# TROMPA

TROMPA: Towards Richer Online Music Public-domain Archives

# Deliverable 3.4

# Visual Analysis of Scanned Scores

| | |
|---|---|
| Grant Agreement nr | 770376 |
| Project runtime | May 2018 - April 2021 |
| Document Reference | TR-D3.4-Visual Analysis of Scanned Scores |
| Work Package | WP3 - Automated Music Data Processing and Linking |
| Deliverable Type | Report |
| Dissemination Level | PU-Public |
| Document due date | 30 September 2020 |
| Date of submission | 1 October 2020 |
| Leader | PN |
| Contact Person | Vladimir Viro (PN) |
| Authors | Vladimir Viro (PN), Cynthia Liem (TUD), Jaehun Kim (TUD), Tim Crawford (GOLD) |
| Reviewers | David Weigl (MDW) |

## Executive Summary

In this document, we present work on visual score analysis that has been performed in the context of the TROMPA project. We describe existing systems and our experience with them, as well as techniques that we have created and implemented as alternatives to existing optical music recognition (OMR) systems.

Sheet music plays an important role in Western music, and considering currently available online public-domain resources, many PDF documents with sheet music scans can be found. While musicians already extensively play from such scores, they also have important potential for increasing digital music accessibility and enrichment, e.g. by being synchronized to recordings, by becoming searchable for motives or patterns, or by being offered as a flexible digital edition allowing for annotations. Such applications all will require the extraction of musical information from the visual information in the scanned score. At the same time, they may not all require for a full OMR pipeline to be run (i.e., going from a full PDF to a full transcription). Furthermore, considering TROMPA's interest in human-in-the-loop approaches, rather than running full OMR pipelines which will always need post-correction, other hybrid annotation workflows (as also discussed in D4.4) are possible in which intermediate output of an OMR pipeline can already be transcribed, corrected or annotated. This deliverable also focuses on extracting such intermediate outputs from a PDF file, which is the most common container format for digital sheet music.

In Section 3 we describe the general state of existing OMR systems and how they can find their place within TROMPA. We show how OMR applied to early music prints can be used for content-based searching of large collections and how this might be adapted as a general music-search strategy for TROMPA resources.

Considering intermediate output, we discuss issues and challenges of efficiently extracting raster image data from PDFs. Subsequently, we discuss various techniques to extract measures from score pages: visually structured information units within a page, for which the information extraction could be done automatically, or alternatively through embedding in human-in-the-loop frameworks. We discuss how measure extraction can both be done with deep learning based models, as well as knowledge-based methods employing more traditional image processing techniques, which are lighter-weight and more transparent to run and tune.

We furthermore discuss deep learning approaches that we have employed for recognizing coarse and fine notation elements. These approaches can be used for an end-to-end OMR system. This approach, although attractive, has its own challenges, which we discuss and which we deal with by introducing domain-specific information on the music notation structure and certain heuristics that take care of the tasks that are still challenging for deep learning models to robustly solve.

Within TROMPA, the choice was made to offer digitally encoded music in the MEI format, both because of its scholarly and open origins, as well as for its flexibility in handling partial content, and the possibility to embed it in enriched contents (e.g. in the MELD framework). Therefore, we discuss how (partial) visual analysis outcomes will be processed into the MEI format. Furthermore, we discuss how our visual analysis components are integrated in the broader TROMPA context, including connections to the human-in-the-loop workflows of WP4, as well as a set of RESTful APIs that can interact with the Contributor Environment.

| Version Log | | |
|---|---|---|
| # | Date | Description |
| v0.1 | 23 September 2020 | Initial version submitted for internal review |
| v0.2 | 30 September 2020 | Revised version after internal review |
| v1.0 | 1 October 2020 | Final version submitted to EU |

# Table of Contents

# 1. Introduction

A large portion of presently available public-domain digital music resources represents, or relates to, sheet music. Several notable online collections provide access to scanned images of sheet music in PDF format. While the PDF representation is sufficient for musicians to read and perform music, the PDF format does not encode music semantics. Therefore, the format does not easily allow for multimodal enrichment or content-based analysis. Therefore, to benefit from the musical meaning in large-scale digital sheet music collections, processing steps are needed to extract such meaning out of the scans.

Traditionally, this would involve the use of Optical Music Recognition (OMR) systems. However, OMR systems are known to be imperfect, and may not be easily integrated in more scalable and generic workflows. Therefore, as part of research efforts under TROMPA's WP3, we have both investigated existing OMR systems, but also developed new techniques to process and extract visual musical content from scans of scores.

In doing this, we will not only focus on functionality to implement a full OMR pipeline, but also on functionality to yield intermediate output, that can subsequently be processed and corrected in a human-in-the-loop fashion, thus connecting to efforts in WP4.

The remainder of this deliverable is structured as follows. Chapter 2 will discuss possible uses of visual score content that are relevant to TROMPA's broader mission of enriching and expanding the accessibility of public-domain music resources. As different uses will have different functional demands, we also discuss to what extent full OMR would be necessary, or whether intermediate or rougher output may suffice. Subsequently, Chapter 3 describes how existing OMR functionality is currently being used as part of the workflow of music scholars. After this, Chapter 4 describes pre-processing techniques that convert PDF scans to intermediate representations (i.e. representations that do not yet include fully transcribed music, but extract meaningful structural elements from a score). Such intermediate representations can be included in human-in-the-loop or separate automated frameworks. Chapter 5 follows up with a discussion of efforts towards deep learning based music object detection. Then, Chapter 6 discusses how outcomes of Chapter 4 and Chapter 5 are integrated into the MEI format, which is TROMPA's format of choice, due to its scholarly and open origins, as well as for its flexibility in handling partial content, and the possibility to embed it in enriched contents (e.g. in the MELD framework). Chapter 7 discusses how the outcomes of this deliverable are integrated and accessible in the broader TROMPA context, after which Chapter 8 provides a conclusion.

# 2. TROMPA-relevant uses of visual score information

Visual score analysis results can be useful for multiple tasks. Some require notation element level granularity, for others only large-scale features of music notation, such as systems and measure, are relevant. In the following we briefly discuss various use cases within TROMPA that benefit from visual score analysis, as well as different levels of sheet music representation that can be provided by various OMR approaches.

## 2.1 Use cases benefitting from OMR

One motivating use case for OMR in the context of TROMPA is the creation of digital score editions. In order to bootstrap this process we would like to use as many existing score editions as possible (most digitally available editions are available only as scans). Converting scans of editions in the public domain into a digital music notation format that can be further improved upon can relieve the need for the people creating the digital editions to start from a blank slate. Instead, only the OMR errors and the edition errors would need to be manually fixed. When there is only one human editor, this approach is useful only if fixing the OMR errors can be done quicker than typesetting the complete score from scratch by hand. If the process is crowdsourced, however, to an audience that is less proficient in typesetting, even significant error rates may still facilitate the requested tasks.

Typically, scholarly editions are created using proprietary tools such as Finale or Sibelius, or open encoding standards - such as MEI, which is specifically developed with scholarly use-cases in mind. We discuss MEI and its use as an OMR output format in more detail in chapter 6.

Further use cases that can benefit from OMR results are score matching (linking multiple editions together), which can be very useful to scholars and musicians comparing different editions of the same work, score similarity (ability to easily find works similar to a given work), which can be used as an exploratory tool by scholars, musicians and music enthusiasts alike, search by musical content or notation (using score similarity to query particular score excerpts), and audio-to-score linking (as for example implemented in the TuttiTempi performance comparison tool that we have previously developed; see also D3.5 on multimodal music information alignment).

## 2.2 Levels of sheet music representation

While some (particularly, commercial) OMR systems aim to generate complete digital representations of sheet music, some use cases can benefit from more limited intermediate music representations.

If one is interested in slicing scanned scores so that they better fit on screens of different sizes, knowing where the systems are on a page can be sufficient. Highlighting particular measures can be useful in the context of audio-to-score playback synchronization. If one is interested in search and similarity, one can use the positions of music notation elements on the page without having to generate a complete digital score encoding.

In the following chapters we discuss approaches that can be used to derive these representations.

# 3. Existing OMR systems and their use within TROMPA

## 3.1 Overview

Although relatively accurate OMR programs have been available on the market for around two decades, the lack of commercial incentive (in terms of likely profits) has meant that very little significant investment has been made in this technology. It has, however, occasionally been adopted as a case-study in research within standard computer-science disciplines such as image recognition and pattern detection, sometimes enhanced with such AI techniques as machine-learning and, most recently, deep learning. The latter, in particular, shows great promise for moving the technology forward, but demands an enormous effort in providing comprehensive coverage and sufficiently large numbers of examples of 'ground-truth' judgments on which such methods depend. Deep learning is beginning to yield very promising results in the very tricky domain of handwritten music.[1]

The main difference between printed and handwritten music is that in the former case the glyphs ('atomic' graphical symbols used in the notation) are more likely to be consistent in appearance, since they are traditionally produced using either type characters or punches which by and large generate identical patterns for a given symbol. Written music symbols, on the other hand, can vary greatly, and rely on the reader's or performer's own perception, cognition, and experience to compensate for this lack of consistency. (In fact, some engraved music is more like handwritten music in this regard, since in the same way the method allows almost complete freedom to create new or unusual symbols.)

Commercial and open-source OMR software is almost exclusively trained on works produced by late 19th-century engravers working in cities such as Leipzig, Mainz or Vienna; this is associated with the output of music publishers such as Breitkopf und Härtel, Schott or Doblinger, who produced large quantities of music by the great composers for domestic and professional consumption. By and large, they are similar in appearance, with some 'house style' differences in detail, which makes the development of software for OMR somewhat easier than it would be for a full coverage of styles and periods. However, the music of the 19th century and later becomes increasingly complex in terms of the texture expressed in the notation - especially so in the case of piano music, where multiple lines of intricate music need to be accommodated on a pair of staves. Furthermore, the separation of the music into voices is often deliberately disguised in an effort to make the music more 'readable' (that is, more 'human-readable') by occasional infringements of the traditional 'rules' of music-engraving. While some OMR software can produce good results with clearly-printed and well-photographed pages, this becomes less true with decreasing image-quality and with increasing musical complexity.

As a contribution within an earlier project (MetaMuse, Mellon Foundation, 2006) with Tim Crawford and others at Goldsmiths, Donald Byrd has codified many of the features of music notation that render it an extremely complex problem for image-recognition systems [2].[2] This work alone shows why OMR has not been used as the means to increase dramatically the amount of encoded music derived from historical collections of the world's music libraries, as might be hoped. If automatic score-encoding mechanisms like this were possible, the search and discovery methods of

[1] These and other aspects of OMR are explored in [1]

[2] See also the two web-pages compiled by Donald Byrd: Gallery of Interesting Music Notation ( http://homes.sice.indiana.edu/donbyrd/CMNExtremes.htm ) and Extremes of Conventional Music Notation ( http://homes.sice.indiana.edu/donbyrd/CMNExtremes.htm ).

Music Information Retrieval (MIR) would offer great advantages for the disciplines of musicology and music analysis, as well as for the general user, in making it possible to rapidly compare musical passages from comprehensive resources covering large proportions of the historical repertory. Furthermore, as intended as an outcome of TROMPA, the preparation of new editions of the standard repertory based on OMR'd public-domain materials would be made much easier.

## 3.2 OMR for Early Music

On the other hand, within specialised domains, such as early music, whose OMR requirements may be simpler than the general case, conventional methods such as pattern matching can yield reliably good results where the range of symbols is limited. Such is the case in 16th- and 17th-century typeset music (vocal or instrumental), and researchers at Goldsmith's have long-term experience in this field. In a third phase (2006-11)[3] of the Electronic Corpus of Lute Music (ECOLM) project,[4] Crawford and others carried out a number of experiments on printed lute tablatures using OMR systems based on Gamera,[5] showing that recognition accuracy of the typeset glyphs well over 90% is possible given good quality images of well-printed original sources.[6] While this still requires a good deal of manual correction, it means that much of the printed renaissance repertory of lute, cittern and guitar music can in principle be encoded semi-automatically in reasonable time.

The specialist OMR program, Aruspix,[7] developed by Laurent Pugin, was designed initially as a bibliographical tool for detecting differences between printed examples of the same book of typeset music, but has the additional advantage of generating MEI output which captures the position of symbols together with their likely musical semantics. Pugin and Crawford presented an evaluation of Aruspix's capabilities in [6].

Within the ECOLM project, a command-line version of Aruspix[8] was developed to allow batch processing, and is capable of recognising music within large numbers of typeset music page-images (taking about 1 second per page). This was used for experiments at Goldsmiths into MIR methods for early music using the British Library's Early Music Online (EMO)[9] resource; around 250 books, comprising about 32,000 pages of music, were subjected to OMR using Aruspix and an experimental ngram-based MIR system was used for retrieving pages similar at three levels: duplicate photographs; pages containing similar music; and pages containing closely-related music.

A long-term aim of the ECOLM project was to investigate the possibility of cross-searching between early vocal-music sources and instrumental arrangements of the works within them in tablature. There are two major problems in this research: the voice-leading of music in lute-tablature is non-specific, so that extracting 'horizontal' lines of melody from the voices present in the tablature is a hard problem; secondly, the music in the arrangements tends to be elaborated creatively, so

[3] http://doc.gold.ac.uk/isms/ecolm/ECOLM_III_pres.pdf
[4] http://www.ecolm.org
[5] https://gamera.informatik.hsnr.de/index.html
[6] https://www.researchgate.net/publication/281267395_From_Facsimile_to_Content_Based_Retrieval_the_Electronic_Corpus_of_Lute_Music
[7] http://www.aruspix.net
[8] https://github.com/DDMAL/aruspix
[9] https://www.royalholloway.ac.uk/research-and-teaching/departments-and-schools/music/research/research-projects-and-centres/early-music-online/

that the original vocal lines are often very hard to detect computationally. Work by Dr. Reinier de Valk using machine-learning techniques - partly carried out at Goldsmiths - has led to much improvement in both aspects [7][8].

## 3.3 Applications of OMR: Full-Text searching of Early Music Online (F-TEMPO)

Continuing within TROMPA, and in parallel with such recent OMR developments, Crawford at Goldsmiths has developed F-TEMPO (Full-Text searching of Early Music Prints Online),[10] a highly scalable and efficient method for MIR which uses a novel feature extracted from the Aruspix-generated MEI, Minimal Absent Words (MAWs).[11] This currently searches over 500,000 pages of early music in under one second, and would also be suitable for use with conventional (modern) music notation recognised using an OMR system such as Audiveris on the public-domain sources within TROMPA. Using F-TEMPO's RESTful API, TROMPA will be provided by the end of the project with a music-content retrieval facility using F-TEMPO (suitably adapted) as an external task within the CE. (NB the current version of F-TEMPO does not use or return source metadata, though this will shortly be provided in the next phase of development.)

The feature-extraction and indexing that enables F-TEMPO is carried out in the following steps: image-preparation (verifying image-quality; conversion into TIFF images; splitting two-page spreads into single pages), recognition (processing each page-image with the command-line version of Aruspix, which first segments the image into text, graphics and music staves, then crops and binarizes the latter, and generates a compressed 'axz' file containing a copy of the black-and-white binarized and cropped page-image, locations and other details of the segmented page-regions and the recognised music both in an internal format and in MEI) and indexing (a continuous character string representing the diatonic interval-sequence derived from the MEI is extracted, then MAWs are further extracted, using open-source software developed by Solon Pissis,[12] and saved together with an ID code for the page). Run as an offline batch process, this takes an average of approximately 7 seconds per image. At present there is no attempt to separate music and non-music pages, with the result that sometimes Aruspix generates 'recognised music' output from non-music pages, such as title-pages or tables of contents. However, since this very rarely has any musical coherence, it only occasionally produces misleading false positive retrieval results. It is hoped that in the near future, by using specialist algorithms in the pre-processing phase for the purpose, a considerable proportion of such non-music pages can be eliminated from indexing.

Retrieval using the F-TEMPO web-interface is achieved by a simple count of the MAWs in common between a query page and the pages of the entire collection; this linear process is made more efficient by distributed processing, currently involving 27 virtual servers each handling up to approximately 30,000 pages each. At query time, the identical query is sent to each server and truncated best results from each are concatenated and re-sorted using Javascript within the user's web browser; typical queries take around a second, though delays in presenting the user with images from the collection can be caused by network latencies beyond the system's control. A

[10] http://f-tempo.org

[11] Roughly speaking, minimal absent words are a small subset of the words that are not present in a document having the property that if a single character were to be removed from their beginning or end, the resulting substring would be present in the document. A useful introduction to MAWs can be found in [3].

[12] https://github.com/solonas13/maw

further facility is the possibility of uploading an image to the main F-TEMPO server and using this as a query to determine whether the same music is represented in the collection; this can be used to identify otherwise unknown music in a user's image which lacks metadata.

At present the API for F-TEMPO (based on HTTP POST requests) can be used to perform searches with two types of query: one of the internal page-IDs (so that, for example, a sequence of results for multiple pages can be recovered and analysed in a further offline process); or a diatonic interval string in the format used described on the F-TEMPO website at http://f-tempo.org. In the near future, it is intended that searches for uploaded images can be done via the API, e.g. for batch processing. The API returns results in the JSON format.

As examples, here are two commands which can be used on a Unix-like system to recover results for the same page of music by Orlando de Lasso[13] from F-TEMPO; in both cases, the query page is itself the first 'hit'. (Other information returned: the number of MAWs in common, "num"; the total number of MAWs in the target page, "num_words"; and the Jaccard distance between the pages based on these numbers, "jaccard".) Results are ranked by decreasing Jaccard distance (by count of common number of MAWs):

1. Query by internal F-TEMPO ID (on one line):

```
curl -s -d '{"id":"D-Mbs_bsb00091845_00488","num_results":"5"}' -H "Content-Type:
application/json" -X POST http://f-tempo-mbs.rism-ch.org/api/query
```

... produces JSON:


```
[{"id":"D-Mbs_bsb00091845_00488","num":43,"num_words":43,"jaccard":0.022727272727272707},{"
id":"D-Mbs_bsb00089980_00398","num":19,"num_words":50,"jaccard":0.7466666666666666},{"id":"
D-Mbs_bsb00084674_00632","num":11,"num_words":41,"jaccard":0.8513513513513513},{"id":"D-Mbs
_bsb00071839_00241","num":9,"num_words":34,"jaccard":0.8695652173913043},{"id":"D-Mbs_bsb00
071986_00121","num":9,"num_words":43,"jaccard":0.8846153846153846}]
```


... which may be interpreted as this ranked list:

```
1       D-Mbs_bsb00091845_00488       0.022727
2       D-Mbs_bsb00089980_00398       0.746667
3       D-Mbs_bsb00084674_00632       0.851351
4       D-Mbs_bsb00071839_00241       0.869565
5       D-Mbs_bsb00071986_00121       0.884615
```

2. Query by diatonic interval string (or 'codestring'):

```
curl -s -d
'{"codestring":"-CcDcaBcDcacabDaC-cCjGacAdAyC-cDCbcAAdCaA-cAcAcAacAcAaud-B-a--AcAcCu","num_
results":"5"}' -H "Content-Type: application/json" -X POST
http://f-tempo-mbs.rism-ch.org/api/query
```

... which produces identical JSON to the ID query above.

Note that in the latter case, editing, correcting or otherwise altering the codestring will often produce very similar best matches. However, there is a minimum length of codestring from which MAWs can be generated, so this is not generally useful for searching for short passages such as musical motifs or incipits. By substituting conventional textual ngrams for MAWs in an alternative index, shorter queries can in fact be accommodated; this has been tested with experimental versions of F-TEMPO but has not yet been implemented for the full database because of the extra programming necessary to provide match-locations within a page - this will come in the next phase of F-TEMPO development.

[13] Bass part of the chanson, 'Comme la tourterelle', from the 1570 collection, *Mellange D'Orlande De Lassvs, Contenant Plvsievrs Chansons, Tant en Vers Latins Qv'en Ryme Francoyse. A Qvatre, Cinq, Six, Hvit, Dix, Parties* (copy in the Bayerische Staatsbibliothek, Munich)

## 3.4 Future work on F-TEMPO

For use with conventional modern music notation, an OMR program, such as the open-source Audiveris,[14] could be used in the indexing process in place of Aruspix, as long as it outputs recognised music encoded as MusicXML or MEI, from which it is trivially easy to derive the F-TEMPO indexes.

A further enhancement to F-TEMPO, on which we have carried out some preliminary trials, would be to use the open-source OCR program Tesseract to recognise fragments of text from the lyric-syllables underlaid beneath the notes in the printed music. The location of all such text-fragments is reliably computed by Aruspix, but as they are almost certain to be incomplete as words, the normal recognition algorithm in Tesseract (which makes use of standard language dictionaries, including Latin) does not much enhance recognition, although this step can be bypassed. By concatenating the recognised text-fragments from a page, an alphabetic string (similar in some respects to the kind we derive from the MEI) can be used for indexing purposes. Our informal experiments used this principle on works which set the standard liturgical Latin texts likely to be found in the *Liber Usualis*;[15] we cross-searched the online version of the *LU* with our indexes and were able to detect matches in most cases, although a good deal more work is needed to make the process robust and generally useful. In particular, the wide variety of fonts and glyph-shapes used by 16th-century printers (further complicated by the frequent use of abbreviations to save horizontal space) can often be severely damaging to recognition accuracy; as a consequence, the current state of OCR is not very helpful for reconstructing full underlaid texts to anywhere near scholarly standards, although it remains useful for the limited task of document indexing.

[14] https://github.com/Audiveris/

[15] https://ddmal.music.mcgill.ca/research/omr/Search_the_Liber_Usualis/

# 4. Pre-processing to structural intermediate output

Currently-available OMR systems are not at a performance level to allow out-of-the-box application to scalable, varied repertoires of interest. Some of them are specialized to particular repertoires (e.g. Aruspix is specialized in Early Music), while in general, post-correction of outcomes is almost always needed, especially for more complex scores.

Considering TROMPA's interest in human-in-the-loop approaches that can serve as wide of an audience and as many public-domain repertoires as possible, it may not be required to implement a full OMR pipeline, in which a scanned score PDF is given as input to a black-box system, with a full score transcription as the result. Instead, intermediate outputs of an OMR pipeline may be considered, on which partial recognition, transcription or correction steps can be applied. In this chapter, we describe how intermediate output describing visually structured elements (page images and measures) within a score are being generated within the TROMPA project, where the next chapter will focus on musical object detection. In the current chapter, first, we discuss how conversion is done from PDFs to raster images. Subsequently, we describe efforts to extract measures from scanned score images, which are used in the context of WP4, in particular the Hybrid Annotation Workflows, as reported on in D4.4.

## 4.1 From PDFs to raster images

While sheet music is often stored in PDF files, usually, OMR systems expect raster images as input. Also for our intermediate processing purposes, raster images will be used as the main source of visual content. Commonly, each page of the PDF should be converted into an image.

Extracting raster page images from PDFs containing scanned sheet music is not completely straightforward. Open-source libraries such as `pdf2image` in Python can extract JPEG images from PDF files. The extracted images are sufficient for the measure detection tasks described in the current chapter. However, the deep learning based musical object detection techniques described in the next chapter have higher demands, and require the extraction of lossless PNG images.

The PDF format is actually complex: PDF pages may contain (potentially multiple) raster images, vector images, or combinations of both. This can cause libraries that export the image contents of the PDFs to estimate the optimal image resolution incorrectly, producing either very coarse images (which results in information loss and reduced OMR accuracy, especially where small notation elements are concerned), or huge images (which may significantly slow down or crash the export or the downstream processes) when exporting the images for further processing.

It is important to export the raster images at their correct inherent resolution in order not to lose information, and to render vector images at a sufficiently high resolution without overloading down-stream processes. A source of confusion may be either tiny raster images embedded in vector graphics (e.g. a low-resolution background layer), with the rendering algorithm mistaking the dimensions of the raster image as the page dimensions, or the vector images that are rendered with at an exceedingly high resolution in absence of any raster reference. It is usually possible to specify the target image dimensions without knowing the inherent dimension of the pages when exporting page images, but when applied to pages containing raster images this would often change their dimension, thus introducing an information loss.

So far, best results for lossless PNG image exports were achieved with commercial PDF libraries, in particular JPedal[16]. Output from this library was therefore used for the work described in the next chapter.

## 4.2 From images to measures

The hybrid annotation workflows as described in D4.4—and generally, TROMPA's vision of human-in-the-loop digitization efforts—require for more complex musical tasks to be broken down into smaller-scoped sub-tasks. Such sub-tasks can be framed as crowdsourcing tasks, which can be addressed in a distributed fashion.

A main human-in-the-loop task of interest in TROMPA considers the transformation from PDF score scans to MEI transcriptions; in other words, the task that traditionally has been addressed in OMR. In a human-in-the-loop setting, the OMR need not be fully automated; on one extreme end, similar to existing crowd-powered initiatives like MuseScore's OpenScore[17], the information in a PDF score scan may be fully manually described by human experts. However, under a human-in-the-loop paradigm, this would be done with many human experts transcribing small parts of a score in a distributed fashion, rather than one expert transcribing everything. At the same time, a similar workflow setup can be foreseen, in which transcription may be a partially automated (and possibly error-prone) procedure, after which human experts are requested to verify, adjust, or post-correct intermediate output.

In all these situations, we wish to avoid operating on a full musical score; instead, we want to scope down to the level of measures, and possibly even a single measure at a time, to break up and simplify these tasks. This means we will need to perform measure detection. Considering the various use cases within TROMPA, repertoires of interest extend from early music to piano music and orchestral scores. In the latter case, full scores can be very complex from an OMR perspective, with many parts playing at the same time. At the same time, content complexity of individual measures within orchestral parts usually will be much lower, and thus easier to oversee.

### 4.2.1 Using an existing CNN-based measure detector

We have evaluated the measure detector by Waloschek, Hadjakos and Pacha [4], which is employing a convolutional neural network (CNN) to detect measures in a scanned score. However, this detector has turned out suboptimal for our purposes. First of all, especially for orchestral music, the released CNN model was observed to make many mistakes, even for high quality scans with straight barlines. Examples of common errors of the model are visualized in Figure 4.1. Retraining the model would require considerable costly annotation efforts. As a second disadvantage, both in terms of storage and computational demands, the model is resource-intensive and slow to run, making it a considerable performance bottleneck, while the measure detection task itself seems reasonably simple.

The model has a third disadvantage: it only can detect measures as blocks containing multiple voices or staves. While this is semantically justified (all voices within the block would have the same measure number in a score), considering potential crowdsourcing tasks, a more refined segmentation will be needed than this, that also can separate individual parts or voices. This is

[16] https://www.idrsolutions.com/jpedal/
[17] https://musescore.com/openscore

desired to reduce the amount of information that will be displayed to a user at a time; showing a measure with both systems of a piano score maybe comprehensible, but a measure encompassing a dozen of instrumental voices in an orchestra score may be too complex for the user to process, while the information within each voice will be relatively compact.

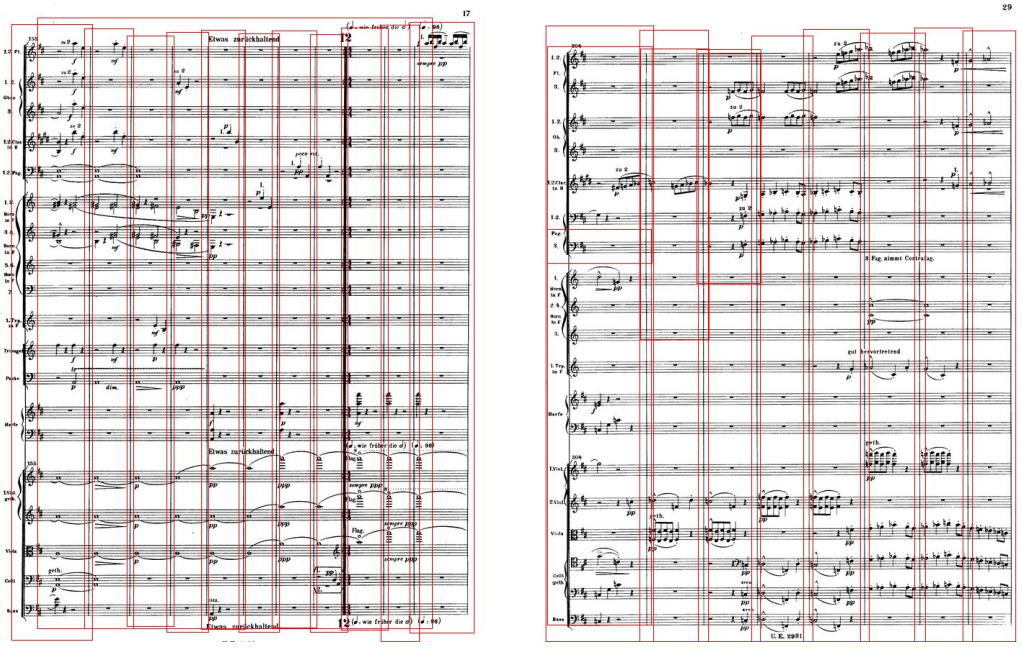

**Figure 4.1** Illustrations of common errors observed for the CNN method of [4]. On the left, multiple measures are wrongfully detected together as an extra single measure, while on the right, smaller subsections of measures are wrongfully detected as measures.

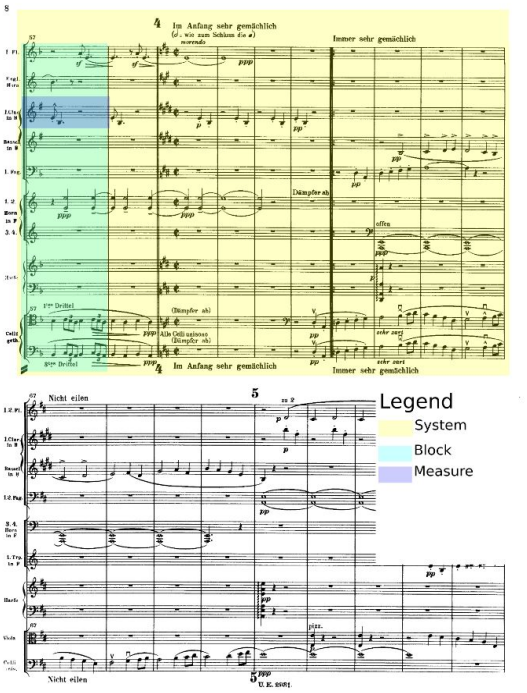


**Figure 4.2** Structure of a score.

### 4.2.2 Knowledge-based image processing

As an alternative to the CNN-based method, we have been researching a knowledge-based approach, rooted in more traditional image processing techniques. Considering a taxonomy of systems, blocks and measures, as illustrated in Figure 4.2, if multiple systems exist on the same page, they are visually separated by whitespace. Blocks are separated from each other by a vertical barline. Within a block, each individual measure involves a staff with 5 horizontal lines. Thus, by considering how intensity patterns fluctuate in horizontal and vertical directions, it should be possible to extract measures in a more heuristic fashion.

Before any segmentation is done, some standard pre-processing is performed on the page raster image. First, the contrast of the page is maximized, after which the page is binarized. Following this, any rotations in the page that might have occurred due to scanning of the original score are rectified.

Following these steps, a top-down approach is followed in analyzing the page structure. First, systems will be separated, which are subsequently segmented into vertical blocks, which are then segmented into measures.

The first segmentation considers musical systems. As there is visible and consistent whitespace in between systems, they do not have any connected components between them. Hence, when applying binary propagation to the image, each of the systems will be filled, allowing for easy detection of one or a few large blocks on the page, each of which is a system.

Next, each of these systems are segmented into vertical blocks, making use of the horizontal intensity profiles of the system. The vertical barlines that separate blocks, span almost all of a systems' height. Therefore, when considering a systems' intensity profile over the horizontal axis, peaks occur whenever a vertical barline occurs. Thus, finding these peaks corresponds to finding the locations where a system should be segmented into separate blocks. After this step, the output will consist of blocks of measures, similarly to the CNN-based detector discussed in the previous subsection, but now obtained in a much lighter-weight fashion.

After the blocks have been segmented, each block will be segmented into measures. There are currently two methods investigated for this correction: a smallest-intersection method, and a largest-region method.

The smallest-intersection method works in two parts. First, considering the vertical intensity profile of the system, we consider subsequent groups of 5 small intensity peaks (signifying the 5 horizontal lines in an individual measure). Treating a group of peaks as one broader peak, we choose the middle point between these subsequent broader peaks as segmentation 'baseline'. However, notes and annotations are not restricted to only occur within the 5 horizontal lines of a measure; in fact, they can occur outside of these lines, and even get quite close to a measure right above or underneath. To avoid for this information to be segmented wrongfully, we search within a predefined distance surrounding the 'baseline' for points with the least amount of horizontal information (i.e., many white pixels), and pick the point closest to the 'baseline' as segmenting point.

The largest-region method departs from the same first step as the smallest-intersection method, but does not set a segmentation baseline. Instead, the space between broader peaks (corresponding to zones with 5 horizontal lines in an individual measure) is separated into empty regions, where regions are separated from each other in a threshold-based fashion. The largest region is chosen, as this indicates the largest 'empty' part between two measures. Then, the middle of this region is

chosen as the segmenting point. In Figure 3, two examples are given of segmentation results from the largest-region method.

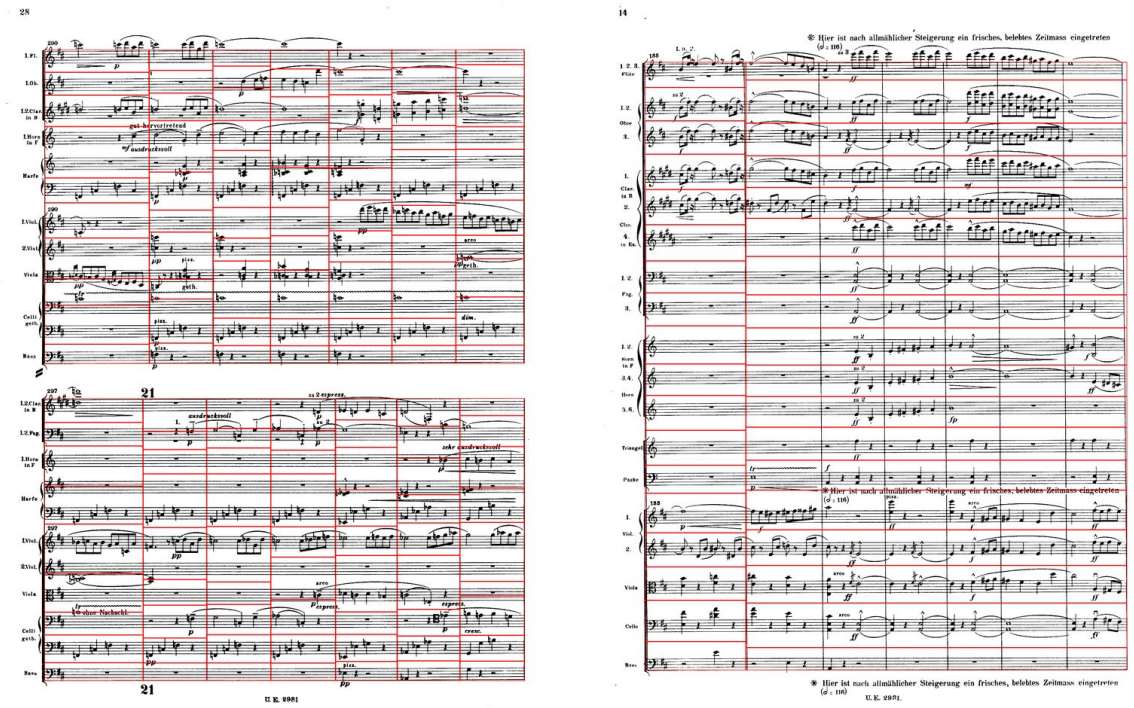

**Figure 4.3** Segmentation result examples obtained through the largest-region method.

### 4.2.3 Future work

The measure detector described in Section 4.2.2 is considerably lighter-weight and more transparent than the detector from Section 4.2.1. As such, we will prefer to continue working with this detector in further integration tasks. At the same time, beyond current manual experiments on several dozens of pages from IMSLP orchestral symphony scores from the Classical and Romantic periods (Mozart, Beethoven and Mahler), there still is need of a more comprehensive and systematic evaluation procedure.

Evaluation cannot be performed in an off-the-shelf fashion; existing bounding box annotation datasets do not contain many orchestral examples, and multiple alternative bounding boxes may differ in terms of absolute coordinates, but be equivalently good for the same measure. Furthermore, while not needing extensive training, our knowledge-based measure detector still needs for various thresholds to be tuned, which currently is performed manually. Therefore, for a more systematic performance diagnosis, we currently are developing additional benchmarking and evaluation tools that will allow for side-by-side comparisons of bounding box outputs, together with model (hyper)parameters. This will allow for the practitioner to comparatively examine outputs of alternative models and (hyper)parameter choices, connected to various possible performance metrics.

In addition, the output of our measure detector is intended to be further used to refine task generation and prioritization. Especially in orchestral scores, many measures contain highly similar information, that only may need transcription once (e.g., as an orchestra does not play tutti all the time, there often will be full-measure rests in multiple parts). Therefore, if similar visual content can

be clustered, crowd work effort can be distributed among these clusters, rather than over individual measures that may be redundant.

# 5. Deep learning approach

Existing OMR software, such as Audiveris and SmartScore, produce MusicXML output, while not exposing the intermediate steps, such as the information on location of the identified music notation elements, which can be useful on its own as described in chapter 2.

Commercial systems, such as SmartScore or PhotoScore, support only a GUI operation mode and do not support headless execution on a server, which makes them very difficult to integrate with any other software components.

Audiveris, to our knowledge the only open source OMR system targeting modern western music notation, by virtue of being open source, allows for inspection of its internal state and integration with outside software. In our experience, however, Audiveris suffers from a certain brittleness (crashing on a significant percentage of sheet music images drawn from a large representative sample, such as IMSLP) and inflexibility due to a lot of music notation rules being hard-coded in the code. Besides, Audiveris does not utilize GPU acceleration, which could potentially speed up the OMR process by an order of magnitude. When dealing with millions of images, this is significant.

In order to be able to take advantage of increased processing power provided by GPUs, to have the flexibility of computing only the relevant information, to improve the stability, and to have the ability to finetune the OMR software according to our needs without depending on the OMR software vendors, we have developed our own OMR pipeline that addresses our needs better than the existing OMR systems.

## 5.1 Recognizing notation elements in two dimensions

We use generic object detection algorithms in order to recognize music notation elements in the images of sheet music. We use a custom synthetic dataset of about 10,000 images of piano music. Hand-written scores are outside of our scope, but for printed notation we have achieved good performance (formal evaluation results are pending).

We distinguish between semantically similar but visually distinct elements. For example, a quarter note with stem up and a quarter note with stem down are two different classes for recognition purposes. In total we recognize 131 distinct classes of notation objects. In our experience such differentiation improves the overall recognition quality.

We used Google's TPUs for training due to a significantly higher performance and efficiency compared to a GPU setup. The current system has been trained for 5 days on one Cloud TPU v3.

## 5.2 Converging 2D to 1D

If one represents a music score in a digital file, the format is one-dimensional already due to the nature of computing abstractions. And for music data formats, the dimensionality is one also by design - the data is stored as sequence of tokens, be it XML tokens as is the case for MusicXML and MEI, or binary MIDI events together with little metadata describing the separation in simultaneous voices. Piano rolls are two-dimensional, but they can be represented canonically as a

one-dimensional sequence of note events that can potentially occur simultaneously, in which case they still can be ordered by pitch.

Our goal is to provide a system that would automatically convert the set of 2-dimensional annotations into a linear score representation. We have been experimenting with an end-to-end deep-learning approach that aims at doing this. However, the fact that the XML output formats impose certain non-local restrictions on data (for example, XML's tree structure must not be broken) makes robust application of these machine learning approaches challenging. A single error at a wrong place can make the whole output invalid. And unlike with HTML, for which extremely lenient parsers have been created that can gracefully handle all kinds of erroneous HTML, there are no such parsers for MusicXML or MEI.

In section 6.2 we describe a compromise approach that takes the annotations produced by the object detection module and then puts them into the MEI tree structure according to a set of heuristics which, while being less flexible, guarantee syntactic correctness of the produced XML. In the meantime we continue to pursue a robust end-to-end approach that promises a higher accuracy once the robustness issues are handled.

# 6. MEI handling

The target digital music score format chosen in TROMPA is the MEI format. Beyond MEI being an open format with scholarly roots, it also allows for scores to be partially populated with detailed content. As such, it is very well suitable for creating iteratively improving digital scores, departing from empty or partial input. In this chapter, we discuss how the visual analysis techniques discussed in the previous chapter relate to the MEI format and the establishment of digital score files.

## 6.1 Merging measure information from different sources

The measure detection steps as described in Chapter 4 are primarily intended to pre-populate an MEI skeleton with empty measures. The content of these measures should subsequently be filled in and improved upon. This can be done through various means; through fully manual entry, through more automated recognition methods, or by pre-populating the file with (possibly partial, and possibly imperfect) content obtained from another file, e.g. as the result of a black-box OMR system.

Regardless of the procedure, in all cases, the intention is for obtained MEI information to be matched to the right elements, compared, and merged into a collective MEI score. For this, both in case of larger files and single-measure input, we consider MEI as a representation of XML trees, and apply tree-based alignment techniques.

Alignment is not trivial, as the amount of elements in two comparison snippets may not be equal. To this end, we apply pairwise alignment employing the Needleman-Wunsch algorithm [5], which is a dynamic programming alignment method from the bioinformatics domain, allowing for gaps in sequences to be aligned, to deal with potential insertions and deletions between sequences to be matched. Respecting the tree-based XML structure, this procedure is recursively repeated from the roots down the trees of the two snippets to be matched.

In case multiple equivalent snippets are to be matched (e.g. multiple alternative transcriptions of the same measure), center star alignment is performed, in which one snippet will serve as center, to which other snippets are aligned.

Subsequently, for the optimal found alignment between different snippets, consensus comparisons can be done for matching nodes, which will determine what information will ultimately be merged into the collective MEI score.

Further details on the alignment and voting procedures, including worked examples, are available as GitHub documentation[18]. In addition, an example walkthrough of how a skeleton is established and how information is merged into it, is shown in an online demonstration video[19].

## 6.2 Converging 2D to 1D

In section 5.2 we have discussed the difficulty of converting the 2-dimensional notation element information to the MEI format using an end-to-end neural network-based approach. A more robust, albeit less flexible approach to the problem of conversion from a 2-dimensional to 1-dimensional representation is as follows. First, one can recognize the most structurally important elements: the systems, measures and staves. Determining the containment relationships among them is easy if one has their relatively accurate coordinates. Second, one can ascribe the remaining notation elements to the staves, again using simple proximity calculations on the corresponding bounding boxes. The temporal ordering of the notation elements is defined by their horizontal ordering in the image. In order to figure out the pitch information of pitched elements one can use their vertical offset from the staff and the staff height. The resulting MEI is valid and can be rendered to an image in a browser. This is useful in crowdsourcing context which we describe in the following section.

## 6.3 Crowdsourcing considerations

MEI provides a convenient format for crowdsourcing the correction of both score element recognition and the translation from the set of 2-dimensional annotations into the proper MEI notation, since MEI contains both types of information in the same file. Thus, any update to either 2-d labels or the MEI content under the <music> tag can be submitted to the TROMPA Contributor Environment and versioned in a version control system.

Besides, since MEI can be rendered in the browser in real-time using Verovio.js[20], it is possible to present to the users the visualization of the MEI-encoded score side by side with the original score image and the labels that have been recognized by the OMR system and possibly additionally corrected by the users. The users then can edit the automatically converted MEI code and see the resulting rendering immediately.

The same immediate interactivity is possible for the previous step - the editing of the automatically generated annotations. Using the 2-d information in the <facsimile> section of the MEI file it is possible to use one of many browser-based image annotation tools to edit the annotations in place. The conversion of the 2-d annotations to the linear notation (in MEI) can be executed quickly upon every update and followed by rendering of the MEI, so that the user can immediately see the effect of their editing on the final result. This workflow would allow users who are not versed in the MEI format and cannot edit the MEI score representation themselves to contribute to improving the OMR by correcting visual annotations while at the same time seeing the immediate effect of their

[18] https://github.com/cakefm/crowd_task_manager/wiki/Aggregator---Technical-Details
[19] https://drive.google.com/file/d/1cp_dWECMWRGB5ynuzWZZn-8_kSeF8-B-/view
[20] https://www.verovio.org/javascript.xhtml

actions on the current OMR result and visually inspecting it for any differences with the original image.

# 7. Integration

The visual analysis techniques discussed in the previous chapters are integrated or accessible to the broader TROMPA context in various ways, as described in this chapter.

## 7.1 Measure detector

Work on measure detection is instrumental to the crowd-powered music digitization pipeline, which is researched under WP4 and documented in D4.4, as well as the online GitHub wiki[21]. A global sketch of the crowd task manager architecture is given in Figure 7.1; the work described in Chapter 4 of this deliverable falls under 'Measure Detector' and 'Score Segmenter', where the steps towards a unified MEI as described in Chapter 6 are part of the 'Score Assembler'. As measure detection will still improve, updated models will be included in the same architecture, and be documented on the wiki.

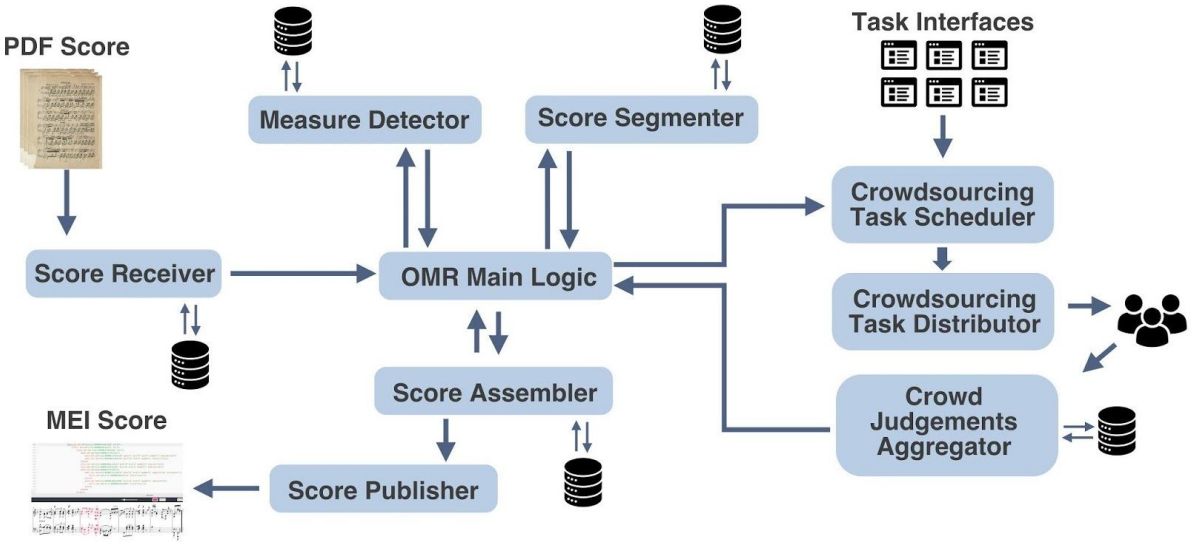


**Figure 7.1** Architecture of crowd task manager

[21] https://github.com/cakefm/crowd_task_manager/wiki/

## 7.2 Image-to-MEI OMR API

We have developed and deployed[22] an OMR API that has two endpoints[23]. The first takes a sheet music image and returns an MEI file that contains a 2-dimensional representation of notation elements recognized in the image (in the <facsimile> section), as well as the sparse semantically structured information (with proper order and containment) on systems, measures, staves and their children elements added to the staves in order from left to right (the elements, however, may lack important information, like the note classes) in the <music> section.

The second endpoint provides a convenient way of inspecting the OMR results: it takes a sheet music image and returns the same image with an overlay showing the recognized elements as colored rectangles along with their numeric certainty estimates.

Given an image of a size up to 10 MB submitted to the REST endpoint at over a POST request, the API returns an MEI file or an annotated image respectively.

The MEI and image endpoints accept the "tags" query parameter that allows for filtering the MEI and image annotations by the MEI tags that the user is interested in. For example, appending "?tags=note,slur" to the URL will result in the MEI file containing only information about notes and slurs (although the system, measure and staff elements are always returned as well), and the visualization endpoint will only highlight those elements, which can be useful due to a potentially high number of elements recognized in the image (the current API returns up to 2,000 elements per image) - see the Appendix B for an example of a complete visualization.

The OMR API takes 0.6 seconds per page on a CPU, 0.08 seconds per page when using a Nvidia 2080 Ti GPU.

The visualization endpoint also accepts GET requests along with an URL of the image that should be analyzed. This makes it possible to see in the browser the results of the OMR process on any image available online by simply embedding an image with the source that includes the URL of that image, like this:

```
<img
src="https://alpha.api.omr.peachnote.net/omr/boxes/jpg?url=http://www.ga
memusicthemes.com/sheetmusic/personalcomputer/thesims/buildingtheme3/The
_Sims_-_Building_Theme_3_-_Since_We_Met_Sheet_Music_by_Pseudo_1.png" />
```

[22] This work used the EGI infrastructure with the dedicated support of the CESNET-MCC provider

[23] https://alpha.api.omr.peachnote.net/omr/mei
https://alpha.api.omr.peachnote.net/omr/boxes

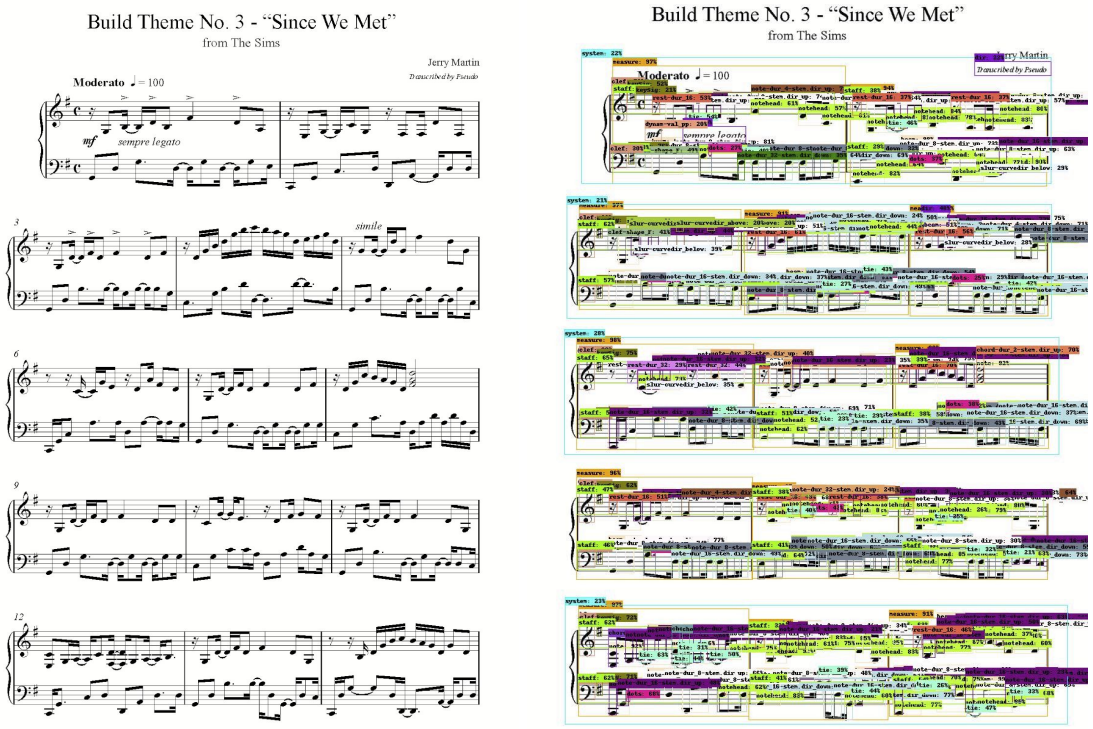


**Figure 7.1** *Example source and annotated images*

Given a local file page.jpg the endpoint that returns MEI can be queried in bash using curl as follows:

```
(echo -n '{"image" : ","'"$( base64 ./page.jpg )"'"}') | curl -H
"Content-Type: application/xml" -d @- -X POST
https://alpha.api.omr.peachnote.net/omr/mei
```

For examples of MEI output and a visual presentation of the extracted 2-dimensional information as provided by the API please refer to Appendices A and B.

# 8. Conclusion

In this document we have discussed the uses of visual sheet music analysis, its challenges, and presented the work on it done prior to and within TROMPA. We have developed a number of novel components that are currently being integrated into the crowdsourcing workflow within the scope of WP4, and we are looking forward to exploring the exploitation potential of applications that we can build on top of this work (more on this in D7.3v3, the final deliverable on exploitation).

# 9. References

## 9.1 References

[1] J. Calvo-Zaragoza, J. Hajič Jr. and A. Pacha, 'Understanding Optical Music Recognition', *ACM Computing Surveys* (July, 2020), https://doi.org/10.1145/3397499
[2] Byrd, Donald; Simonsen, Jakob Grue (2015). Towards a Standard Testbed for Optical Music Recognition: Definitions, Metrics, and Page Images". *Journal of New Music Research*, vol 44 (3): 169–195. doi:10.1080/09298215.2015.1045424
[3] A. Pinho, P. Ferreira, S. Garcia & J.Rodrigues, 'On finding minimal absent words', *BMC Bioinformatics*, volume 10, Article number: 137 (2009)
[4] Waloschek, S., Hadjakos, A., & Pacha, A. (2019). Identification and Cross-Document Alignment of Measures in Music Score Images. In *ISMIR* (pp. 137-143).
[5] Needleman, S B. & Wunsch, C. D. (1970). "A general method applicable to the search for similarities in the amino acid sequence of two proteins". *Journal of Molecular Biology*, vol. 48, iss. 3, pp. 443-453.
[6] Pugin, L. and Crawford, T. (2013). "Evaluating OMR on the Early Music Online Collection". In *ISMIR* (pp. 439-444).
[7] de Valk, R. (2015). "Structuring lute tablature and MIDI data : machine learning models for voice separation in symbolic music representations". PhD thesis, City University London, UK, 2015. [doi]
[8] de Valk, R., Weyde, T. (2018). "Deep Neural Networks with Voice Entry Estimation Heuristics for Voice Separation in Symbolic Music Representations". In *ISMIR* (pp. 281-288).

## 9.2 List of abbreviations

Use the following table format

| Abbreviation | Description |
|---|---|
| OMR | Optical Music Recognition |
| CE | Contributor Environment |
| API | Application Programming Interface |
| MIR | Music Information Retrieval |
| MEI | Music Encoding Initiative |
| GPU | Graphics Processing Unit |

# Appendix A: Sample abridged MEI output of the OMR API

```
<?xml version="1.0" encoding="utf-8"?>
<mei xmlns="http://www.music-encoding.org/ns/mei">
  <meiHead>
    <fileDesc>
      <titleStmt/>
      <pubStmt/>
    </fileDesc>
    <encodingDesc>
      <appInfo>
        <application isodate="2020-09-26T13:55:08" version="1.0.0">
          <name>
            Peachnote OMR v20200618
          </name>
          <p/>
        </application>
      </appInfo>
    </encodingDesc>
  </meiHead>
  <music>
    <facsimile>
      <surface lrx="2745" lry="3611" n="1" ulx="0" uly="0"
xml:id="surface_e3405180-ffff-11ea-a0eb-ad585484e359">
        <graphic height="3611" width="2745" xml:id="graphic_e34051ee-ffff-11ea-a0eb-ad585484e359"/>
        <zone lrx="1633.804443359375" lry="1055.9180908203125" score="1.0" type="measure"
ulx="1346.595703125" uly="672.611572265625" xml:id="zone_e33c0864-ffff-11ea-a0eb-ad585484e359"/>
        <zone lrx="1134.9295654296875" lry="1084.1168212890625" score="1.0" type="measure"
ulx="781.8655395507812" uly="656.6336669921875" xml:id="zone_e33c0a4e-ffff-11ea-a0eb-ad585484e359"/>
        ...
<zone lrx="2034.7462158203125" lry="2194.050537109375" score="0.761" type="note-dur_8-stem.dir_down"
ulx="1996.92431640625" uly="2105.322998046875" xml:id="zone_e33da868-ffff-11ea-a0eb-ad585484e359"/>
        <zone lrx="579.3828125" lry="3209.49169921875" score="0.757" type="notehead"
ulx="542.9995727539062" uly="3180.149658203125" xml:id="zone_e33dacd2-ffff-11ea-a0eb-ad585484e359"/>
        ...
      </surface>
    </facsimile>
    <body>
      <mdiv label="" n="1" xml:id="mdiv_e34bd000-ffff-11ea-a0eb-ad585484e359">
        <score>
          <scoreDef/>
          <section>
            <measure facs="#zone_e33c0fda-ffff-11ea-a0eb-ad585484e359" label="1" n="1"
xml:id="measure_e33c0f30-ffff-11ea-a0eb-ad585484e359">
              <staff facs="#zone_e33d4d32-ffff-11ea-a0eb-ad585484e359"
xml:id="staff_e33d4c7e-ffff-11ea-a0eb-ad585484e359">
                <clef facs="#zone_e33cc59c-ffff-11ea-a0eb-ad585484e359"
xml:id="clef_e33cc4e8-ffff-11ea-a0eb-ad585484e359"/>
                <keySig facs="#zone_e33e5a4c-ffff-11ea-a0eb-ad585484e359"
xml:id="keySig_e33e59ca-ffff-11ea-a0eb-ad585484e359"/>
                <meterSig facs="#zone_e33ebdac-ffff-11ea-a0eb-ad585484e359"
xml:id="meterSig_e33ebd2a-ffff-11ea-a0eb-ad585484e359"/>
                <clef facs="#zone_e3404758-ffff-11ea-a0eb-ad585484e359"
xml:id="clef-shape_F_e34046fe-ffff-11ea-a0eb-ad585484e359"/>
                <note facs="#zone_e33f18e2-ffff-11ea-a0eb-ad585484e359"
```

```
xml:id="note-dur_4-stem.dir_down_e33f1860-ffff-11ea-a0eb-ad585484e359"/>
                <note facs="#zone_e3402764-ffff-11ea-a0eb-ad585484e359"
xml:id="note-dur_4-stem.dir_down_e3402714-ffff-11ea-a0eb-ad585484e359"/>
                <artic facs="#zone_e33e90f2-ffff-11ea-a0eb-ad585484e359"
xml:id="artic-artic_stacc_e33e907a-ffff-11ea-a0eb-ad585484e359"/>
                 <note facs="#zone_e33ff06e-ffff-11ea-a0eb-ad585484e359"
xml:id="note-dur_16-stem.dir_down_e33ff01e-ffff-11ea-a0eb-ad585484e359"/>
               </staff>
               <staff facs="#zone_e33ebca8-ffff-11ea-a0eb-ad585484e359"
xml:id="staff_e33ebc30-ffff-11ea-a0eb-ad585484e359">
                                <clef facs="#zone_e33e5d26-ffff-11ea-a0eb-ad585484e359"
xml:id="clef_e33e5cae-ffff-11ea-a0eb-ad585484e359"/>
                                <keySig facs="#zone_e33de468-ffff-11ea-a0eb-ad585484e359"
xml:id="keySig_e33de3b4-ffff-11ea-a0eb-ad585484e359"/>
                  ...
                </staff>
              </measure>
              <measure facs="#zone_e33c0a4e-ffff-11ea-a0eb-ad585484e359" label="2" n="2"
xml:id="measure_e33c097c-ffff-11ea-a0eb-ad585484e359">
                <staff facs="#zone_e33c1f66-ffff-11ea-a0eb-ad585484e359"
xml:id="staff_e33c1e3a-ffff-11ea-a0eb-ad585484e359">
                  <accid facs="#zone_e34001ee-ffff-11ea-a0eb-ad585484e359"
xml:id="accid_e3400194-ffff-11ea-a0eb-ad585484e359"/>
                  ...
                </staff>
              </measure>
            <sb/>
          </section>
        </score>
      </mdiv>
    </body>
  </music>
</mei>
```

# Appendix B: Example image annotation