a vertex of the skeleton graph, and every vertex pair $u, v \in G$ such that there is an 8-connected path $p \subset S$ from u to v, on which no other vertex v' lies, forms an edge e in G. When computing S, we smooth the foreground boundary by first dilating the image with a 3x3 square structuring element, then eroding it with a 5x5 diamond. (However, evaluation metrics are computed against the unsmoothed input image.)

We compute the oversegmentation on the binary images after staff removal.

To assess the usefulness of a given oversegmentation, we want to compute the *upper bound of segmentation performance*, assuming that the proposed superpixels will not be further subdivided: if we use the given oversegmentation, how much information do we inevitably lose? This is expressed well with *area under the precision-recall curve* (AUC-PR).

This inevitable loss of information is going to happen when a superpixel spans multiple symbols, and is not a subset of any one of them. For instance, the skeleton graph might not have a vertex at the boundary of two symbols s_1, s_2, so the edge is either "sticking out" of whichever symbol we assign it to, and – as SG edges do *not* overlap, except for junction vertices – its pixels are missing from whichever s_1, s_2 we do *not* assign it to.

Because symbols can (and do) overlap arbitrarily, the oversegmentation setting is atypical in that it is a one-to-many alignment: one proposed superpixel can legitimately be a part of multiple symbols, which implies that assigning a superpixel to one symbol does not preclude assigning it to any other symbol. This enables us to treat symbols independently.

For each ground truth symbol s and its intersection $I(s, S)$ with the image skeleton S, we can find: (A) the maximum-recall assignment $A_r(s) = \cup e_{s,1}, \ldots, e_{s,i}$ of SG edges $e_{s,1}, \ldots, e_{s,i} \in E$ such that $\forall e \in A_r(s) : e \subset I(s, S)$; (B) the maximum-precision assignment $A_p(s) = \cup e_{s,1}, \ldots, e_{s,j}$ such that $\forall x \in I(s, S) : x \in A_p(s)$. The size of $A_r(s)$ relative to the size of $I(s, S)$ gives us maximum recall $rec^+(s, E)$ at precision 1.0, and the size of $I(s, S)$ relative to $A_p(s)$ gives us maximum precision $prec^+(s, E)$ at recall 1.0, *given the oversegmentation* E derived from the skeleton graph. We can then compute a lower bound on AUC-PR as $rec^+(s, E) + (1 - rec^+(s, E)) * prec^+(s, E)$. We use macro-averaging over symbols, as larger symbols are not necessarily more important (in fact, noteheads are most important, and they are some of the smallest symbols).

1) Results.: The average AUC-PR lower bound over all symbols in the dataset is 0.767, with average $rec^+(s, E) = 0.548$ and $prec^+(s, E) = 0.649$.

We also measured "hard" recall: the proportion of ground truth symbols that have at least one "dedicated" SG edge (nonzero $rec + (s, E)$), so that they can be at least found (even if not particularly accurately) without "using up" the edge and compromising the ability to find another symbol. This proportion of objects with at least one skeleton graph edge that is a subset of $I(s, S)$, is, however, only 0.67, and unfortunately this disproportionately affects the most important symbols: there are 10121 out of 21356 full noteheads with $rec^+(s, E) = 0$, 194/348 such grace noteheads, and 12205/21416 stems. (However, when we measured hard recall for CCs directly, it was just 0.37.[14])

B. Notation graph construction

For primitives assembly, we establish a binary classification baseline *given gold-standard symbol segmentation and classification* for deciding whether oriented symbol pairs are related. As positive instances, we extract all 82261 symbol pairs connected by a relationship; as negative instances, we extract for each symbol all symbol within a threshold distance d_{neg}, set to 200 pixels (only 52 out of 82261 related symbol pairs are further away). As features for an oriented symbol pair u, v, we use their respective symbol classes, and the relative positions of their bounding boxes.

We used a decision tree classifier.[15] Using a random 80:20 training–test split, we obtained an f-score of 0.92 on recovering the 82261 positive instances. Note that this was achieved even without syntactic constraints (e.g.: "At least one stem per full notehead."). Most frequent problems were in recovering notehead-beam relationships: about 1 in 10 notehead–beam relationships was a false negative. This result suggests that the primary difficulty in notation graph reconstruction will be dealing with symbol detection errors.

VI. Conclusion

In MUSCIMA++, we provide an OMR dataset of handwritten music that allows training and benchmarking OMR systems tackling the symbol recognition and notation reconstruction stages of the OMR pipeline. Building on the CVC-MUSCIMA staff removal ground truth, we provide ground truth for symbol localization, classification, and notation graph construction, which is the step that performs ambiguity resolution necessary for inferring pitch and duration.

However, some requirements discussed in Sec. II, are not yet fully implemented. While stafflines, staves, and the relationships of noteheads to the staff symbols can be found automatically, it is not clear how accurately precedence can

[14]Note that skeleton graph oversegmentation will always perform at least as well as the connected components (CCs) heuristic. The skeleton of each connected component is also a connected component in the skeleton image, so if the given CC corresponds to a symbol (or is part of a multi-CC symbol), all edges in the skeleton of this CC will be assigned to $A_r(s)$ and there will be no edge from this CC which will be in $A_p(s)$ and not in $A_r(s)$. In fact, SG oversegmentation may lead to a better AUC. CC oversegmentation fails when one connected component consists of multiple symbols. However, the skeleton graph of the CC may consist of multiple edges, and some of these may be unrelated to one or more of the ground truth symbols, thereby not appearing in $A_p(s, E)$ and improving – at least – $prec^+(s, E)$.

[15]We used the `scikit-learn` implementation, setting maximum tree depth to 20 and minimum number of instances per leaf to 10.

be inferred. Second, while the variety of handwriting collected by Fornés et al. [15] is impressive, it is all *contemporary* – whereas the application domain of handwritten OMR is also in early music, where different handwriting styles have been used. The dataset should also be re-encoded in a standard format. From the available musical score encodings, the Music Encoding Initiative (MEI[16]) is a format that can theoretically represent the notation graph and all its vertex attributes.

Finally, evaluation procedures over the notation graph need to be established. We are confident that the conceptual clarity of the MUSCIMA++ ground truth definition will simplify this task, although the relationship of simple metrics such as adjacency matrix f-score to semantical correctness of the output needs to be explored.

In spite of its imperfections, the MUSCIMA++ dataset is the most complete and extensive dataset for OMR to date. Together with the provided software, it should enable the OMR field to establish a more robust basis for comparing systems and measuring progress. Although evaluation procedures will need to be developed for the notation graph, we believe the fine-grained annotation will enable automatically evaluating at least the stage 2 and stage 3 tasks, in isolation and jointly, with a methodology close to those suggested in [5], [6], or [13]. Finally, it can also serve as the training data for extending the machine learning paradigm of OMR described by Calvo-Zaragoza et al. [12] to symbol recognition and notation assembly tasks.

We hope that the MUSCIMA++ dataset will be useful to the broad OMR community.

Acknowledgment

First of all, we thank our annotators for their dedicated work. We are also thankful to Alicia Fornés of CVC UAB[17], who generously decided to share the CVC-MUSICMA dataset under the CC-BY-NC-SA 4.0 license, thus enabling us to share the MUSCIMA++ dataset in the same open manner as well.

This work is supported by the Czech Science Foundation, grant number P103/12/G084, the Charles University Grant Agency grants number 1444217 and 170217, and SVV project 260 453.

References

[1] D. Bainbridge and T. Bell, "The challenge of optical music recognition," *Computers and the Humanities*, vol. 35, pp. 95–121, 2001.

[2] Ana Rebelo, Ichiro Fujinaga, Filipe Paszkiewicz, Andre R. S. Marcal, Carlos Guedes, and Jaime S. Cardoso, "Optical Music Recognition: State-of-the-Art and Open Issues," *Int J Multimed Info Retr*, vol. 1, no. 3, pp. 173–190, Mar 2012.

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

[4] D. Byrd, "Music Notation by Computer," Ph.D. dissertation, 1984.

[5] Donald Byrd and Jakob Grue Simonsen, "Towards a Standard Testbed for Optical Music Recognition: Definitions, Metrics, and Page Images," *Journal of New Music Research*, vol. 44, no. 3, pp. 169–195, 2015.

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

[16] http://www.music-encoding.org

[17] http://www.cvc.uab.es/people/afornes/

[7] J. Hajič jr., J. Novotný, P. Pecina, and J. Pokorný, "Further Steps towards a Standard Testbed for Optical Music Recognition," in *Proceedings of the 17th International Society for Music Information Retrieval Conference*, M. Mandel, J. Devaney, D. Turnbull, and G. Tzanetakis, Eds., New York University. New York, USA: New York University, 2016, pp. 157–163.

[8] Arnau Baro, Pau Riba, and Alicia Fornés, "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, 2016, pp. 465–470.

[9] M. V. Stuckelberg and D. Doermann, "On musical score recognition using probabilistic reasoning," *Proceedings of the Fifth International Conference on Document Analysis and Recognition. ICDAR '99 (Cat. No.PR00318)*, no. Did, pp. 115–118, 1999.

[10] A. Rebelo, F. Paszkiewicz, C. Guedes, A. R. S. Marcal, and J. S. Cardoso, "A Method for Music Symbols Extraction based on Musical Rules," *Proceedings of BRIDGES*, no. 1, pp. 81–88, 2011.

[11] Jorge Calvo-Zaragoza and Jose Oncina, "Recognition of Pen-Based Music Notation: The HOMUS Dataset," *22nd International Conference on Pattern Recognition*, Aug 2014.

[12] J. Calvo Zaragoza, G. Vigliensoni, and I. Fujinaga, "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*. A Coruña: University of A Coruña, 2017.

[13] Pierfrancesco Bellini, Ivan Bruno, and Paolo Nesi, "Assessing Optical Music Recognition Tools," *Computer Music Journal*, vol. 31, no. 1, pp. 68–93, Mar 2007.

[14] H. Miyao and R. M. Haralick, "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*, 2000, p. 497506.

[15] A. Fornés, A. Dutta, A. Gordo, and J. Llads, "CVC-MUSCIMA: a ground truth of handwritten music score images for writer identification and staff removal," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 15, no. 3, pp. 243–251, 2012.

[16] Baoguang Shi, Xiang Bai, and Cong Yao, "An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition," *CoRR*, vol. abs/1507.05717, 2015.

[17] A. Fornes, A. Dutta, A. Gordo, and J. Llados, "The ICDAR 2011 music scores competition: Staff removal and writer identification," in *Document Analysis and Recognition (ICDAR), 2011 International Conference on*. IEEE, 2011, pp. 1511–1515.

[18] K. T. Reed and J. R. Parker, "Automatic computer recognition of printed music," *Proceedings - International Conference on Pattern Recognition*, vol. 3, pp. 803–807, 1996.

[19] Liang Chen, Rong Jin, and Christopher Raphael, "Renotation from Optical Music Recognition," in *Mathematics and Computation in Music*. Springer Science + Business Media, 2015, pp. 16–26.

[20] I. Fujinaga, "Optical Music Recognition using Projections," Master's thesis, 1988.

[21] B. Coüasnon and J. Camillerapp, "Using Grammars To Segment and Recognize Music Scores," *Pattern Recognition*, pp. 15–27, October 1994.

[22] D. Bainbridge and T. Bell, "A music notation construction engine for optical music recognition," *Software - Practice and Experience*, vol. 33, no. 2, pp. 173–200, 2003.

[23] M. Szwoch, "Guido: A musical score recognition system," *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, vol. 2, no. 3, pp. 809–813, 2007.

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

[25] Christoph Dalitz, Michael Droettboom, Bastian Pranzas, and Ichiro Fujinaga, "A Comparative Study of Staff Removal Algorithms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 5, pp. 753–766, May 2008.

[26] A. Rebelo, G. Capela, and J. S. Cardoso, "Optical recognition of music symbols," *International Journal on Document Analysis and Recognition*, vol. 13, pp. 19–31, 2010.