CLASSICAL MUSIC ON THE WEB – USER INTERFACES AND DATA REPRESENTATIONS

Martin Gasser[1], Andreas Arzt[1,2], Thassilo Gadermaier[1], Maarten Grachten[1] , Gerhard Widmer[1,2]
martin.gasser@ofai.at, andreas.arzt@jku.at, thassilo.gadermaier@ofai.at, maarten.grachten@ofai.at, gerhard.widmer@jku.at

[1]Austrian Research Institute for Artificial Intelligence (OFAI), Vienna, Austria

[2]Dept. of Computational Perception Johannes Kepler Universität, Linz, Austria

ABSTRACT

We present a set of web-based user interfaces for explorative analysis and visualization of classical orchestral music and a web API that serves as a backend to those applications; we describe use cases that motivated our developments within the PHENICX project, which promotes a vital interaction between Music Information Retrieval research groups and a world-renowned symphony orchestra.

Furthermore, we describe two real-world applications that involve the work presented here. Firstly, our web applications are used in the editorial stage of a periodically released subscription-based mobile app by the Royal Concertgebouw Orchestra (RCO) [1] , which serves as a content-distribution channel for multi-modally enhanced recordings of classical concerts. Secondly, our web API and user interfaces have been successfully used to provide real-time information (such as the score, and explanatory comments from musicologists) to the audience during a live concert of the RCO.

1. INTRODUCTION

The ways we enjoy music have changed significantly over the past decades, not least as a result of the increased use of internet and technology to deliver multimedia content. Services such as iTunes, Spotify, and YouTube offer easy access to vast collections of music, at any time and any place, through tablets and mobile telephones. Such services typically rely on APIs (Application Programming Interface, a set of HTTP-callable URL's or *API endpoints* providing certain data or functionality) to index and stream multimedia content.

These API's are often exposed (e.g. last.fm, Soundcloud) to third parties for embedding functionalities into new applications. Services and APIs such as the ones mentioned above are generally geared towards a broad audience, and offer functionality peripheral to music listening, like searching for music, and creating playlists.

As far as the music listening process itself is concerned, average listeners of popular music access music in a linear fashion, i.e., a piece is consumed from the beginning to the end. However, in the world of classical music, we observed very different requirements - there are many cases that benefit from a more content-oriented infrastructure for delivering music.

We argue there are two important characteristics of classical music that call for a more elaborate treatment of the musical content. First of all, classical pieces tend to be longer and typically have a more elaborate and complex structure than pop songs. Consequently, part of the appraisal of classical music tends to lie in the awareness, and interpretation of that structure, both by musicologists and by listeners. Secondly, as opposed to pop music, in classical music the roles of composing and performing the music are usually clearly separated. This distinction leads to a stronger notion of *piece* on the one hand, and *performance* on the other.

The desire to gain insight into structural aspects of the piece and its performance can be formulated as a use case for a general interested audience, which we will call *overseeing the music*. A second use case, *comparing performances*, is centered around the question how different performances may embody different interpretations of the same piece. This use case may be more pertinent to musicologists, or musicians who wish to prepare their performance of a piece. In the *virtual concert guide* use case, audience members are provided with multi-modal information about the music during a concert. As all efforts towards providing a digital concert experience require considerable *editorial support* by experts behind the scenes, we explicitely consider this use case as well. See [8] for a more detailed description and some initial user feedback justifying those use cases.

It is clear that serving these use cases leads to requirements on the service infrastructure that go beyond mere streaming of the data. Most importantly, dealing transparently with synchronized multi-modal information sources including video, audio, musical scores, and structural annotations and visualizations, requires these sources to be

[1] http://www.concertgebouworkest.nl/en/rco-editions/

aligned to a common timeline, the musical time. In this paper, we present an API for dealing with multi-modal (video, audio, score) data that is geared towards these requirements. Rather than describing the API in detail, we choose to give a brief overview of the entities involved and present various prototype applications that illustrate how this API allows for an in-depth content-oriented presentation of music. In addition to these prototypes, we discuss two real-world applications that rely on this API.

2. RELATED WORK

The general idea of providing multi-modal and content-based access to music has been expressed in a variety of forms in prior work. In [10], Müller et al. present an audio player for multi-modal access to music data. The goal of the *Freischütz Digital* (cf. [11], [12]) project is the development of a multi-modal repository that comprises digitized versions of the libretto, various editions of the musical score, and a large number of audio/video recordings of performances. Dixon et al. demonstrate seamless switching between different versions of the same piece during playback in their MATCH application [3]. Raimond et al. [13] present an extensive framework for publishing and linking music-related data on the web.

As for the symbolic representation of musical scores, MusicXML [2] is the *de facto* standard format for exchange of digital sheet music, and as such it has largely replaced MIDI [3], which is frequently considered an inadequate representation, especially in the field of classical music.
Another approach towards a comprehensive representation of western music notation is the Music Encoding Initiative [5]. While we are aware of the advantages of general and flexible frameworks such as Music Ontology [13] and MEI [5], we have settled on a more stripped-down, use case-centric approach that allowed us to reach our goals quickly. We understand that this might mean a redesign of system components at a certain stage, but we believe that an agile approach is beneficial in our case.

In order to be able to process graphic score sheets, we use a custom bar line finding algorithm, since we currently have no need for a complete transcription of the graphical score. See Viglienson et al. [18] for a description of the problem and the various problems that might occur. For a general discussion of Optical Music Recognition errors and their impact on aligning score to audio, the reader may refer to [16].

3. WEB API

As already mentioned in the introduction, we did not have one single use case in mind. In order to stay as flexible as possible, we decided on implementing a Service Oriented Architecture (SOA) [4]. By explicitly representing the data in the form of HTTP-accessible JSON files, we are able to serve many different applications, either web-based ones or implemented in the form of native desktop or mobile applications (see section 3 for a brief outline of the functionality currently offered by the web API).

Because we are working with copyrighted material, we had to protect our API (and consequently, also the user interfaces) with an authentication/authorization system that provides different access levels to different users; furthermore, all communication between the front end HTML5 application and the web service API is encrypted.

3.1 Authentication and authorization

Access to all API endpoints is secured via an API key. A special API endpoint is provided that returns an API key as response to submission of username/password credentials. An API key is associated internally with a certain access level that gives the user access to a pre-defined set of resources within the service.

3.2 API resources

The main resources represented in our web service are:

3.2.1 Person

A person can either be a natural person (such as a composer or a conductor) or another acting entity (such as an orchestra).

3.2.2 Piece

A piece is the most general form of a musical composition. A piece references a composer (a person), a set of scores and a set of performances.

3.2.3 Score

A score represents the notated form of a composition. We made the distinction between pieces and scores in order to be able to represent different versions/editions or different orchestrations of the same piece. A score references the corresponding piece and a set of score images.

The score resource also hosts several sub-resources such as score images, a mapping from abstract score position in musical beats to the corresponding graphical position in the score image and information about the position of bar lines and time signature changes in the score. We have also defined a *variant* sub-resource, which represents a derivative version of the score with a certain repetition structure. This is motivated by the fact that the recording of a piece may very well not include all repetitions written in its underlying score, and this is reflected in the actual *score variant* of the recorded performance.

3.2.4 Performance

A performance represents a musical piece performed by a musician or an orchestra. Apart from the actual audio file, the performance resource also contains the alignment information. A score-to-audio alignment provides links between time instants in a symbolic representation of music (such as the beginnings of bars in a score) and corresponding time instants (e.g., the actual note onsets) in a recording

[2] http://www.musicxml.com/
[3] http://www.midi.org/
[4] http://www.opengroup.org/soa/source-book/soa/soa.htm

of the performance. Alignments have been created automatically in the first place by the approach described in [4], but they may have been reviewed and corrected by human annotators, in order to increase the accuracy of musical event positions in the audio file.

From the alignment information, it is quite straightforward to compute the musical tempo for each of these events, thus yielding a tempo-curve of the performance. As an alignment can be defined on different granularity levels, such as for each bar or each beat of a bar, an API request can include a parameter that specifies a certain granularity at which the tempo information is to be calculated (e.g., one tempo value per bar or beat). This *tempo* information is also exposed as a sub–resource of *performance* via the API.

Finally, the API provides functionality to calculate perceived loudness values from a given performance recording. In order to calculate loudness information from digital audio signals, we decided on using the LUFS (Loudness Units relative to Full Scale) measure that was introduced in the EBU R128 recommendation [17]. Like the tempo information, loudness values are available at different granularity levels specified in musical time.

4. EXPLORATIVE USER INTERFACES

In this Section, we sketch five different interactive visualizations based on the API described above. We start by discussing some general aspects and design choices of the visualizations.

In our user interface, we strictly follow the concept of *deep linking* [1]. The idea is that if a user wants to discuss an interesting musical passage in a written conversation, she wants to be able to simply send an URL to another user, who can subsequently click on the link, whereupon the receiver sees the user interface in the same state as the sender. Consider the URL `/score/?score=315&variant=40&performance=1328&position=1823.10`, which opens our score viewer interface with a configuration of a score, a score variant, and a performance audio file, and it also jumps directly to beat position 1823.10.

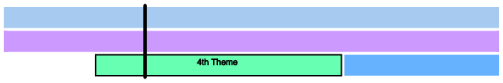
Because of the highly dynamic nature of the user interfaces, we have decided to develop a single page application [5] that talks directly to our API. In order to simplify development, improve testability of the code, and to enforce modular and reusable development, we use the popular AngularJS [6] web development framework.

The user interface prototypes are largely inspired by the use cases mentioned in section 1. While the *overseeing the music* use case has been the main motivation behind the hierarchical navigation element (see section 4.1) and the score viewer (see section 4.2), the *comparing performances* use case has led to the development of user interfaces visualizing performance-related parameters of two performances side by side. The *virtual concert guide* use case motivates the integration of a score following and

[5] http://itsnat.sourceforge.net/php/spim/spi_manifesto_en.php
[6] http://angularjs.org

(a) Overview

(b) Zoom



(c) Playback

Figure 1: Hierarchical multi–scale navigation in Beethoven's *Eroica*

real-time score display component into a mobile appllication (see section 5.2), but many components that covered the first two use cases were reused for this use case. Also, in the *editorial support* use case (which is largely covered by the application described in section 5.1), the user interface prototypes hvae proven useful, as will be discussed below.

4.1 Navigation element

Rather than providing a flat timeline, we propose a navigation element based on a multi–level segmentation of a piece. In his seminal paper [15], Shneiderman laid down some basic principles of how user interfaces for interactive visualization/navigation should be designed: First overview, then zoom and filter, then details on demand (the so-called *Visual Information Seeing Mantra*). Fig. 1 and 2 show how we reflected those principles in our user interface. Fig. 1a shows a three-level visualization: On the top level, we see the name of the musical piece (in this case, the first movement of Beethovens *Eroica*). The medium level shows the rough structure (Exposition, Development, Recapitulation, Code), and at the lowest level, the position of musical themes is shown. Fig. 1b corresponds to the *zoom* level - the user can use the mouse wheel to literally zoom into the musical structure and study the form of the piece. By dragging the playback cursor to a certain position or clicking on a structural element on any level, the score viewer (see fig. 2) shows the detailed musical notation corresponding to this position. The individual structural elements are also color-coded (this feature can be used to encode repetitions of the same section/theme by using the same color for the visual elements) and the textual annotations appear only on a certain level of detail, in order to prevent text clutter.

See the subsequent sections for a brief description of the *detail* views.

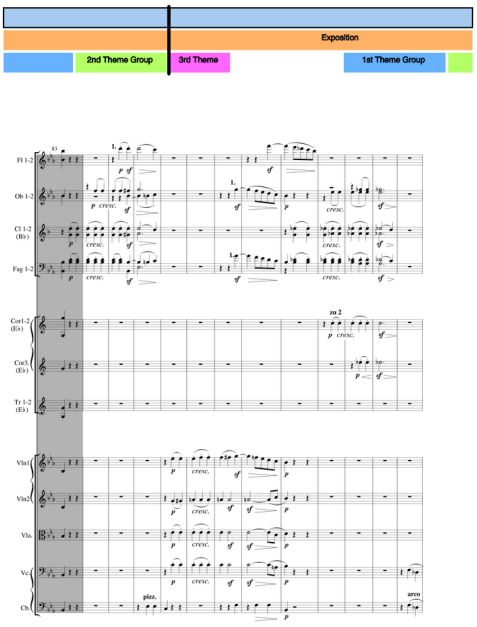


Figure 2: The score viewer with the interactive structure navigation element on top.

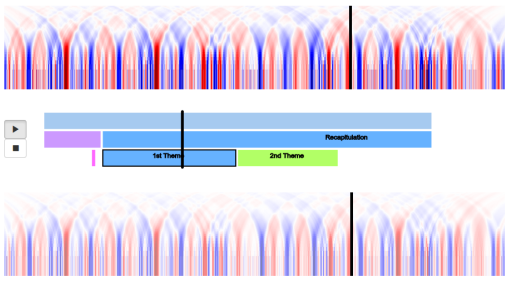


Figure 3: Dynagram visualization of two performances (differences of dynamic levels are shown, where red means increasing, blue means decreasing loudness).

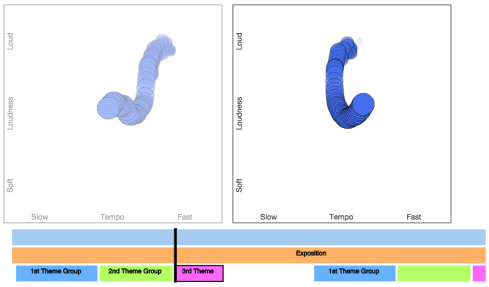


Figure 4: Performance worm visualization of two performances with structure navigation element below.

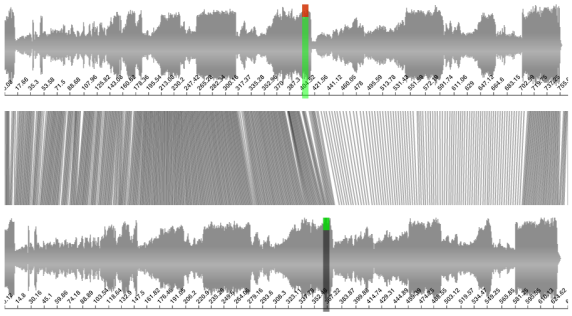


Figure 5: Direct visualization of an alignment

4.2 Score viewer

The score viewer element highlights the bar enclosing the current position (given in musical beats) on the corresponding score sheet. We are currently using scanned and annotated score sheets, but in the future, other score sheet representations (e.g., rendered on demand from MusicXML data) are imaginable and should be evaluated.

4.3 Dynagrams and tempograms

Dynagrams and tempograms show the evolution of a single parameter of the performance (loudness in the case of dynagrams, musical tempo in the case of tempograms) over time, and the parameter is shown on multiple temporal levels. This is achieved by smoothing the parameter-curve over increasing amounts of time and horizontally stacking the results, where the parameter strength is mapped to color. Fig. 3 shows dynagram visualizations of two performances of the same piece that are linked via the navigation element described in 4.1. It integrates short term variations of the respective feature with long term variations into one picture. Therefore, it allows to grasp short term events as well as long term evolution revealing more of the overall structure of the performed piece. This type of visualization builds on earlier work by Langner et al. [7], Sapp [14], and Martorell et al. [9].

We provide two different flavors of these visualizations, (1) where the parameter itself is used as input, showing the absolute values evolving over time and (2) where the derivative of the parameter is used, such that only changes (e.g. crescendo, decrescendo in case of loudness) become visible, as is shown in fig. 3.

4.4 Performance worm

The performance worm is a visualization metaphor integrating the evolution of tempo and dynamics of a musical performance over time in a two-dimensional tempo-loudness space [6]. Its purpose is to uncover hidden characteristics of shaping a performance and the relations between tempo and loudness that are characteristic of a certain style of interpretation. For a given temporal level, the 2 dimensional tempo-loudness-trajectory is displayed where

older values fade into the background as newer data-points are added on top. Fig. 4 shows performance worm visualizations of two performances that are linked via the structural navigation element below.

4.5 Alignment viewer

The alignment visualization shows the waveform displays of the audio signals of two performances of the same piece, and it connects the respective bar line positions (the downbeats) in the two performances. The resulting line pattern reflects the tempo structures of both pieces, and also how they interrelate.

Fig. 5 shows two performances of the fourth movement of Beethoven's Eroica (Wilhelm Furtwängler conducting the Berlin Philharmonic Orchestra vs. John Eliot Gardiner conducting the Orchestre Révolutionnaire et Romantique). We can conclude from the visualization the intrinsic tempo structure of the performances; while Furtwängler plays the first part slower and becomes faster in the second part, Gardiner chooses to play faster in the first part and become slower afterwards; in both performances, we can observe a strong ritardando in the middle of the piece (i.e., the tempo slows down dramatically), Furtwängler's ritardando being stronger than Gardiner's.

5. APPLICATIONS

We employed our API and our visual interfaces in two concrete applications:

5.1 Application 1: Editorial review for a multi-modal music-publishing app

Our PHENICX project partners develop a mobile app for Apple's iOS devices, and this app ist used as a distribution channel for multi-modally enriched recordings of music played by the RCO. One of their selling points is an animated view of the musical score while the audio/video is played back. Fig. 6 sketches the workflow during the production of an edition of the app, and it demonstrates how our API and the score viewer user interface are involved in this process.

The task at hand is to align a recording of orchestral music to a score representation. This problem is twofold: As the graphical score representation is often not available in machine-readable form [7], the first problem is to find the graphical positions corresponding to musical events (e.g., notes, barlines, or staves). Methods based on Optical Music Recognition turned out to be usable if high–quality graphical scores are available, but we often deal with scanned images from aged printed or even hand-written scores, where standard OMR results are unsatisfactory – therefore, manual intervention is sometimes necessary in this stage. Secondly, the alignment of a recorded performance to a synthesized performance of a score (we used the implementation from [4]) needs to be checked, and manually corrected where necessary.

[7] MusicXML does encode some score layout information, but not all

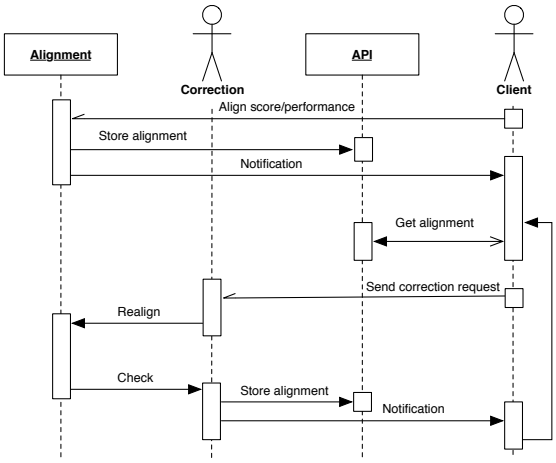


Figure 6: Production/publishing workflow

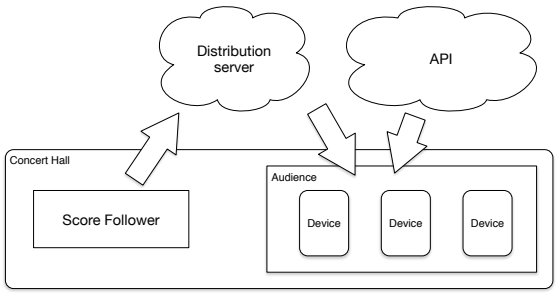


Figure 7: Live score following application

The initial score performance alignment is done by synthesizing the piece from a symbolic representation such as MIDI or MusicXML. After an internal reviewing and correction phase, our client has the opportunity to review the alignment in the score viewer interface and object in case anything is wrong – if this is the case, the alignment goes back into the internal correction phase again. The deep linking functionality – as described in section 4 – has proven to be useful in this iterative process, as it enables the client to pinpoint problematic spots very easily in written conversations. Once the client is satisfied with the quality of the alignment, it is fetched in the form of a JSON [8] file from our web service API, and further used in the app development process.

5.2 Application 2: Interactive program notes with integrated live score following

The idea is to provide the audience during a concert with additional information about the piece currently played via mobile devices. We have decided to give audience members the possibility to choose between three options, based on personal preference or expertise: (a) an interactive musical score display with the current musical position highlighted, (b) text comments by a musical expert and (c) an artistic video visualization of the music.

[8] http://www.json.org

This application also implies an editorial stage: All three options (a), (b), and (c) rely on sequenced series of events (be it the display of bar positions on a score sheet, timed text messages, or video clips that are played back at certain time instants) that have to be prepared beforehand. In this stage our API and user interfaces are already of use, as editors usually rely on a score representation to pinpoint certain annotations to the music in advance.

During the live event in the concert venue, the client applications, usually running on tablets or smartphones in the audience, access the data (score image sheets, mapping of musical positions to graphical positions) stored in our API. The score follower constantly analyzes the incoming audio stream, and sends the estimated score position to a distribution server. The distribution server subsequently forwards this information to the mobile devices that fetch score images through our API (timed text messages and videos are provided by an additional data source) and use this information to realize an enriched experience for the audience member. Fig. 7 roughly sketches the data flow in this application. We have documented the practicability of our approach in [2].

6. CONCLUSIONS AND FUTURE WORK

We have presented (1) a web service API providing access to structure- and performance-related music data including multimedia elements like score images and audio files and (2) a set of web-based explorative user interface prototypes that act as a frontend to this API. We have also presented two real–life examples of where our API and the user interface prototypes proved to be useful.

We are currently investigating various extensions to our current infrastructure; the workflow described in section 5.1 provides much potential for improvement – we are considering building user interfaces that allow application clients to directly mark alignment errors or even correct alignments directly in the user interface. By fully migrating the alignment service to be based on a complete score representation such as MusicXML instead of scanned score sheets and a MIDI as the corresponding machine-readable score representation, the alignment process would be greatly simplified and the quality of automatic alignments could be improved – in this case, we could reliably identify graphical positions of musical events, and it would be even possible to generate the graphical score directly from the symbolic representation. In addition, more detailed knowledge about instrumentation and performance parameters could also improve the quality of synthesized performances and therefore the quality of resulting automatic alignments (i.e., if a high–quality sample library such as the Vienna Symphonic Library [9], that allows for precise control of performance parameters, is used).

[9] https://vsl.co.at/

7. ACKNOWLEDGMENTS

This research is supported by the European Union Seventh Framework Programme FP7 / 2007-2013, through the PHENICX project (grant agreement no. 601166).

8. REFERENCES

[1] "Deep Linking" in the World Wide Web. http://www.w3.org/2001/tag/doc/deeplinking.html. Accessed: 2015-05-01.

[2] Andreas Arzt, Harald Frostel, Thassilo Gadermaier, Martin Gasser, Maarten Grachten, and Gerhard Widmer. Artificial Intelligence in the Concertgebouw. pages 165–176, 2015.

[3] Simon Dixon and Gerhard Widmer. MATCH: A Music Alignment Tool Chest. In *Proceedings of the 6th International Conference for Music Information Retrieval*, number Ismir, pages 492–497, 2005.

[4] Maarten Grachten, Martin Gasser, Andreas Arzt, and Gerhard Widmer. Automatic Alignment of Music Performances With Structural Differences. In *Proceedings of the 14th International Society for Music Information Retrieval Conference*, 2013.

[5] Andrew Hankinson, Perry Roland, and Ichiro Fujinaga. The Music Encoding Initiative as a Document-Encoding Framework. In *Proceedings of the 12th International Society for Music Information Retrieval Conference*, pages 293–298, Miami (Florida), USA, October 24-28 2011.

[6] Jörg Langner and Werner Goebl. Visualizing Expressive Performance in Tempo–Loudness Space. *Computer Music Journal*, 27(4):69–83, 2003.

[7] Jörg Langner, Reinhard Kopiez, Christian Stoffel, and Martin Wilz. Realtime analysis of dynamic shaping. In *Proceedings of the 6th International Conference on Music Perception and Cognition. Keele, UK: Keele University, Department of Psychology*, pages 452–455, 2000.

[8] Cynthia C.S. Liem, Ron van der Sterren, Marcel van Tilburg, Álvaro Sarasúa, Juan J. Bosch, Jordi Janer, Mark Melenhorst, Emilia Gómez, and Alan Hanjalic. Innovating the Classical Music Experience in the PHENICX Project: Use Cases and Initial User Feedback. In *1st International Workshop on Interactive Content Consumption (WSICC) at EuroITV 2013*, Como, Italy, 06/2013 2013.

[9] Agustín Martorell and Emilia Gómez. Hierarchical multi-scale set-class analysis. *Journal of Mathematics and Music*, 9(1):95–108, 2015.

[10] Meinard Müller, Frank Kurth, David Damm, and Christian Fremerey. Lyrics-based Audio Retrieval and

Multimodal Navigation in Music Collections. In *European Conference on Research and Advanced Technology for Digital Libraries*, volume 554975, pages 112–123, 2007.

[11] Meinard Müller, Thomas Pratzlich, Benjamin Bohl, and Joachim Veit. Freischutz digital: A multimodal scenario for informed music processing. In *Image Analysis for Multimedia Interactive Services (WIAMIS), 2013 14th International Workshop on*, pages 1–4, July 2013.

[12] Thomas Praetzlich and Meinard Mueller. Freischuetz digital: a case study for reference-based audio segmentation for operas. In *Proceedings of the 14th International Society for Music Information Retrieval Conference*, November 4-8 2013.

[13] Yves Raimond, Samer Abdallah, Mark Sandler, and Frederick Giasson. The Music Ontology. *ISMIR 2007: 8th International Conference on Music Information Retrieval*, 8:417–422, 2007.

[14] Craig Sapp. *Computational Methods for the Analysis of Musical Structure*. PhD thesis, Stanford University, Department of Music, 2011.

[15] Ben Shneiderman. The eyes have it: a task by data type taxonomy for information visualizations. *Proceedings 1996 IEEE Symposium on Visual Languages*, 1996.

[16] Verena Thomas, Christian Fremerey, Meinard Müller, and Michael Clausen. Linking sheet music and audio-Challenges and new approaches. *Dagstuhl Follow-Ups*, 3:1–22, 2012.

[17] European Broadcasting Union. Loudness Normalisation and Permitted Maximum Level of Audio Signals. `https://tech.ebu.ch/docs/r/r128.pdf`. Accessed: 2015-05-01.

[18] Gabriel Vigliensoni, Gregory Burlet, and Ichiro Fujinaga. Optical measure recognition in common music notation. In *Proceedings of the 14th International Society for Music Information Retrieval Conference*, November 4-8 2013.