pyAMPACT: A Score-Audio Alignment Toolkit for Performance Data Estimation and Multi-modal Processing

Johanna Devaney
Brooklyn College, CUNY
jcdevaney@gmail.com

Daniel McKemie
Brooklyn College, CUNY
daniel.mckemie@gmail.com

Alexander Morgan
Independent
alexanderpmorgan@gmail.com

ABSTRACT

pyAMPACT (Python-based Automatic Music Performance Analysis and Comparison Toolkit) links symbolic and audio music representations to facilitate score-informed estimation of performance data from audio. It can read a range of symbolic formats and can output note-linked audio descriptors/performance data into MEI-formatted files. pyAMPACT uses score alignment to calculate time-frequency regions of importance for each note in the symbolic representation from which it estimates a range of parameters from the corresponding audio. These include frame-wise and note-level tuning-, dynamics-, and timbre-related performance descriptors, with timing-related information available from the score alignment. Beyond performance data estimation, pyAMPACT also facilitates multi-modal investigations through its infrastructure for linking symbolic representations and annotations to audio.

1. INTRODUCTION

pyAMPACT uses score-audio alignment to facilitate expressive performance modeling. Although this approach requires a score, it provides more robust estimates of onsets and offsets than automatic transcription methods. Although there have recently been significant improvements in the accuracy of automatic transcription, largely due to advances in deep-learning architecture, accuracy on this task is still limited [1, 2]. Symbolic representations also contain additional information that can also be difficult to accurately estimate solely from the audio signal, including meter and note spellings, which is useful for musicological analysis of performance data. Beyond expressive performance, pyAMPACT's tools for linking symbolic and audio representations are useful for multi-modal processing of audio and score-based representations with annotations and other types of human-generated data. pyAMPACT is based on AMPACT, a MATLAB-based toolkit originally released in 2011 [3, 4]. It is not simply a Python-based re-implementation of AMPACT, but rather a complete redesign that connects to and extends the functionality of an array of other open-source music analysis tools, most notably librosa [5] and music21 [6]. It also moves away from the proprietary nature of MATLAB while building on the signal-processing algorithms integrated into AMPACT [7, 8, 9, 10]. pyAMPACT is also able to read a range of annotation encoding formats, including Dezrann [11], Humdrum's analytic spines [12], and CRIM intervals [13]). It can also visualize both score and performance data in Verovio [14].

In this paper, we first contextualize pyAMPACT within related work on score-audio alignment, performance data estimation, and multi-modal processing (Section 2). We then present an overview of pyAMPACT's workflow (Section 3) before detailing its symbolic importing (Section 4), audio processing (Section 5), and symbolic exporting (Section 6) functionality, as well as its documentation and tutorials (Section 7). We conclude with a summary of pyAMPACT and a discussion of future directions (Section 8).

2. RELATED WORK

Estimation of performance parameters from polyphonic audio requires accurate extraction of frequency and power information for each note's fundamental frequency and all of its partials. While deep-learning approaches in the polyphonic note-transcription have improved the state-of-the-art, there remains a performance ceiling for both note event detection and f_0 accuracy [1, 2]. An alternative, which has been explored since the 1990s [15], is to use score-audio alignment to inform the estimation of performance parameters. Standard algorithms estimate a single time point estimate for the start of each notated simultaneity [16, 17] while some algorithms estimate onset and offset for each note in a notated simultaneity [18, 19, 20]. As with most other MIR tasks, deep-learning approaches have been applied to score-audio alignment [21, 22], including work with the goal of improving multi-pitch estimation with weakly aligned scores [23].

Performance parameter estimation is valuable for expressive performance modeling, which traditionally has been done for the purposes of both analysis and generation [24, 25, 26]. Work on expressive performance is still largely focused on the piano [27, 28, 29], although the voice and other instruments are increasingly being studied [30, 31, 32, 33]. Some recent work on performance parameter estimation through score-audio alignment has also examined the use case of detecting errors in performance, both to improve the score-audio alignment accuracy [34, **?**] and to assess performance [35, 36]. Score-audio alignment has also been leveraged for multi-modal processing [37] and producing large-scale multi-modal datasets themselves [38, 39, 40]. This helps to address the data paucity issue facing multi-modal processing [41].

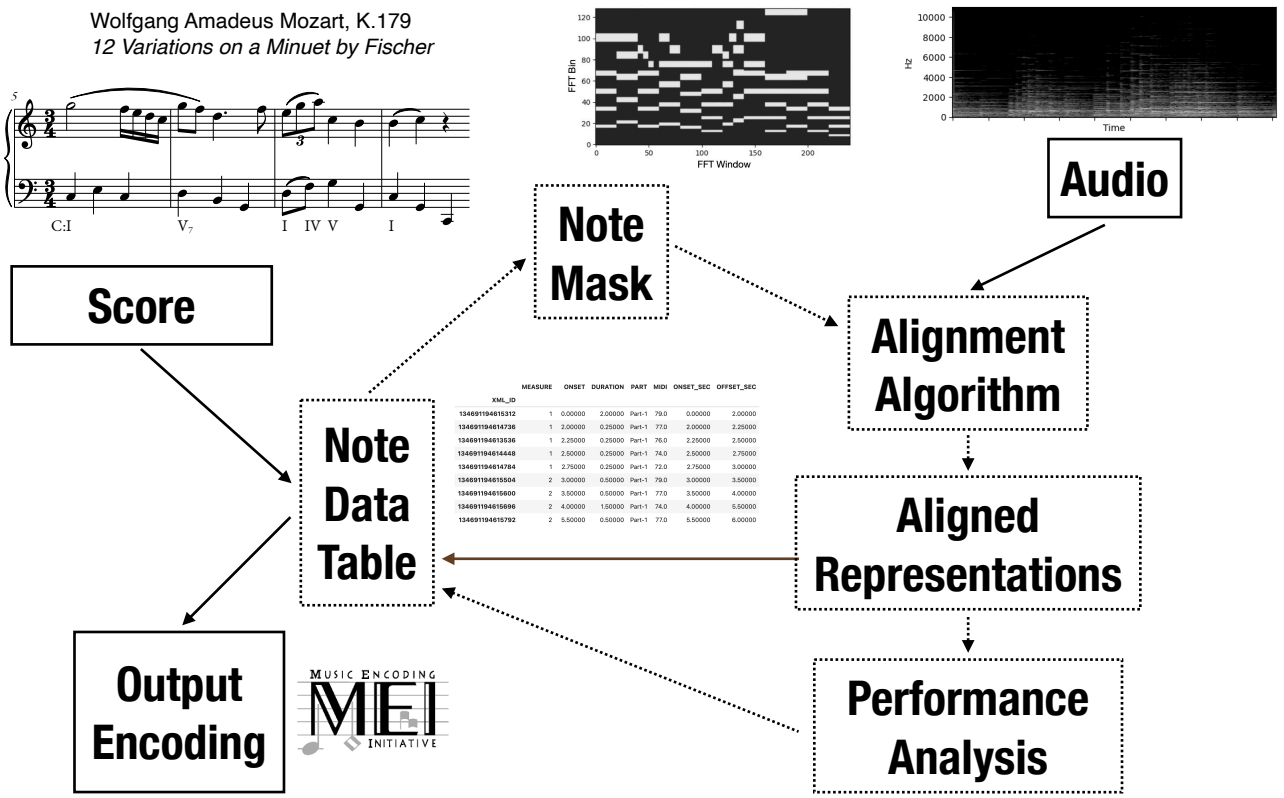

Figure 1. Overview of pyAMPACT's workflow.

3. OVERVIEW OF PYAMPACT'S WORKFLOW

Figure 1 provides an overview of pyAMPACT's workflow. Symbolic data ('Score') along with any time-aligned annotations are imported and stored in a series of Pandas DataFrames (including a 'Note Data Table'). The score-based events in the DataFrames are linked to the original symbolic representation. This allows for the analysis and export of the estimated performance data. The score data stored in the 'Note Data Table' is converted into a spectrogram-like mask ('Note Mask') to facilitate alignment with the imported audio. The 'Alignment Algorithm' calculates a mapping between each note in the symbolic representation and specific time-frequency regions in the audio spectrogram ('Aligned Representations'). The alignment is currently performed by the DTW-based algorithm used in AMPACT, we plan to make this extensible, so other alignment algorithms can be used. The timing information in the 'Note Data Table' is updated with the timing estimates from the alignment stored in the 'Aligned Representations'. The note-wise time-frequency regions estimated through the alignment process are used to estimate tuning-, dynamics-, and timbre-related performance parameters ('Performance Analysis'). The estimated frame- and note-level performance parameters are also stored in the 'Note Data Table', which facilitates the analysis of the performance data in relation to symbolic data and any linked annotations. The performance data in the 'Note Data Table' can either be analyzed within Python or exported into a standard encoding format (i.e., MEI) linked with the symbolic events in the original imported symbolic data ('Output Encoding') for analysis in other coding environments.

4. IMPORTING SYMBOLIC DATA

pyAMPACT imports scores through an exposed music21 score object representation, allowing it to leverage music21 support all major symbolic notation file types (e.g., Humdrum `**kern`, MEI, MIDI, and MusicXML), as well as less commonly used ones (e.g., ABC, TinyNotation, and Volpiano). This is an expansion of file formats from the original AMPACT, which only supports MIDI files.

Symbolic data is accessible through pyAMPACT's Score class, which encodes the following information for each note in the symbolic representation: XML ID (for linking to imported score data), measure number, onset and duration in beats (measured from the beginning of the piece), part number, MIDI note number, and onset and offset times in seconds. The onset and offset times are originally populated with a placeholder value based on 60 bpm. These values are subsequently updated by the alignment algorithm with times from the aligned audio file.

To calculate accurate durations from the imported file, it is first converted to a temporary MIDI file and parsed by track. Each track's messages are processed, with types read for tempo, note-on, note-off, and end-of-track events. If tempo information is defined in the source file, it is used throughout; otherwise, a default of 60 BPM is assigned. Note-on and note-off messages with a velocity greater than 0 (to ignore rests) are captured and used to calculate the start and end times of each note, based on pulses per quarter note (PPQN) [1].

[1] Start time is calculated using `start_time = current_tempo / (1_000_000 * ppqn)`, where the tempo is converted to microseconds to determine the pulses. These calculations are cumulatively

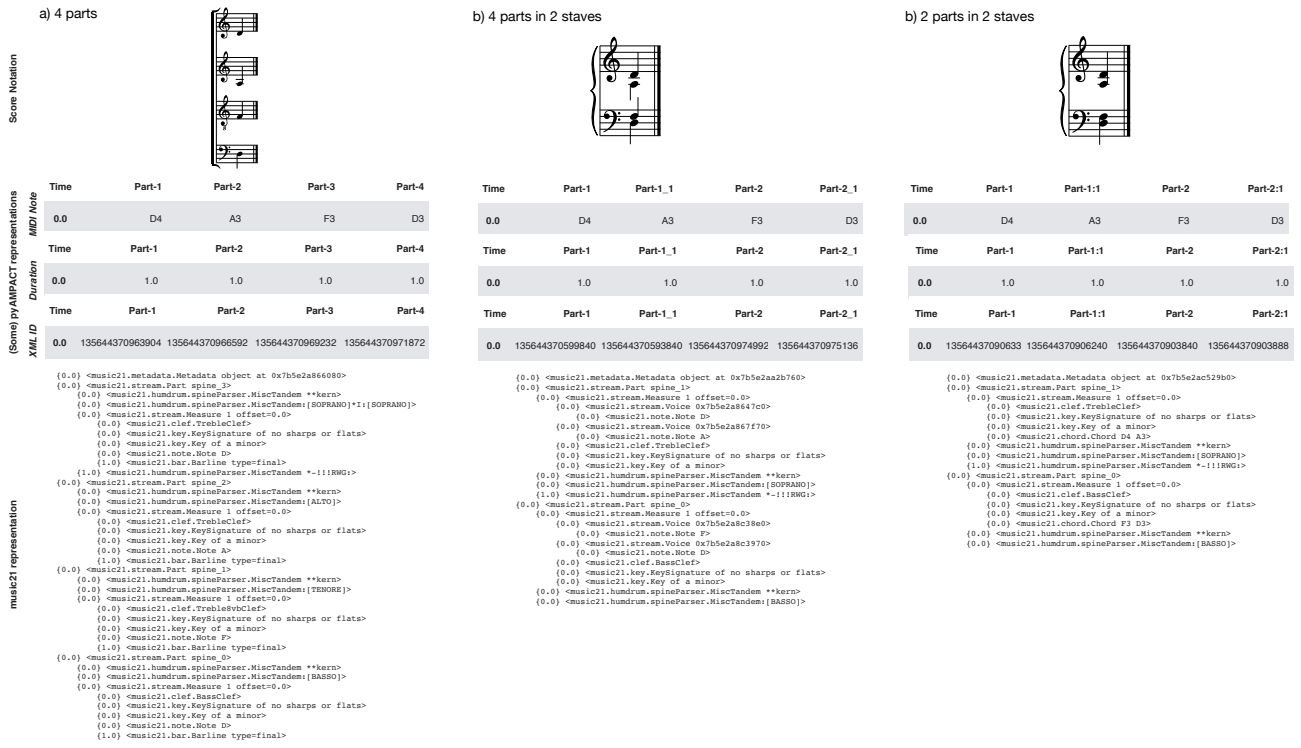

Figure 2. Musical score data (upper row) represented in DataFrames by `pyAMPACT` (middle row) and as trees by `music21`. Subplot (a) shows a chord voiced as 4 separate voices, each in their own staves. Subplot (b) shows the same chord voiced as 4 separate voices, grouped into two staves. Subplot (c) shows the same chord with the notes in a single voice in each of the two staves.

When importing symbolic files, `pyAMPACT` converts `music21`'s stream-based representation of score data in Python lists to a tabular representation. Specifically as `Pandas` DataFrames which allows users to interact with the core `pyAMPACT` symbolic representation in a standardized way. In these DataFrames, the `music21` symbolic note data is simplified to events and timings: score components are represented as columns and score time as rows, with score time measured in `music21` offsets (where an eighth note is 0.5, a quarter note 1, a half note 2, etc.)[2] Separate `nmat` DataFrames are generated for each audio file aligned to a symbolic representation and populated with estimated performance data. Figure 2 shows the compactness of the DataFrame-based representation used in `pyAMPACT` versus the tree-based stream representation used in `music21`, particularly across different voicings of the musical material.

The Score object has several methods which output different representations of the data in the symbolic notation file, including a piano roll, a note data table (`nmat`) and a spectrogram-like mask (verb—mask—). The note data table, or note matrix, (`nmat`) DataFrame is based on the representation in the MIDI Toolbox [42]. The `mask` DataFrame, based on Dan Ellis' `alignmidiwav` code [43], provides a spectrogram-like representation of the symbolic data to align with a spectrogram of an audio recording. In addition to facilitating the audio-score alignment necessary for estimating performance parameters within `pyAMPACT`, the alignment of the score and audio representations generated by `pyAMPACT` is useful for multi-modal processing more generally.

4.1 Multi-modal Processing

The representations shown in Figure 2 are some of the possible time-aligned representations in `pyAMPACT`. The symbolic and spectrogram representations at the top of the figure are produced from the input score and audio representations. The lower part of the figure shows the piano roll (`pianoroll`), Roman numeral, pop chord, and harmonic function representations, a spectrogram-like mask of the symbolic representation (as shown in Figure 1) is also available. This is facilitated by `pyAMPACT`'s support for importing aligned analytic annotations from a range of established formats (including Humdrum and Dezrann).

The Humdrum ecosystem includes numerous established and highly capable analytical methods with results encoded in a variety of spine types. `pyAMPACT` can directly read in several Humdrum spine types (`**harm`, `**chord`, and `**function`) and offers a generalized solution for importing any user-defined spine type found in a Humdrum `**kern` file. This allows users to take a working Humdrum analysis workflow, and import its output into pyAMPACT where it is possible to engage with audio analysis or other symbolic analysis available in pyAMPACT. Analytic annotations from .dez files are also supported. In addition to exploring connections be-

added to determine the `ONSET_SEC` time of each note. Similarly, the note-off messages follow the same process to calculate the end time (`OFFSET_SEC`). Finally, the duration of each note is determined as `DURATION = OFFSET_SEC - ONSET_SEC`.

[2] This basic format, with score parts on the x-axis and time on the y-axis, is similar to Humdrum kern. However, unlike Humdrum kern where notes, measures, and time signatures are all in one place, each cell in a `pyAMPACT` DataFrame typically corresponds to a single type of information.

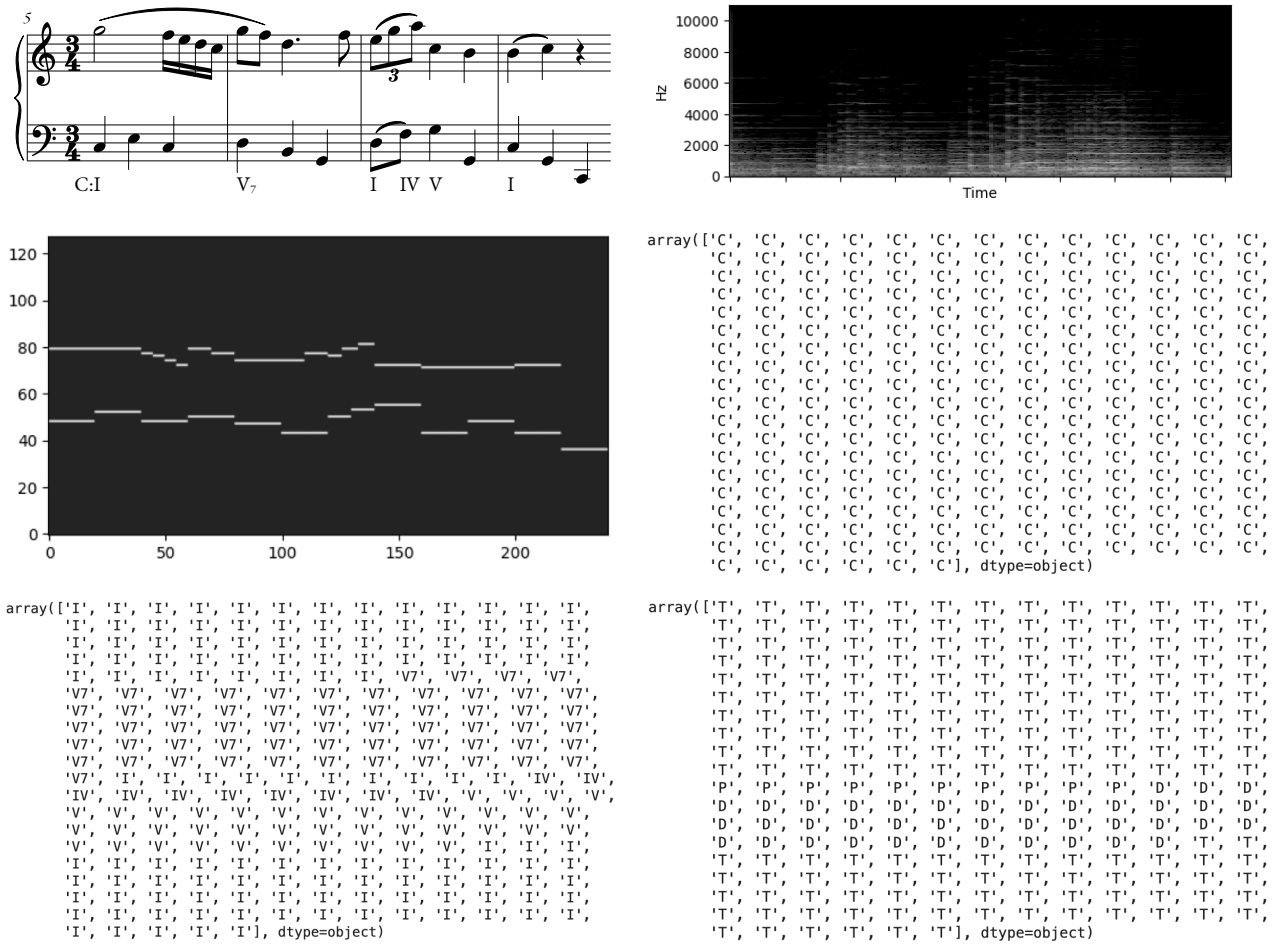

Figure 3. Annotations.

tween analyses from .dez files and corresponding audio observations from pyAMPACT, pyAMPACT users benefit from the ability to use Dezrann's feature-rich and web-based platform as a graphic interface for precisely annotating scores. pyAMPACT.

In addition to the analytic tools in music21, pyAMPACT can also integrate analytic annotations from the Citations: the Renaissance Imitation Mass (CRIM) suite of symbolic analysis. pyAMPACT's data structuring is sufficiently similar to CRIM Intervals that users can directly compare analysis results from a CRIM Intervals method with audio analysis results from pyAMPACT. In the event that an imported score does not have any Roman Numeral annotations, pyAMPACT runs the relevant music21 analysis methods to estimate the Roman Numerals.

5. AUDIO PROCESSING

The imported symbolic data is aligned to a corresponding audio file to estimate note onset and offset times, which replace the initial place-holder onset and offset times in the NMAT representation (described in Section 4). These onset and offset estimates identify time-frequency regions of importance, which facilitates performance parameter estimation in both monophonic and polyphonic audio. This score-guided approach to performance data estimation is implemented in the original AMPACT toolkit and described in [9]. The basic score alignment implemented in pyAMPACT is a standard dynamic time warping (DTW) approach [44], however, pyAMPACT is extensible so that any Python-based score-alignment can be used be used. Once the symbolic data has been aligned to the audio data, a number of frame-level descriptors are estimated per aligned note and also summarized into a set of note-level descriptors. An example of the performance data estimated for a single note is shown in Figure 4.

5.1 Frame-Level Descriptors

pyAMPACT uses an instantaneous frequency representation, produced by librosa's reassigned_spectrogram function, to estimate frame-level fundamental frequency (f_0) and power. A harmonic spectral representation is calculated from the fundamental frequency and power estimates. A set of frame-level descriptors (currently spectral centroid spectral slope, and spectral flux).

5.2 Note-Level Descriptors

From the frame-wise estimates of f_0, power, spectral descriptors described above, pyAMPACT currently estimates five pitch-related note-level descriptors, two dynamics-related note-level descriptors, and four timbre-related note-level descriptors, largely from the frame-wise estimates described above. It is also extensible to estimate addi-

```
<score>
    <section>
        <staff n="1">
            <layer>
                <note
                    xml:id="note_1"
                    oct="4"
                    accid.ges="s"
                    pname="g"/>
            </layer>
        </staff>
    </section>
</score>

<when absolute="00:00:10:087" xml:id="when_1" data="#note_1">
    <extData>
        <![CDATA[>{
        "dur": 0.082721088,

        "f0Vals":[414.45386767783316,414.84188220077607,417.119948592810
        83,418.7548894291117,419.67427527499052,419.88269125537988,419.
        60159061638382,418.73664891143341,418.47482327276629,417.11096
        933439654,414.69842540645334,414.42229415989641],

        "ppitch":419.67413863689859,
        "jitter":0.98992915209365406,
        "vibratoDepth":2.888632876544873,
        "vibratoRate":14.35546875,

        "pwrVals":[8.4166092817353366,9.144024591233709,8.6792749070702
        442,8.1992246427389084,8.0572772724328168,7.8109609776142381,6.
        6733139350701407,3.6537759157163019,-1.0602398380142597,-
        7.234214541488889,-11.225446182086552,-13.86979480892343],

        "avgPwr":7.8353,
        "shimmer":6.8849733724901849,

        "specCent":[1546.7365173078974,1617.2715395815958,1605.075927156
        236,1561.6422826719077,1476.8428858194552,1347.4266112312898,11
        97.3744170739965,1016.1844455901972,951.902680390457,1081.38485
        24751631,1474.80499933333,1565.2662805979317],

        "specCentMean":1370.1594532691213,

        "specSlope":[-1.7768428974010891E-5,-2.0888422076019609E-5,-
        1.8742902905324632E-5,-1.6789512367472517E-5,-
        1.6328428456165875E-5,-1.5571161782510315E-5,-
        1.2104484464312405E-5,-6.1197666655196631E-6,-
        2.0677592030314972E-6,-4.8864724586351032E-7,-
        1.8508729333384716E-7,-1.0118481168516414E-7],

        "meanSpecSlope":-1.0596315520437492E-5,"

        specFlux":[0,0.0005836056244841355 8,0.0006333114129442841 4,0.000
        5412324843509997 4,0.0006038538554486251 4,0.000262794587720547
        04,0.0005071827298047385 7,0.0004535023772644411 5,0.00020384363
        555109953,3.8636166577475942E-5,1.8268535913745484E-
        6,1.3172339563094507E-7],

        "meanspecFlux":0.00031916012092777943,

        "specFlat":[2.0897882834210336E-16,1.583077320977022E-
        16,1.5541642514550739E-16,1.6706134696425619E-
        16,1.6029429339859205E-16,1.6438862448445516E-
        16,2.5946802361150878E-16,7.3519052062822085E-
        16,4.4723439037415433E-15,7.0367629688161888E-
        14,5.4518991543244309E-13,1.7515934200533991E-12],

        "meanspecFlat":1.9780270123936815E-13}
        ]]>
    </extData>
</when>
```

Figure 4. Example of performance data encoded in MEI using the `<extData>` tag. The performance data is linked to the corresponding note in the `<score>` tag through the `data` attribute of the `<when>` tag. The performance data is encoded as a JSON object in a `<![CDATA[>` tag.

tional descriptors estimated by other packages, either from the frame-wise descriptors or the spectral representation. All of the note-level summary descriptors are added in their own columns in the `nmat` DataFrame, which facilitates both analysis of these parameters within Python and exporting the estimated performance data with note-level linking to the corresponding symbolic data in MEI format (see Section 8 for details).

Pitch-Related Descriptors: `pyAMPACT` calculates five pitch-related descriptors: mean f_0, perceived pitch, jitter, vibrato rate, and vibrato depth. Mean f_0 is calculated as a geometric mean, following from the results reported in [45], which indicate that of the simple mean types, the geometric is the closest to the perceived pitch of vibrato tones. A more complex perceived pitch model is also calculated, based on the results reported [46]. This is a weighted mean based on the rate of change, where frames with a slower rate of change in the f_0 estimates are given more weight. Jitter is approximated by calculating the difference between sequential frame-wise f_0 estimates. Vibrato descriptors are calculated by first computing the spectrum of the note-segmented f_0 trace with an FFT. Vibrato extent is estimated by doubling the maximum absolute value of the f_0 trace spectrum and vibrato rate is estimated by finding the position of the maximum absolute value in the spectrum. We are currently working on implementing more sophisticated models of the evolution of pitch and pitch-related parameters over the duration of each note.

Dynamics-Related Descriptors: Mean power is calculated from the note-level frame-wise power estimates using the arithmetic mean. Additionally, shimmer is approximated in an analogous manner to jitter, but by calculating the difference between sequential frame-wise power estimates. Work is currently ongoing to integrate loudness estimation.

Timbre-Related Descriptors: Timbre descriptors are estimated from the harmonic spectral representation derived from the note-wise frame-wise f_0 and power estimates. Currently, `pyAMPACT` calculates all of the spectral features available in librosa (spectral bandwidth, spectral centroid, spectral contrast, spectral flatness, and spectral rolloff) and calculates the arithmetic mean of each of these to generate a note-level summary descriptor. Since the spectral representation only models harmonic partials, the accuracy of these descriptors, particularly spectral flatness, is limited. Work is currently ongoing to incorporate the features in the `Timbre Toolbox` [47], including ADSR, harmonic energy, noisiness, inharmonicity, spectral spread, spectral skewness, spectral kurtosis, and spectral crest.

6. EXPORTING DATA

`pyAMPACT` includes functions for exporting musical scores with note-linked performance data to MEI format. MEI [48] was primarily designed to encode symbolic musical data. The recent inclusion of the `<extData>` [3] element in the latest release of MEI [4] facilitates the linking of symbolic events (such as notes) to data related to specific time points in linked audio files. `<extData>` can contain a standard XML `<![CDATA[]]>` tag, thus the exact specifications of the linked data are flexible. Each `<extData>` element wrapped in a `<when>` element, which requires linking to both a specific time-point

[3] `https://music-encoding.org/guidelines/v5/elements/extData.html`

[4] `https://music-encoding.org/guidelines/v5/content/introduction.html#modelChanges`

in an external representation (in our case an audio file) and linking to a specific symbolic event defined in the ¡score¿ section of the MEI file (linked by XMLID). pyAMPACT encodes a JSON-formatted object [49] with symbolic data into an <extData> element linked to each note in the symbolic representation.

An example of the MEI encoding is shown in Figure 4. Basic MEI data for a single note is shown in the <score> element, which includes the XML ID ('note 1'), which is used to link the performance data in the <when> element. The onset time of the note in the audio file is specified within the <when> stage (absolute="00:00:10:087"). The pitch-, dynamics-, and timbre-related frame-wise and note-level descriptors are encoded within the <extData> element.

7. AVAILABILITY AND DOCUMENTATION

pyAMPACT is available in our GitHub repository [5] as well as as a pip package [6]. Function-level Sphinx documentation of pyAMPACT is also available on a GitHub.io site [7], and a set of Google Colab tutorial notebooks [8] have been developed to help guide users through standard use cases. There are currently three Google Colab tutorial notebooks: (1) an introductory notebook that provides an overview of pyAMPACT's main workflow (corresponding to Section 3); (2) a notebook details how pyAMPACT imports and represents symbolic data (corresponding to Section 4), (3) a notebook related to multi-modal-processing, which details how annotations are imported and represented in pyAMPACT (corresponding to Section 4.1). These tutorials guide users through performing specific tasks with pyAMPACT and provide reusable code that can be integrated into users' own projects. One of our goals as we further develop these tutorials is to make pyAMPACT accessible to musicologists, similar to the way that the Humdrum User Guide [9] and HumdrumR[50] vignettes [10] do.

8. CONCLUSIONS AND FUTURE WORK

pyAMPACT uses score-audio alignment to link symbolic and audio music representations in order to estimate note-wise frame-level and note-level tuning-, dynamics-, and timbre-related performance descriptors. pyAMPACT can read a range of symbolic formats and can output note-linked audio descriptors/performance data into MEI-formatted files. pyAMPACT also facilitates multi-modal investigations through its infrastructure for linking symbolic representations and annotations to audio (as described in [51]).

As mentioned above, we are currently working to integrate pyAMPACT with a loudness estimation package to calculate note-wise loudness estimates and to expand the number of timbral descriptors calculated. To facilitate this, we plan to implement a more sophisticated version of the note-wise spectral representation that leverages phase information in order to capture inharmonic partials. We also plan to implement the DTW-hidden Markov model (HMM)-based model included in AMPACT [19], which estimates individual note onsets and offsets in notated simultaneities. We are also working to support importing under-specified scores, such as annotations from Tony[52]. With under-specified scores, only note onset, duration, and nominal pitch estimates would be required to guide pyAMPACT's audio processing algorithms, which would expand the repertoire that can be analyzed to music without notated scores. We also plan to support the export of linked performance data to more data formats (such as Humdrum) and explore how pyAMPACT can be more directly integrated with other symbolic music data frameworks (such as DIMCAT[53] and musif [54]). And, finally, since pyAMPACT's protocol for importing annotations is extensible and can accommodate a range of note-aligned data, we plan to extend this to offer support for motion capture data and response data from psychological studies.

[5] https://github.com/pyampact/pyampact

[6] https://pypi.org/project/pyampact/

[7] https://pyampact.github.io/

[8] https://github.com/pyampact/pyampacttutorials/

[9] https://www.humdrum.org/guide/

[10] https://humdrumr.ccml.gtcmt.gatech.edu/

9. ACKNOWLEDGMENTS

This work is supported by the National Endowment for the Humanities (NEH) award HAA-281007-21. The opinions expressed in this work are solely those of the authors and do not necessarily reflect the views of the NEH.

10. REFERENCES

[1] E. Benetos, S. Dixon, Z. Duan, and S. Ewert, "Automatic music transcription: An overview," *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 20–30, 2018, publisher: IEEE.

[2] R. M. Bittner, J. J. Bosch, D. Rubinstein, G. Meseguer-Brocal, and S. Ewert, "A lightweight instrument-agnostic model for polyphonic note transcription and multipitch estimation," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 781–785.

[3] J. Devaney, M. I. Mandel, D. P. Ellis, and I. Fujinaga, "Automatically extracting performance data from recordings of trained singers." *Psychomusicology: Music, Mind and Brain*, vol. 21, no. 1-2, p. 108, 2011.

[4] J. Devaney, M. I. Mandel, and I. Fujinaga, "A Study of Intonation in Three-Part Singing using the Automatic Music Performance Analysis and Comparison Toolkit (AMPACT)." in *Proceedings of International Society for Music Information Retrieval (ISMIR) conference*, 2012, pp. 511–516.

[5] 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 of the Python in Science Conference*, vol. 8, 2015, pp. 18–25.

[6] M. S. Cuthbert and C. Ariza, "music21: A toolkit for computer-aided musicology and symbolic music data," in *Proceedings of International Society for Music Information Retrieval (ISMIR) conference*, 2010.

[7] J. Devaney, M. I. Mandel, and D. P. Ellis, "Improving MIDI-audio alignment with acoustic features," in *Proceedings of Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2009, pp. 45–48.

[8] J. Devaney, M. I. Mandel, and I. Fujinaga, "Characterizing singing voice fundamental frequency trajectories," in *Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2011, pp. 73–76.

[9] J. Devaney and M. Mandel, "An evaluation of score-informed methods for estimating fundamental frequency and power from polyphonic audio," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 181–185.

[10] J. Devaney and H. Léveillé Gauvin, "Encoding music performance data in Humdrum and MEI," *International Journal on Digital Libraries*, vol. 20, pp. 81–91, 2019.

[11] M. Giraud, R. Groult, and E. Leguy, "Dezrann, a web framework to share music analysis," in *International Conference on Technologies for Music Notation and Representation (TENOR)*, 2018, pp. 104–110.

[12] D. Huron, "Music information processing using the Humdrum toolkit: Concepts, examples, and lessons," *Computer Music Journal*, vol. 26, no. 2, pp. 11–26, 2002.

[13] R. Freedman, A. Morgan, and D. Russo-Batterham, "Musicologists and Data Scientists Pull out all the Stops: Defining Renaissance Cadences Systematically," in *Proceedings of Music Encoding Conference*, 2022.

[14] L. Pugin, R. Zitellini, and P. Roland, "Verovio: A library for engraving MEI music notation into SVG," in *Proceedings of International Society for Music Information Retrieval (ISMIR) conference*, 2014.

[15] E. Scheirer, "Using musical knowledge to extract expressive performance information from audio recording," *Readings in Computational Auditory Scene Analysis*, 1998.

[16] N. Orio and D. Schwarz, "Alignment of monophonic and polyphonic music to a score," in *International Computer Music Conference (ICMC)*, 2001, pp. 1–1.

[17] C. Raphael, "A Hybrid Graphical Model for Aligning Polyphonic Audio with Musical Scores." in *International Conference on Music Information Retrieval*, 2004, pp. 387–394.

[18] B. Niedermayer and G. Widmer, "A Multi-pass Algorithm for Accurate Audio-to-Score Alignment." in *International Society for Music Information Retrieval Conference*, 2010, pp. 417–422.

[19] J. Devaney, "Estimating onset and offset asynchronies in polyphonic score-audio alignment," *Journal of New Music Research*, vol. 43, no. 3, pp. 266–275, 2014.

[20] J. J. Carabias-Orti, F. J. Rodriguez-Serrano, P. Vera-Candeas, N. Ruiz-Reyes, and F. J. Canadas-Quesada, "An Audio to Score Alignment Framework Using Spectral Factorization and Dynamic Time Warping." in *International Society for Music Information Retrieval Conference*, 2015, pp. 742–748.

[21] F. Simonetta, S. Ntalampiras, and F. Avanzini, "Audio-to-score alignment using deep automatic music transcription," in *2021 IEEE 23rd International Workshop on Multimedia Signal Processing (MMSP)*. IEEE, 2021, pp. 1–6.

[22] R. Agrawal, D. Wolff, and S. Dixon, "A convolutional-attentional neural framework for structure-aware performance-score synchronization," *IEEE Signal Processing Letters*, vol. 29, pp. 344–348, 2021.

[23] M. Krause, C. Weiß, and M. Müller, "Soft dynamic time warping for multi-pitch estimation and beyond," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023, pp. 1–5.

[24] A. Kirke and E. R. Miranda, *Guide to computing for expressive music performance*. Springer Science & Business Media, 2012.

[25] C. E. Cancino-Chacón, M. Grachten, W. Goebl, and G. Widmer, "Computational models of expressive music performance: A comprehensive and critical review," *Frontiers in Digital Humanities*, vol. 5, p. 25, 2018.

[26] A. Lerch, C. Arthur, A. Pati, and S. Gururani, "An Interdisciplinary Review of Music Performance Analysis," *Transactions of the International Society for Music Information Retrieval*, vol. 3, no. 1, pp. 221–246, 2020.

[27] Z. Shi, "Computational analysis and modeling of expressive timing in Chopin's Mazurkas." in *ISMIR*, 2021, pp. 650–656.

[28] F. Simonetta, S. Ntalampiras, and F. Avanzini, "Acoustics-specific Piano Velocity Estimation," in *2022 IEEE 24th International Workshop on Multimedia Signal Processing (MMSP)*. IEEE, 2022, pp. 1–7.

[29] H. Zhang, J. Tang, S. R. Rafee, S. Dixon, G. Fazekas, and G. A. Wiggins, "ATEPP: A Dataset of Automatically Transcribed Expressive Piano Performance," in *International Society for Music Information Retrieval Conference*, 2022.

[30] S. Giraldo, A. Nasarre, I. Heroux, and R. Ramirez, "A Machine Learning Approach to Study Expressive Performance Deviations in Classical Guitar," in *Machine Learning and Knowledge Discovery in Databases: International Workshops of ECML PKDD 2019, Würzburg, Germany, September 16–20, 2019, Proceedings, Part II*. Springer, 2020, pp. 531–536.

[31] Y. Wang, Y. Jing, W. Wei, D. Cazau, O. Adam, and Q. Wang, "PipaSet and TEAS: A Multimodal

Dataset and Annotation Platform for Automatic Music Transcription and Expressive Analysis Dedicated to Chinese Traditional Plucked String Instrument Pipa," *IEEE Access*, vol. 10, pp. 113 850–113 864, 2022, publisher: IEEE.

[32] S. Dai, S. Chen, Y. Wu, R. Diao, R. Huang, and R. B. Dannenberg, "Singstyle111: A multilingual singing dataset with style transfer," in *in Proc. of the 24th Int. Society for Music Information Retrieval Conf*, vol. 1, 2023, pp. 4–2.

[33] N. C. Tamer, Y. Özer, M. Müller, and X. Serra, "High-Resolution Violin Transcription Using Weak Labels," in *in Proc. of the 24th Int. Society for Music Information Retrieval Conf*, 2023.

[34] E. Nakamura, K. Yoshii, and H. Katayose, "Performance Error Detection and Post-Processing for Fast and Accurate Symbolic Music Alignment." in *ISMIR*. Suzhou, 2017, pp. 347–353.

[35] J. Huang and A. Lerch, "Automatic Assessment of Sight-reading Exercises." in *International Society for Music Information Retrieval Conference*, 2019, pp. 581–587.

[36] R. Li and K. Yu, "Regularized DTW in Offline Music Score-Following for Sight-Singing Based on Sol-fa Name Recognition," in *2022 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*. IEEE, 2022, pp. 1–6.

[37] F. Foscarin, A. Mcleod, P. Rigaux, F. Jacquemard, and M. Sakai, "ASAP: a dataset of aligned scores and performances for piano transcription," in *International Society for Music Information Retrieval Conference*, 2020, pp. 534–541.

[38] G. Meseguer-Brocal, A. Cohen-Hadria, and G. Peeters, "Creating DALI, a Large Dataset of Synchronized Audio, Lyrics, and Notes." *Trans. Int. Soc. Music. Inf. Retr.*, vol. 3, no. 1, pp. 55–67, 2020.

[39] F. Simonetta, S. Ntalampiras, F. Avanzini *et al.*, "ASMD: An Automatic Framework for Compiling Multimodal Datasets," in *SMC Sound and Music Computing Conference*. SMC, 2020.

[40] C. Weiß, V. Arifi-Müller, M. Krause, F. Zalkow, S. Klauk, R. Kleinertz, and M. Müller, "Wagner Ring Dataset: A complex opera scenario for music processing and computational musicology," *Transactions of the International Society for Music Information Retrieval*, vol. 6, no. 1, pp. 135–149, 2023.

[41] F. Simonetta, S. Ntalampiras, and F. Avanzini, "Multimodal music information processing and retrieval: Survey and future challenges," in *2019 international workshop on multilayer music representation and processing (MMRP)*. IEEE, 2019, pp. 10–18.

[42] T. Eerola and P. Toiviainen, "MIR In Matlab: The MIDI Toolbox." in *Proceedings of International Society for Music Information Retrieval (ISMIR) conference*, 2004.

[43] D. P. Ellis, "Aligning MIDI files to music audio," https://www.ee.columbia.edu/~dpwe/resources/matlab/alignmidi/, 2013, accessed: 2024-03-10.

[44] N. Orio and D. Schwarz, "Alignment of monophonic and polyphonic music to a score," in *Proceedings of International Computer Music Conference (ICMC)*, 2001.

[45] J. I. Shonle and K. E. Horan, "The pitch of vibrato tones," *The Journal of the Acoustical Society of America*, vol. 67, no. 1, pp. 246–252, 1980.

[46] H. Gockel, B. C. Moore, and R. P. Carlyon, "Influence of rate of change of frequency on the overall pitch of frequency-modulated tones," *The Journal of the Acoustical Society of America*, vol. 109, no. 2, pp. 701–712, 2001.

[47] G. Peeters, B. L. Giordano, P. Susini, N. Misdariis, and S. McAdams, "The timbre toolbox: Extracting audio descriptors from musical signals," *The Journal of the Acoustical Society of America*, vol. 130, no. 5, pp. 2902–2916, 2011.

[48] P. Roland, "The music encoding initiative (MEI)," in *Proceedings of the First International Conference on Musical Applications Using XML*, vol. 1060, 2002, pp. 55–59.

[49] D. Crockford, "The application/JSON media type for javascript object notation (JSON)," Tech. Rep., 2006. [Online]. Available: https://www.rfc-editor.org/rfc/rfc4627

[50] N. Condit-Schultz and C. Arthur, "humdrumR: a New Take on an Old Approach to Computational Musicology." in *ISMIR*, 2019, pp. 715–722.

[51] J. Devaney, "Using note-level music encodings to facilitate interdisciplinary research on human engagement with music," *Transactions of the International Society for Music Information Retrieval*, vol. 3, no. 1, 2020.

[52] M. Mauch, C. Cannam, R. Bittner, G. Fazekas, J. Salamon, J. Dai, J. Bello, and S. Dixon, "Computer-aided melody note transcription using the Tony software: Accuracy and efficiency," in *Proceedings of International Conference on Technologies for Music Notation and Representation*, 2015.

[53] J. Hentschel, A. McLeod, Y. Rammos, M. Rohrmeier, and I. Fraunhofer, "Introducing DiMCAT for processing and analyzing notated music on a very large scale," in *Proceedings of International Society for Music Information Retrieval Conference*, 2023.

[54] A. Llorens, F. Simonetta, M. Serrano, and A. Torrente, "musif: a Python package for symbolic music feature extraction," *arXiv preprint arXiv:2307.01120*, 2023.