HEARING YOUR WAY THROUGH MUSIC RECORDINGS: A TEXT ALIGNMENT AND SYNTHESIS APPROACH

Sebastian STRAHL (sebastian.strahl@audiolabs-erlangen.de) [1],
Yigitcan ÖZER (yigitcan.oezer@audiolabs-erlangen.de) [1],
Hans-Ulrich BERENDES (hans-ulrich.berendes@audiolabs-erlangen.de) [1], and
Meinard MÜLLER (meinard.mueller@audiolabs-erlangen.de) [1]

[1] **International Audio Laboratories Erlangen**, 91058 Erlangen, Germany

ABSTRACT

Annotations related to musical events such as chord labels, measure numbers, or structural descriptions are typically provided in textual format within or alongside a score-based representation of a piece. However, following these annotations while listening to a recording can be challenging without additional visual or auditory display. In this paper, we introduce an approach for enriching the listening experience by mixing music recordings with synthesized text annotations. Our approach aligns text annotations from a score-based timeline to the timeline of a specific recording and then utilizes text-to-speech synthesis to acoustically superimpose them with the recording. We describe a processing pipeline for implementing this approach, allowing users to customize settings such as speaking language, speed, speech positioning, and loudness. Case studies include synthesizing text comments on measure positions in Schubert songs, chord annotations for Beatles songs, structural elements of Beethoven piano sonatas, and leitmotif occurrences in Wagner operas. Beyond these specific examples, our aim is to highlight the broader potential of speech-based auditory display. This approach offers valuable tools for researchers seeking a deeper understanding of datasets and their annotations, for evaluating music information retrieval algorithms, or for educational purposes in instrumental training, music-making, and aural training.

1. INTRODUCTION

Researchers in the field of music information retrieval (MIR) are always looking for methods and tools to improve the understanding and accessibility of musical data. Annotations describing musical properties, such as chord labels and structural elements, are essential for the understanding of musical pieces. However, such annotations are not easily accessible while listening because they are typically provided in textual format within or alongside a score-based representation of a piece.

Visualizations play a crucial role for making such information more accessible. For example, the Score Viewer interface presented in [2] visualizes the musical score of a piece while playing back a corresponding audio recording. The information contained in the score becomes more accessible for the user because the currently played measure is highlighted within the shown musical score.

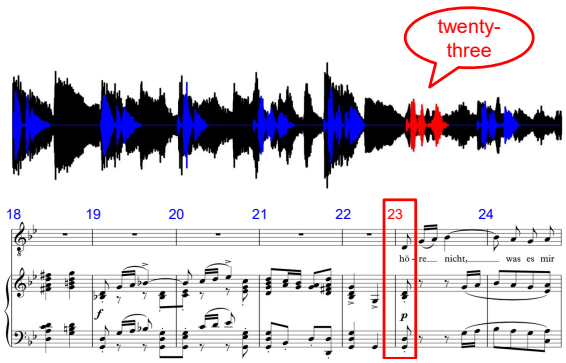

Figure 1. Illustration of synthesizing measure annotations using a song from Schubert's *Winterreise* [1]. Measure numbers are extracted from the musical score and automatically transferred to the audio recording.

Another common way of making music annotations more accessible is sonification, which refers to using non-speech audio to represent information [3]. For example, the Python package `librosa` [4] includes a function (`librosa.clicks`) that generates an audio signal with click sounds positioned at designated times, with options to adjust frequency and duration of the clicks. Moreover, the Python toolbox `libsoni` [5] provides methods for sonifying fundamental frequency trajectories or piano rolls using sinusoids.

While these approaches fall under *sonification*, a more general term for conveying information through sound is *auditory display*, which also includes speech-based methods [3]. In this paper, we introduce a processing pipeline that employs text-to-speech (TTS) synthesis as a form of auditory display, converting textual annotations into speech and allowing annotations to be acoustically integrated into the music recording. This approach offers a flexible alternative to previous methods, allowing users to access a variety of annotations while listening.

An example is shown in Figure 1, where we use this approach for announcing measure numbers within a record-

ing. While measure numbers can be directly read off from the musical score, they are not accessible within a recording. Measure number annotations with respect to a specific recording are obtained by identifying the musically corresponding temporal positions within the recording, e. g., using score-audio alignment techniques [6, 7]. After this step, our approach uses a TTS system [8–10] for synthesizing the annotations, and then acoustically superimposes generated speech and the given recording.

This paper is structured as follows. In Section 2, we describe the individual steps of our processing pipeline and which customization options are available to the user. Then, in Section 3, we present several case studies of how this approach may be used by MIR researchers, including text comment synthesis of measure positions in Schubert songs, chord annotations for Beatles songs, structural elements of Beethoven piano sonatas, and leitmotif occurrences in Wagner operas. Finally, in Section 4, we conclude the paper.

2. PROCESSING PIPELINE

The processing pipeline comprises four steps. First, the desired information regarding musical events needs to be extracted from the score. Second, all events need to be localized within the given recording by temporally aligning score and audio. After that, a suitable text snippet needs to be generated for every event, which is then converted into an audio signal using a TTS system. Finally, the synthesized text comments are post-processed and superimposed with the given recording. All of these steps are explained in more detail in the following subsections.

2.1 Music Information Retrieval

In this initial step, we gather information regarding musical events of interest for the given recording. One option is to extract that information from the musical score corresponding to the same piece. Certain types of information may be directly available, e. g., measure numbers are usually written at the start of every staff system, or the chords being played in pop songs are written above the lyrics in a chord chart. Other types of information, however, may not be specified explicitly but need to be extracted by analyzing the score, e. g., structural information, or the occurrences of important patterns such as melodies or motifs.

Depending on the type of information, the retrieval process may be carried out manually or using automated methods. Section 3 provides examples of how this step may be performed in different scenarios.

Within this step, we only need to find the position of every musical event with respect to the timeline of the score, i. e., the position on a continuous measure axis. In the next step, we align the measure timeline of the score with the physical timeline of a specific recording in order to associate every event with a physical point in time.

2.2 Temporal Alignment

We now convert measure positions with respect to the score into absolute time positions within the recording. To this end, one can employ score-audio synchronization [6], where the goal is to find for every position in the score the musically corresponding position in the recording. A common approach for music synchronization is to first convert both versions into more suitable representations, e. g., sequences of chroma features, which capture harmonic and melodic information, and then perform the alignment of these representations using techniques based on dynamic time warping [7, 11–13]. After this step, we have a list of musical events together with their respective time positions on the physical timeline of the recording.

2.3 Text Comment Generation and Synthesis

Next, we need to find text snippets that describe the musical events and that can then be given as input to the TTS system. For some types of events, this step is straightforward, e. g., measure numbers in numerical format can typically be directly articulated by TTS systems. Other events may require a mapping to a suitable format, e. g., chord annotations like D♭min first need to be brought into a format like "D flat minor". If required, such a mapping needs to be defined by the user.

For TTS synthesis, we rely on already trained deep learning-based models, for which we have the following requirements. First, synthesis should work on any computer, i. e., the model should not necessarily require a GPU for generating speech signals, and we do not want to rely on API-based solutions requiring internet connection. Second, multilingual synthesis should be supported in order to make the use of the pipeline attractive to users from different languages. Also, there may be musical events for which the description can not be well translated into other languages, e. g., the occurrence of leitmotifs in Wagner operas are best announced in German for that reason. Third, the generated speech should be customizable in speed, e. g., to avoid temporally overlapping comments. Fourth, the implementation should be open source and model checkpoints should be publicly available.

Most of these requirements are fulfilled by the `TTS` Python package [1] . This package provides implementations for several architectures together with model checkpoints for multiple languages, and most of the provided models can be employed without the need of a GPU. The only requirement not fulfilled is the customizability of speed. A remedy to this problem is suggested in Section 2.4.

The majority of TTS models work with a two-step process: The acoustic model takes text as input and predicts a mel spectrogram, and then the vocoder takes this spectrogram as input and generates a corresponding time-domain audio signal. From the `TTS` package, we use models for different languages with Tacotron 2 [8] as the acoustic model and HiFi-GAN [9] or UnivNet [10] as the vocoder. Among the other options, these models provided the best impression in an informal listening test. We further noticed that it is beneficial to end the text snippet with a period for a more natural prosody in the synthesized speech.

As a result of this and the previous steps, we have a set of synthesized text comments, i. e., short audio signals, to-

[1] `https://github.com/coqui-ai/TTS`

ISBN 978-3-200-10642-0

gether with the respective time positions in the recording to which they are related.

2.4 Post-Processing and Superposition with Recording

In this last step, we post-process the synthesized text comments individually and superimpose them with the music recording. More formally, let x be the given audio signal with samples $x[n]$ and discrete time index n. The number of samples is denoted by N, and the sampling rate by F_s. Further, we assume to have M synthesized text comments c_m with $m \in \{1, 2, \ldots, M\}$ and different durations N_m given in terms of number of samples. As described in Section 2.2, we have obtained physical time positions t_m for all the comments, which are given in seconds. Furthermore, we define gain factors $w_m > 0$ for the individual comments. In the following subsections, we describe options for modifying the synthesized text comments and time positions, and for choosing the gain factors, giving the user the possibility for customization.

2.4.1 Speed

As part of this processing pipeline, we want to provide users with the option to adjust the speed and duration of synthesized text comments. As already mentioned, the TTS models employed in this work do not directly offer this flexibility. Therefore, we adopt an approach of modifying the comments' durations afterwards.

As a first step, we apply the `trim` effect provided by the `librosa` [2] Python package [4] to remove leading and trailing silence in the synthesized speech. Thereafter, we utilize time-scale modification (TSM) techniques [14, 15], which allow for changing the speed of an audio signal without altering its pitch. For this purpose, we use the `libtsm` Python package [3], which is a re-implementation of the Matlab TSM toolbox [16]. From the functions this package provides, we use the technique based on harmonic-percussive separation [17].

We provide the user with three parameters, `speed`, `t_min`, and `t_max`, to adjust the duration of the comments. The first parameter, `speed`, specifies a factor by which the speed should be changed, i. e., it modifies the speaking tempo. The remaining parameters, `t_min` and `t_max`, additionally set hard limits to the durations in seconds. Based on these parameters, we determine for the m^{th} synthesized text comment a factor α_m by which it will then be uniformly stretched. This factor describes the change in the signal's duration and can be calculated as

$$\alpha_m = \min\left(\max\left(\frac{N_m/F_\mathrm{s}}{\texttt{speed}}, \texttt{t\_min}\right), \texttt{t\_max}\right) \cdot \frac{F_\mathrm{s}}{N_m}, \tag{1}$$

where N_m/F_s is the original signal's duration. If either of the parameters `t_max` or `t_min` is set to `None`, the respective min or max term in Equation (1) is omitted, and no temporal limitation is applied in that regard.

By default, we use `speed` $= 1.0$ and `t_min` $=$ `t_max` $=$ `None`, i. e., the TTS model's original speaking tempo is preserved and no temporal limitation is applied. In the following, we denote the time-scale-modified text comments by c_m.

[2] `https://github.com/librosa/librosa`

[3] `https://github.com/meinardmueller/libtsm`

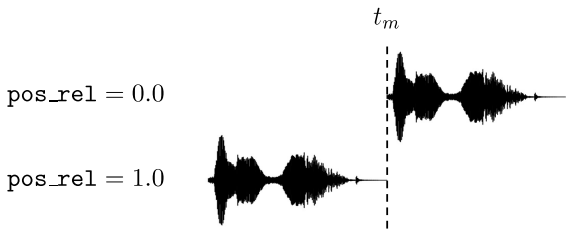

Figure 2. Illustration of relative positioning as defined by the parameter `pos_rel`. For `pos_rel` $= 0.0$, the comment starts at time t_m, whereas for `pos_rel` $= 1.0$, it ends at time t_m.

2.4.2 Positioning

Every comment is associated with a single time stamp t_m. However, it is not yet specified how the comments should be temporally positioned with respect to these points in time. For specifying the positioning, we introduce two parameters, `pos_rel` and `pos_offset_abs`. With `pos_rel`, we define the relative positioning, i. e., the fraction of the comments' durations that should have been played back before time t_m. The effect of this parameter is illustrated in Figure 2. Additionally, `pos_offset_abs` describes an absolute temporal offset in seconds, providing more flexibility in temporal positioning.

Denoting with n_m the discrete time index at which the m^{th} comment starts, this index is given by

$$n_m = \mathrm{round}\big((t_m + \texttt{pos\_offset\_abs}) \cdot F_\mathrm{s} - \texttt{pos\_rel} \cdot \alpha_m \cdot N_m\big). \tag{2}$$

In the default setting, we use `pos_rel` $= 0.0$ and `pos_offset_abs` $= 0.0$ s. In this case, the m^{th} comment starts at time t_m.

2.4.3 Loudness

As another last post-processing option, we allow for adjusting the individual loudnesses of the comments by determining appropriate gain factors w_m. Due to the complexity of the human auditory system, it is hard to measure loudness as it is perceived by humans [18]. Following industry standards, we measure perceived loudness as described in ITU-R BS.1770-4 [19], using loudness units relative to full scale (LUFS) as metric. Based on this standard, we use the *local loudness* of a given audio signal, which is time-varying and measured over segments of 400 ms length, and the *global loudness*, which describes an average loudness over the entire duration of the signal. For computing the global loudness, a gating mechanism is used, i. e., segments with a local loudness below a certain threshold are excluded from the averaging to obtain a metric that better aligns with human perception of audible content. The difference between loudness values in LUFS is denominated in loudness units (LU). The `pyloudnorm` [4] Python pack-

[4] `https://github.com/csteinmetz1/pyloudnorm`

ISBN 978-3-200-10642-0

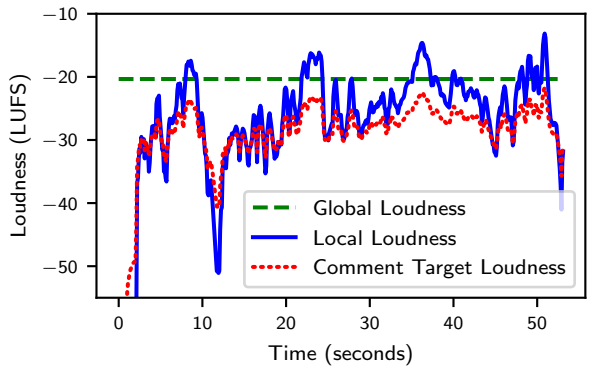

Figure 3. Loudness curves for an excerpt of a Beethoven piano sonata recording. The local loudness is computed for overlapping blocks of 400 ms, and the comment target loudness is computed using Equation (3) with the default parameters mentioned in Section 2.4.3.

age [20] implements this standard, and an example of local and global loudness for an excerpt of a Beethoven piano sonata recording is shown in Figure 3.

One option is to choose the loudness of the comments relative to the loudness of the given recording. However, it is unclear whether to choose global or local loudness as a reference. As an example, consider the time position at around 12 s of the recording for which the loudness curves are shown in Figure 3, where we observe a large gap between local and global loudness. Choosing the comment loudness based on the global loudness renders the comment orders of magnitudes louder than the recording itself, which may distract the listening experience. On the other hand, choosing the comment loudness based on the low local loudness at that position may impair the intelligibility of that comment.

Instead, we suggest using both the recording's local and global loudness to determine a comment's target loudness. We denote the local loudness of the recording over the segment where the m^{th} comment should be placed as $L_{\text{loc},m}$, and the global loudness over the entire duration of the music recording by L_{glob}.

The loudness of the m^{th} comment c_m is denoted by L_m. Here, we use the average loudness over the whole comment duration and ignore fluctuations, assuming that the comments are typically short. Let L'_m be the comment target loudness, which may be computed by

$$L'_m = \texttt{w\_glob\_loc} \cdot (L_{\text{loc},m} + \texttt{offset\_loc}) + (1 - \texttt{w\_glob\_loc}) \cdot (L_{\text{glob}} + \texttt{offset\_glob}), \quad (3)$$

where `offset_loc` and `offset_glob` are offsets in LU relative to local and global loudness of the given recording, respectively. The factor $\texttt{w\_glob\_loc} \in [0, 1]$ allows for balancing the impact of local and global loudness. As default parameters, we suggest $\texttt{w\_glob\_loc} = 0.5$ and $\texttt{offset\_loc} = \texttt{offset\_glob} = -5$ LU, where both local and global loudness of the recording are equally considered, but the comments are reduced in volume compared to the recording. Using these values, the comment target loudness curve in Figure 3 was computed.

A change in loudness denominated in LU can be realized by applying a gain of the same amount in dB. Therefore, the linear gain factors are calculated as

$$w_m = 10^{(L'_m - L_m)/20}. \quad (4)$$

2.4.4 *Superposition*

Finally, we define a comment track c as

$$c[n] = \sum_{m=1}^{M} w_m \cdot c_m[n - n_m], \quad (5)$$

and the superimposed recording y is obtained as

$$y[n] = x[n] + c[n] \quad (6)$$

for all n. Note that the length of c may exceed the length of x if there exists a comment which starts ahead of the given recording, or if there exists a comment that ends after the music recording. Furthermore, note that it is not ensured that comments do not overlap.

An overview of the parameters of this processing pipeline is given in Table 1. Depending on the need of specific applications, the post-processing may be extended in future work.

Parameter	Description
`speed`	speed factor
`t_min`	minimum duration per comment (in s)
`t_max`	maximum duration per comment (in s)
`pos_rel`	relative positioning
`pos_offset_abs`	absolute temporal offset (in s)
`offset_loc`	offset local loudness (in LU)
`offset_glob`	offset global loudness (in LU)
`w_glob_loc`	balancing factor between global and local loudness

Table 1. Overview of the parameters of the processing pipeline.

While the first two steps of our processing pipeline require adaptation to the specific scenario, the last two steps described in Sections 2.3 and 2.4 are similar for all scenarios. We provide a Python implementation [5] of these steps, making use of the mentioned packages. The implementation does not require GPU acceleration and runs reasonably fast on standard hardware.

3. CASE STUDIES

In this section, we present four case studies to illustrate how this processing pipeline may be used within the MIR field. With these case studies, we aim to demonstrate the capability of this approach in assisting researchers across diverse scenarios.

3.1 Measure Numbers

In Western classical music, measure numbers serve as essential temporal markers within musical scores, guiding

[5] `https://github.com/groupmm/textalignsynth`

ISBN 978-3-200-10642-0

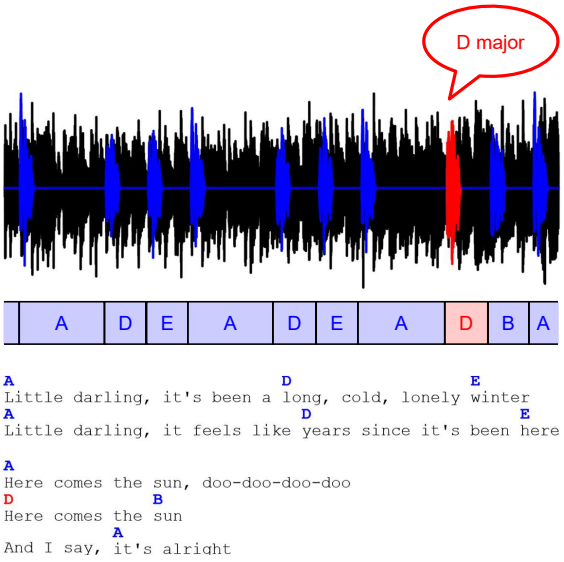

Figure 4. Illustration of synthesizing chord annotations with an excerpt of the song "*Here Comes the Sun*" by The Beatles. Chords may be extracted from the chord chart or by using automated methods [24, 25].

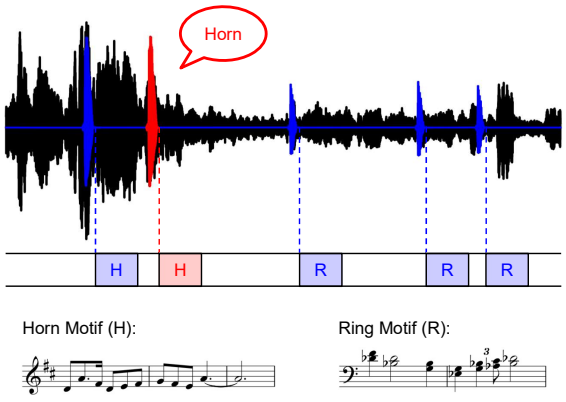

Figure 5. Illustration of announcing leitmotif occurrences within an excerpt of Wagner's "*Götterdämmerung*". Occurrences of Horn and Ring motif are annotated in the orchestral score and transferred to the audio recording.

both musicians and researchers through the structure of a piece. Usually, these numbers are placed at the beginning of every staff system or at regular intervals. Our pipeline offers a novel approach to highlight measure numbers within a recording by announcing them acoustically. This scenario is illustrated by Figure 1, where measure number comments are synthesized for a song from the *Schubert Winterreise Dataset* [1]. The measure position annotations contained in this dataset were created manually for two of the nine versions per song and were automatically transferred to the remaining seven versions using an audio-audio synchronization technique [21]. Alternatively, measure positions may be directly extracted from the music recording via downbeat tracking [22, 23], eliminating the need to perform temporal alignment. While measure positions could also be sonified using simple clicks, our TTS-based approach is advantageous for navigating within a piece: When starting to listen to a recording in the middle of the piece, a person trying to follow the score can easily identify the current measure number from a commented recording, but from a recording that only contains clicks at the measure positions, it is hardly possible. For MIR researchers, this approach may represent a useful option for checking measure number annotations.

For this scenario, we find it beneficial to increase the speed of the TTS output, or to even specify a maximum duration per comment in order to obtain comments which convey temporally accurate information about the downbeat position. For the same reason, we opt for comments that start at the given time stamps, i. e., setting `pos_rel` $= 0.0$.

3.2 Chords and Harmonies

In pop music, chord symbols play a central role in defining the harmonic progression of a song. Unlike classical music, pop songs are mainly described by a chord chart, which contains lyrics and chords. Our pipeline makes chord and harmony analysis easier by announcing chord changes extracted from the chord chart at the right positions within the recording. This scenario is illustrated in Figure 4 with an excerpt of the song "*Here Comes the Sun*" by The Beatles. Beyond the auditory display of annotations, this approach may also be used to examine the predictions made by automatic chord recognition systems [24, 25].

Regarding the positioning, there may be several reasonable options depending on the needs of the user. If the goal is to play along, the chord change needs to be announced before it takes place so that the user can prepare. On the other hand, a person who examines annotation quality may prefer comments that start at the exact time of chord change, providing the user with the temporally most accurate information about the annotations. For a user who wants to do aural training for chord recognition, a slightly delayed playback of the comments may be a useful option as feedback mechanism and for verification.

3.3 Leitmotifs

Leitmotifs are short musical themes describing a specific character, emotion, place, or item within a piece of music [26]. For instance, leitmotifs are massively used in the opera cycle *Der Ring des Nibelungen* by Richard Wagner. Leitmotif occurrences can be hard to detect for a listener because the motifs may appear in modified versions with regard to, e. g., instrumentation, key, rhythm, or tempo. To obtain the information regarding leitmotif occurrences, either expert knowledge is required to identify them within the orchestral score, or leitmotifs may be automatically detected using deep learning-based methods [27]. Having found all leitmotif occurrences, our idea is then to use our processing pipeline to announce them within the music recording. This scenario is illustrated in Figure 5 with an excerpt of Wagner's "*Götterdämmerung*", where all occurrences of the Horn motif and the Ring motif are announced. As indicated by the dashed lines, we suggest to announce leitmotifs before they occur so that a listener can concen-

ISBN 978-3-200-10642-0

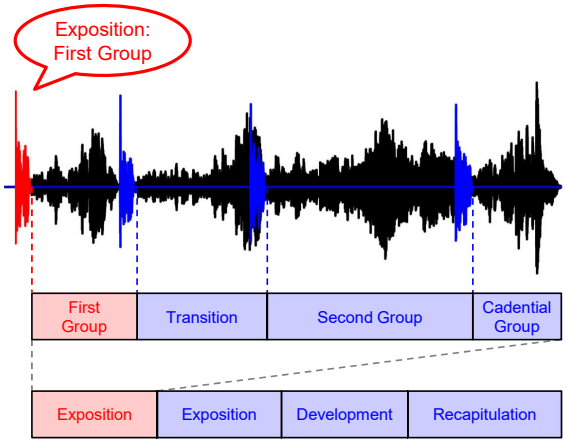

Figure 6. Illustration of synthesizing structure annotations using an excerpt of Beethoven's piano sonata No. 1 F minor, Op. 2 No. 1.

trate on the respective following segments where they actually appear.

3.4 Structure

Analyzing the structure of a piece of music is a central task in musicology. In this case study, we explore the use of auditory display to make the structure of a piece acoustically accessible within a recording. This approach may be applied, e. g., to pop songs with verse–chorus structure, or to classical sonatas adhering to the sonata form consisting of exposition, development, and recapitulation. Such structures can be determined based on the musical score or may be derived from the recording using automated methods [28, 29]. As an example, we use Beethoven's piano sonata No. 1 F minor, Op. 2 No. 1, which exhibits the conventional sonata form. The exposition of this piece can be further divided into first group, transition, second group, and cadential group. The information regarding structure is given as textual information, and we can use our processing pipeline to announce the upcoming section as visualized in Figure 6. As for the leitmotifs in Section 3.3, we choose to position the comments in such a way that they precede the start time of the respective announced section, giving the listener the chance to carefully listen to the beginning of each section.

4. CONCLUSION

With the processing pipeline presented in this paper, we demonstrated that TTS synthesis is a viable option for auditory display, making certain kinds of information related to a music recording accessible while listening to the recording. The key feature of our approach is that the information is conveyed via the same acoustic modality as the recording itself. As a consequence, there is no need for a listener to keep track of that information within the visual domain. This is beneficial for situations where it is difficult for a user to temporally align information from both domains, or for situations where information from the visual domain can not be observed by the user.

For researchers in the MIR field, the use of this pipeline allows for gaining a deeper understanding of datasets together with their annotations. Compared to visual approaches, this acoustic approach of making annotations accessible may facilitate the assessment of annotation quality for certain scenarios as demonstrated by our case studies. With customization options regarding language, comment duration, positioning, and loudness, this processing pipeline represents a versatile asset for MIR researchers, which allows for a novel way to interact with musical data.

In future work, we want to explore how our approach may serve educational purposes. The case studies presented in this paper suggest various scenarios: For orchestral musicians, measure number comments may assist in synchronizing sheet music with recordings. An orchestral musician's sheet typically contains only their own part, and it may be difficult to focus the hearing on that part within a recording that contains all parts. Synthesized measure number annotations restore the synchronization for every measure. Another scenario could be a guitarist trying to play along with a song. If the chords are announced shortly before they are to be played, the guitarist can prepare for the change and then play along without having to read the chord chart. Furthermore, our approach may be used for aural training, e. g., chord recognition or interval estimation, where the ground truth can be given to the user shortly after the query. To assess the practical usability of our approach and determine optimal parameter settings, future work will also involve conducting user studies.

Beyond providing the listener with information about the piece of music itself, this processing pipeline can also be used to synthesize various types of text comments related to the music. This includes, e. g., instructions for music students given by the teacher or choreographic instructions for dancers associated with specific positions within the music recording.

Although we present this approach for the music domain, it should be noted that it can be used for any kind of audio signals with corresponding text annotations, including speech signals, environmental sounds, or biomedical signals—to highlight certain events and to guide the listener.

Acknowledgments

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Grant No. 500643750 (MU 2686/15-1). The authors are with the International Audio Laboratories Erlangen, a joint institution of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and Fraunhofer Institute for Integrated Circuits IIS.

5. REFERENCES

[1] C. Weiß, F. Zalkow, V. Arifi-Müller, M. Müller, H. V. Koops, A. Volk, and H. Grohganz, "Schubert Winterreise dataset: A multimodal scenario for music analysis," *ACM Journal on Computing and Cultural Heritage (JOCCH)*, vol. 14, no. 2, pp. 25:1–18, 2021.

ISBN 978-3-200-10642-0

[2] F. Kurth, D. Damm, C. Fremerey, M. Müller, and M. Clausen, "A framework for managing multimodal digitized music collections," in *Proceedings of the European Conference on Research and Advanced Technology for Digital Libraries (ECDL)*, ser. Lecture Notes In Computer Science, vol. 5173. Aarhus, Denmark: Springer, Sep. 2008, pp. 334–345.

[3] T. Hermann, A. Hunt, , and J. G. Neuhoff, *The Sonification Handbook*. Logos Verlag, 2011.

[4] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "Librosa: Audio and music signal analysis in Python," in *Proceedings the Python Science Conference*, Austin, Texas, USA, 2015, pp. 18–25.

[5] Y. Özer, L. Brütting, S. Schwär, and M. Müller, "libsoni: A Python toolbox for sonifying music annotations and feature representations," *Journal of Open Source Software (JOSS)*, vol. 9, no. 96, pp. 06 524:1–6, 2024.

[6] M. Müller, *Fundamentals of Music Processing – Using Python and Jupyter Notebooks*, 2nd ed. Springer Verlag, 2021.

[7] S. Dixon and G. Widmer, "MATCH: A music alignment tool chest," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, London, UK, 2005, pp. 492–497.

[8] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R.-S. Ryan, R. A. Saurous, Y. Agiomyrgiannakis, and Y. Wu, "Natural TTS synthesis by conditioning WaveNet on MEL spectrogram predictions," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Calgary, Canada, 2018, pp. 4779–4783.

[9] J. Kong, J. Kim, and J. Bae, "Hifi-GAN: Generative adversarial networks for efficient and high fidelity speech synthesis," in *Advances in Neural Information Processing Systems (NeurIPS)*, Virtual, 2020.

[10] W. Jang, D. Lim, J. Yoon, B. Kim, and J. Kim, "Univnet: A neural vocoder with multi-resolution spectrogram discriminators for high-fidelity waveform generation," in *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech)*, Brno, Czech Republic, 2021, pp. 2207–2211.

[11] S. Ewert, M. Müller, and P. Grosche, "High resolution audio synchronization using chroma onset features," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Taipei, Taiwan, 2009, pp. 1869–1872.

[12] M. Müller, Y. Özer, M. Krause, T. Prätzlich, and J. Driedger, "Sync Toolbox: A Python package for efficient, robust, and accurate music synchronization," *Journal of Open Source Software (JOSS)*, vol. 6, no. 64, pp. 3434:1–4, 2021.

[13] I. Bukey, J. Zhang, and T. Tsai, "FlexDTW: Dynamic time warping with flexible boundary conditions," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Milano, Italy, 2023.

[14] J. Driedger and M. Müller, "A review on time-scale modification of music signals," *Applied Sciences*, vol. 6, no. 2, pp. 57–82, February 2016.

[15] S. Grofit and Y. Lavner, "Time-scale modification of audio signals using enhanced WSOLA with management of transients," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 1, pp. 106–115, 2008.

[16] J. Driedger and M. Müller, "TSM Toolbox: MATLAB implementations of time-scale modification algorithms," in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, Erlangen, Germany, 2014, pp. 249–256.

[17] J. Driedger, M. Müller, and S. Ewert, "Improving time-scale modification of music signals using harmonic–percussive separation," *IEEE Signal Processing Letters*, vol. 21, no. 1, pp. 105–109, 2014.

[18] S. S. Stevens, "The measurement of loudness," *The Journal of the Acoustical Society of America (JASA)*, vol. 27, no. 5, pp. 815–829, 1955.

[19] International Telecommunications Union, *ITU-R BS.1770-4: Algorithms to measure audio programme loudness and true-peak audio level*, Std., 2015.

[20] C. J. Steinmetz and J. D. Reiss, "pyloudnorm: A simple yet flexible loudness meter in python," in *Audio Engineering Society Convention 150*, Online, May 2021.

[21] F. Zalkow, C. Weiß, T. Prätzlich, V. Arifi-Müller, and M. Müller, "A multi-version approach for transferring measure annotations between music recordings," in *Proceedings of the AES International Conference on Semantic Audio*, Erlangen, Germany, 2017, pp. 148–155.

[22] S. Böck, F. Krebs, and G. Widmer, "Joint beat and downbeat tracking with recurrent neural networks," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, New York City, New York, USA, 2016, pp. 255–261.

[23] B. Jia, J. Lv, and D. Liu, "Deep learning-based automatic downbeat tracking: a brief review," *Multimedia Systems*, vol. 25, no. 6, pp. 617–638, 2019.

[24] T. Fujishima, "Realtime chord recognition of musical sound: A system using common lisp music," in *Proceedings of the International Computer Music Conference (ICMC)*, Beijing, China, 1999, pp. 464–467.

ISBN 978-3-200-10642-0

[25] J. Pauwels, K. O'Hanlon, E. Gómez, and M. B. Sandler, "20 years of automatic chord recognition from audio," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, 2019, pp. 54–63.

[26] M. Bribitzer-Stull, *Understanding the Leitmotif*. Cambridge University Press, 2015.

[27] M. Krause, M. Müller, and C. Weiß, "Towards leitmotif activity detection in opera recordings," *Transactions of the International Society for Music Information Retrieval (TISMIR)*, vol. 4, no. 1, pp. 127–140, 2021.

[28] R. B. Dannenberg and M. Goto, "Music structure analysis from acoustic signals," in *Handbook of Signal Processing in Acoustics*, D. Havelock, S. Kuwano, and M. Vorländer, Eds. New York, NY, USA: Springer, 2008, vol. 1, pp. 305–331.

[29] J. Paulus, M. Müller, and A. Klapuri, "Audio-based music structure analysis," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Utrecht, The Netherlands, 2010, pp. 625–636.

ISBN 978-3-200-10642-0