libsoni: A Python Toolbox for Sonifying Music Annotations and Feature Representations

Yigitcan Özer ⓘ¹, **Leo Brütting**¹, **Simon Schwär** ⓘ¹, **and Meinard Müller** ⓘ¹

1 International Audio Laboratories Erlangen

Summary

Music Information Retrieval (MIR) stands as a dedicated research field focused on advancing methodologies and tools for organizing, analyzing, retrieving, and generating data related to music. Key tasks within MIR include beat tracking, structural analysis, chord recognition, melody extraction, and source separation, just to name a few. These tasks involve extracting musically relevant information from audio recordings, typically accomplished by transforming music signals into feature representations such as spectrograms, chromagrams, or tempograms (Müller, 2015). Furthermore, musically relevant annotations such as beats, chords, keys, or structure boundaries become indispensable for training and evaluating MIR approaches.

When evaluating and enhancing MIR systems, it is crucial to thoroughly examine the properties of feature representations and annotations to gain a deeper understanding of algorithmic behavior and the underlying data. In the musical context, alongside conventional data visualization techniques, data sonification techniques are emerging as a promising avenue for providing auditory feedback on extracted features or annotated information. This is particularly advantageous given the finely tuned human perception to subtle variations in frequency and timing within the musical domain.

This paper introduces *libsoni*, an open-source Python toolbox tailored for the sonification of music annotations and feature representations. By employing explicit and easy-to-understand sound synthesis techniques, libsoni offers functionalities for generating and triggering sound events, enabling the sonification of spectral, harmonic, tonal, melodic, and rhythmic aspects. Unlike existing software libraries focused on creative applications of sound generation, libsoni is designed to meet the specific needs of MIR researchers and educators. It aims to simplify the process of music exploration, promoting a more intuitive and efficient approach to data analysis by enabling users to interact with their data in acoustically meaningful ways. As a result, libsoni not only improves the analytical capabilities of music scientists but also opens up new avenues for innovative music analysis and discovery. Furthermore, libsoni provides well-documented and stand-alone functions covering all essential building blocks crucial for both sound generation and sonification, enabling users to efficiently apply and easily extend the methods. Additionally, the toolbox includes educational Jupyter notebooks with illustrative code examples demonstrating the application of sonification and visualization methods to deepen understanding within specific MIR scenarios.

Statement of Need

Music data, characterized by attributes such as pitch, melody, harmony, rhythm, structure, and timbre, is inherently intricate. Visualizations are crucial in deciphering this complexity by presenting music representations graphically, enabling researchers to identify patterns, trends, and relationships not immediately evident in raw data. For instance, visualizing time-dependent two-dimensional feature representations such as spectrograms (time–frequency),

chromagrams (time–chroma), or tempograms (time–tempo) enhances comprehension of signal processing concepts and reveals insights into the musical and acoustic properties of audio signals. Moreover, the combined visualization of extracted features and reference annotations facilitates detailed examination of algorithmic approaches at a granular level. These qualitative assessments, alongside quantitative metrics, are essential for comprehending the strengths, weaknesses, and assumptions underlying music processing algorithms. The MIR community has developed numerous toolboxes, such as essentia (Bogdanov et al., 2013), madmom (Böck et al., 2016), Chroma Toolbox (Müller & Ewert, 2011), Tempogram Toolbox (Grosche & Müller, 2011), Sync Toolbox (Müller et al., 2021), Marsyas (Tzanetakis, 2009), or the MIRtoolbox (Lartillot & Toiviainen, 2007), offering modular code for music signal processing and analysis, many of which also include data visualization methods. Notably, the two Python packages librosa (McFee et al., 2015) and libfmp (Müller & Zalkow, 2021) aim to lower the barrier to entry for MIR research by providing accessible code alongside visualization functions, bridging the gap between education and research.

As an alternative or addition to visualizing data, one can employ data sonification techniques to produce acoustic feedback on extracted or annotated information (Kramer et al., 1999). This is especially important in music, where humans excel at detecting even minor variations in the frequency and timing of sound events. For instance, people can readily perceive irregularities and subtle changes in rhythm when they listen to a pulse track converted into a sequence of click sounds. This ability is particularly valuable for MIR tasks such as beat tracking and rhythm analysis. Moreover, transforming frequency trajectories into sound using sinusoidal models can offer insights for tasks like estimating melody or separating singing voices. Furthermore, an auditory representation of a chromagram provides listeners with an understanding of the harmony-related tonal information contained in an audio signal. Therefore, by converting data into sound, sonification can reveal subtle audible details in music that may not be immediately apparent within visual representations.

In the MIR context, sonification methods have been employed to provide deeper insights into various music annotations and feature representations. For instance, the Python package librosa (McFee et al., 2015) offers a function `librosa.clicks` that generates an audio signal with click sounds positioned at specified times, with options to adjust the frequency and duration of the clicks. Additionally, the Python toolbox libf0 (Rosenzweig et al., 2022) provides a function (`libf0.utils.sonify_trajectory_with_sinusoid`) for sonifying F0 trajectories using sinusoids. Moreover, the Python package libfmp (Müller & Zalkow, 2021) includes a function (`libfmp.b.sonify_chromagram_with_signal`) for sonifying time–chroma representations. Testing these methods, our experiments have revealed that current implementations sometimes rely on inefficient event-based looping, which may result in excessively long runtimes.

In our Python toolbox, libsoni, we offer implementations of various sonification methods, including those mentioned above. These implementations feature a coherent API and are based on straightforward methods that are transparent and easy to understand (ensuring a trade-off between efficiency and code readability). Additionally, libsoni includes all essential components for sound synthesis, operating as a standalone tool that can be easily extended and customized. The methods in libsoni enable interactivity, allowing for data manipulation and sonification, as well as the ability to alter feature extraction or sonification techniques. While real-time capabilities are not currently included in libsoni, this could be a potential future extension. Hence, libsoni may not only be beneficial for MIR researchers but also for educators, students, composers, sound designers, and individuals exploring new musical concepts.

Core Functionalities

In the following, we briefly describe some of the main modules included in the Python toolbox libsoni. For an illustration of some core functionalities, we also refer to Figure 1. A compre-

hensive API documentation of libsoni is publicly accessible through GitHub[1]. Furthermore, the applications of core functionalities are illustrated by educational Jupyter notebooks as an integral part of libsoni, providing illustrative code examples within concrete MIR scenarios.
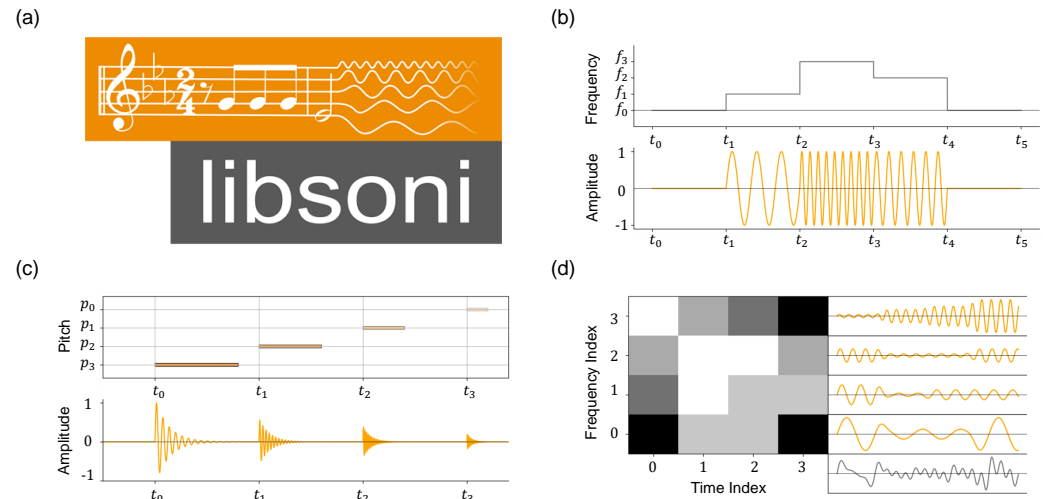

Figure 1: Illustration of some core functionalities provided by the Python toolbox libsoni. **(a)** Logo of libsoni. **(b)** Sonification of F0 trajectories. **(c)** Sonification of piano roll representations. **(d)** Sonification of time–frequency representations.

Triggered Sound Events (`libsoni.tse`)

The Triggered Sound Events (TSE) module of libsoni contains various functions for the sonification of temporal events. In this scenario, one typically has a music recording and a list of time positions that indicate the presence of certain musical events. These events could include onset positions of specific notes, beat positions, or structural boundaries between musical parts. The TSE module allows for generating succinct acoustic stimuli at each of the time positions, providing the listener with precise temporal feedback. Ideally, these stimuli should be perceivable even when overlaid with the original music recording. Often, the time positions are further classified into different categories (e.g., downbeat and upbeat positions). To facilitate this classification, similar to librosa (McFee et al., 2015), the TSE module allows for generating distinguishable stimuli with different "colorations" that can be easily associated with the different categories. Additionally, the TSE module enables the playback of pre-recorded stimuli at different relative time positions and, if specified by suitable parameter settings, with time–scale modifications and pitch shifting.

Fundamental Frequency (`libsoni.f0`)

When describing a specific song, we often have the ability to sing or hum the main melody, which can be loosely defined as a linear succession of musical tones expressing a particular musical idea. In the context of a music recording (rather than a musical score), the melody corresponds to a sequence of fundamental frequency values (also called F0 values) representing the pitches of the tones. In real performances, these sequences often form complex time–frequency patterns known as frequency trajectories, which may include continuous frequency glides (glissando) or frequency modulations (vibrato). In libsoni, the F0 module allows for sonifying a sequence of frame-wise frequency values that correspond to manually annotated or estimated F0 values (see also Figure 1b). This module offers a variety of adjustable parameters, allowing for the inclusion of additional partials to tonally enrich the sonification, thereby generating sounds of

---

[1] https://groupmm.github.io/libsoni

different timbre. Moreover, users have the option to adjust the amplitude of each predicted F0 value based on its confidence level, as provided by an F0 estimator. This allows for insights into the reliability of the predictions.

Piano-Roll Representations (`libsoni.pianoroll`)

A symbolic score-based representation describes each note by parameters such as start time, duration, pitch, and other attributes. This representation is closely related to MIDI encodings and is often visualized in the form of two-dimensional piano-roll representations (see also Figure 1c). In these representations, time is encoded on the horizontal axis, pitch on the vertical axis, and each note is represented by an axis-parallel rectangle indicating onset, pitch, and duration. This representation is widely used in several MIR tasks, including automatic music transcription (Benetos et al., 2019) and music score–audio music synchronization (Müller, 2015). The simplest method in libsoni to sonify piano-roll representations is based on straightforward sinusoidal models (potentially enriched by harmonics). When the score information is synchronized with a music recording (e.g., using alignment methods provided by the Sync Toolbox, Müller et al., 2021), libsoni enables the creation of a stereo signal with the sonification in one channel and the original recording in the other channel. This setup provides an intuitive way to understand the accuracy for a range of musical analysis and transcription tasks. Furthermore, these sonifications may be superimposed with further onset-based stimuli provided by the TSE module.

Chromagram Representations (`libsoni.chroma`)

Humans perceive pitch in a periodic manner, meaning that pitches separated by an octave are perceived as having a similar quality or acoustic color, known as chroma. This concept motivates the use of time–chroma representations or chromagrams, where pitch bands that differ spectrally by one or several octaves are combined to form a single chroma band (Müller & Ewert, 2011). These representations capture tonal information related to harmony and melody while exhibiting a high degree of invariance with respect to timbre and instrumentation. Chromagrams are widely used in MIR research for various tasks, including chord recognition and structure analysis. The Chroma module of libsoni provides sonification methods for chromagrams based on Shepard tones. These tones are weighted combinations of sinusoids separated by octaves and serve as acoustic counterparts to chroma values. The functions offered by libsoni enable the generation of various Shepard tone variants and can be applied to symbolic representations (such as piano roll representations or chord annotations) or to chroma features extracted from music recordings. This facilitates deeper insights for listeners into chord recognition results or the harmony-related tonal information contained within an audio signal.

Spectrogram Representations (`libsoni.spectrogram`)

Similar to chromagrams, pitch-based feature representations can be derived directly from music recordings using transforms such as the constant-Q-transform (CQT), see (Schörkhuber & Klapuri, 2010). These representations are a special type of log-frequency spectrograms, where the frequency axis is logarithmically spaced to form a pitch-based axis. More generally, in audio signal processing, there exists a multitude of different time–frequency representations. For example, classic spectrograms have a linear frequency axis, usually computed via the short-time Fourier transform (STFT). Additionally, mel-frequency spectrograms utilize the mel scale, which approximates the human auditory system's response to different frequencies. The Spectrogram module of libsoni is designed to sonify various types of spectrograms with frequency axes spaced according to linear, logarithmic, or mel scales. Essentially, each point on the scale corresponds to a specific center frequency, meaning that each row of the spectrogram represents the energy profile of a specific frequency over time. Our sonification approach generates sinusoids for each center frequency value with time-varying amplitude values, in accordance with the

provided energy profiles, and then superimposes all these sinusoids. Transforming spectrogram-like representations into an auditory experience, our sonification approach allows for a more intuitive understanding of the frequency and energy characteristics within a given music recording. Finally, we would like to emphasize that sonifying each time–frequency bin is computationally expensive. To enhance efficiency, we integrated additional functions that employs multiprocessing, alongside a simpler function that uses a for loop.

Design Choices

When designing the Python toolbox libsoni, we had several objectives in mind. Firstly, we aimed to maintain close connections with existing sonification methods provided in McFee et al. (2015) and libfmp (Müller & Zalkow, 2021). Secondly, we re-implemented and included all necessary components (e.g., sound generators based on sinusoidal models and click sounds), even though similar basic functionality is available in librosa and libfmp. By doing so, libsoni offers a coherent API along with convenient but easily modifiable parameter presets. Thirdly, we adopted many design principles suggested by librosa (McFee et al., 2015) and detailed in (McFee et al., 2019) to lower the entry barrier for students and researchers who may not be coding experts. This includes maintaining an explicit and straightforward programming style with a flat, functional hierarchy to facilitate ease of use and comprehension. The source code for libsoni, along with comprehensive API documentation[2], is publicly accessible through a dedicated GitHub repository[3]. We showcase all components, including introductions to MIR scenarios, illustrations, and sound examples via Jupyter notebooks. Finally, we have included the toolbox in the Python Package Index (PyPI), enabling installation with the standard Python package manager, pip[4].

Acknowledgements

References

Benetos, E., Dixon, S., Duan, Z., & Ewert, S. (2019). Automatic music transcription: An overview. *IEEE Signal Processing Magazine*, *36*(1), 20–30. https://doi.org/10.1109/MSP.2018.2869928

Böck, S., Korzeniowski, F., Schlüter, J., Krebs, F., & Widmer, G. (2016). Madmom: A new Python audio and music signal processing library. *Proceedings of the ACM International Conference on Multimedia (ACM-MM)*, 1174–1178. https://doi.org/10.1145/2964284.2973795

Bogdanov, D., Wack, N., Gómez, E., Gulati, S., Herrera, P., Mayor, O., Roma, G., Salamon, J., Zapata, J. R., & Serra, X. (2013). Essentia: An audio analysis library for music information retrieval. *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 493–498. https://doi.org/10.5281/zenodo.1415016

[2] https://groupmm.github.io/libsoni
[3] https://github.com/groupmm/libsoni
[4] https://pypi.org/project/libsoni

Grosche, P., & Müller, M. (2011). Tempogram Toolbox: MATLAB tempo and pulse analysis of music recordings. *Demos and Late Breaking News of the International Society for Music Information Retrieval Conference (ISMIR)*.

Kramer, G., Walker, B., Bonebright, T., Cook, P., Flowers, J. H., Miner, N., & Neuhoff, J. (1999). *Sonification report: Status of the field and research agenda*. International Community for Auditory Display.

Lartillot, O., & Toiviainen, P. (2007). MIR in MATLAB (II): A toolbox for musical feature extraction from audio. *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 127–130. https://doi.org/10.5281/zenodo.1417145

McFee, B., Kim, J. W., Cartwright, M., Salamon, J., Bittner, R. M., & Bello, J. P. (2019). Open-source practices for music signal processing research: Recommendations for transparent, sustainable, and reproducible audio research. *IEEE Signal Processing Magazine*, *36*(1), 128–137. https://doi.org/10.1109/MSP.2018.2875349

McFee, B., Raffel, C., Liang, D., Ellis, D. P. W., McVicar, M., Battenberg, E., & Nieto, O. (2015). Librosa: Audio and music signal analysis in Python. *Proceedings the Python Science Conference*, 18–25. https://doi.org/10.25080/Majora-7b98e3ed-003

Müller, M. (2015). *Fundamentals of music processing – audio, analysis, algorithms, applications* (pp. 1–480) [Monograph]. Springer Verlag. https://doi.org/10.1007/978-3-319-21945-5

Müller, M., & Ewert, S. (2011). Chroma Toolbox: MATLAB implementations for extracting variants of chroma-based audio features. *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 215–220. https://doi.org/10.5281/zenodo.1416032

Müller, M., Özer, Y., Krause, M., Prätzlich, T., & Driedger, J. (2021). Sync Toolbox: A Python package for efficient, robust, and accurate music synchronization. *Journal of Open Source Software (JOSS)*, *6*(64), 3434:1–4. https://doi.org/10.21105/joss.03434

Müller, M., & Zalkow, F. (2021). libfmp: A Python package for fundamentals of music processing. *Journal of Open Source Software (JOSS)*, *6*(63), 3326:1–5. https://doi.org/10.21105/joss.03326

Rosenzweig, S., Schwär, S., & Müller, M. (2022). libf0: A Python library for fundamental frequency estimation. *Demos and Late Breaking News of the International Society for Music Information Retrieval Conference (ISMIR)*. https://doi.org/10.5281/zenodo.7512227

Schörkhuber, C., & Klapuri, A. P. (2010). Constant-Q transform toolbox for music processing. *Proceedings of the Sound and Music Computing Conference (SMC)*. https://doi.org/10.5281/zenodo.849741

Tzanetakis, G. (2009). Music analysis, retrieval and synthesis of audio signals MARSYAS. *Proceedings of the ACM International Conference on Multimedia (ACM-MM)*, 931–932. https://doi.org/10.1145/1631272.1631459