JS-XTRACT: A REALTIME AUDIO FEATURE EXTRACTION LIBRARY FOR THE WEB

Nicholas Jillings
Digital Media Technology Lab
Birmingham City University
Birmingham, UK
nicholas.jillings@mail.bcu.ac.uk

Jamie Bullock
Integra Lab
Birmingham City University
Birmingham, UK
jamie.bullock@bcu.ac.uk

Ryan Stables
Digital Media Technology Lab
Birmingham City University
Birmingham, UK
ryan.stables@bcu.ac.uk

ABSTRACT

JS-Xtract is an efficient modular JavaScript library for audio feature extraction, capable of operating on arbitrary time-series data, or being bound to Web Audio objects. The library implements an extensive range of vector and scalar feature extractors, and allows both procedural and object-oriented function calls. We show it performs well across a range of desktop and mobile browsers, and is capable of extracting audio features in realtime.

1. INTRODUCTION

With the introduction of the Web Audio API [1], high resolution audio can now be handled natively in the web browser. This allows for the development of tools such as listening test platforms [5] and online audio processing environments [13], which are developed natively using JavaScript and HTML 5. These tools often require audio feature extraction to perform analysis on time-series data, however libraries that are developed for compiled languages such as LibXtract [2] for C, YAAFE [7] for Python or jAudio [8] for Java can be difficult to deploy due security limitations imposed by the browser. As an alternative, libraries such as Meyda [12] allow for inline Javascript implementation, with a limited feature set, bound to Web Audio API objects. This limits the extent to which non-realtime analysis can be implemented.

Typically these libraries utilise a subset of features described by the CUIDADO [14] and MPEG 7 [6] feature sets, where extensive groups of descriptors are presented. These have been used as a benchmark for inclusion in review papers [9] incorporating temporal, spectrotemporal and abstracted feature representations. Here, we present JS-Xtract, [1] a light-weight JavaScript library for feature extraction, which is agnostic of the data's type or origin.

[1] Library available at http://www.semanticaudio.co.uk/projects/js-xtract/

2. LIBRARY DESIGN

We follow the design philosophy of LibXtract [2] by defining low-level, modular function calls which can be combined to create complex feature extraction graphs. The library is written in JavaScript following the ECMAScript 5 standard for maximum support and functionality. [2] Functions can be passed an arbitrary datasource, providing it meets the dimensionality criteria, and can be bound to Web Audio objects for real-time processing. The library supports both procedural and object-oriented function calls.

2.1 Feature Set

We extend the LibXtract feature set, which includes statistical moments and shape descriptors such as centroid, spread and roll-off, each of which can be extracted from a range of input representations such as a magnitude spectrum, peak- and harmonic-peak spectrum (HPS), MFCCs and Bark coefficients. Input representations can be categorised as temporal, spectrotemporal, and abstracted, where abstracted representations typically model an external system (including MFCCs, Bark or Chroma). Two additional input representations are included, namely chroma features (included in the Chroma Toolbox [3]) which characterise spectral energy distributed across pitch classes, and Equivalent Rectangular Bandwidth (ERB) filters (included in the Timbre Toolbox [11]), which represent auditory filters proposed by Moore and Glasberg [10]. In addition, we incorporate temporal envelope, spectral flux and filter-bank deltas, which were omitted from LibXtract due to the lack of frame-based decomposition. From this, we can extract a range of features from the amplitude envelope, such as Log-attack time and temporal centroid. The library primarily follows the structure in Figure 1, in which most scalar features can be extracted for an arbitrary input representation, including the original time-series frame of audio, with the exception of f_0, loudness, HPS and noisiness, which rely on specific input representations.

2.2 Web Audio API Integration

The Web Audio API [1] provides audio processing functionality for every major mobile and desktop browser. [3]

[2] Current ECMAScript 5 support can be found at http://kangax.github.io/compat-table/es5/

[3] At time of publication, only Opera Mini is not supported http://caniuse.com/#feat=audio-api

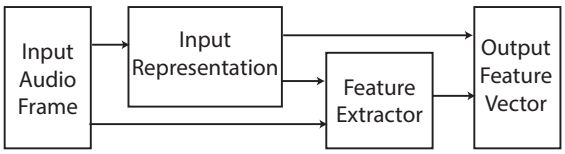

Figure 1: The JS-Xtract feature extraction topology

The API defines several nodes to enable processing and is configured using client-side JavaScript. JS-Xtract uses an Analyser node to extract the current time and frequency domain blocks for analysis, then adds a prototype function to set up an accurate interval callback using a ScriptProcessorNode to call a user-supplied function on each frame. This causes multiple function calls to occur inside the main JavaScript thread. Using the Web Audio API's AudioWorker node would be preferable for this task since it would be possible to run extraction functions in a separate JavaScript thread, however at the time of publication there are no browsers that implement this node, since the underlying WebWorker standard is incomplete [4]. To ensure separability, the current version of JS-Xtract includes the Web Audio API specific functions in a separate file (*jsXtract-wa.js*), allowing sites or projects not using the Web Audio API to still use the library.

2.3 Additional Functionality

Frames: The library implements frame-based decomposition with variable frame and hop sizes for customisable overlap. This converts a vector representing a stream of audio into sub-regions. The decomposed array is iterable and JS-Xtract supplies a prototype to simplify processing. Each frame is iterated over by calling the user supplied function with the current frame, previous frame and the previous computed value. This allows for the extraction of deltas and delta-deltas without recomputing features.
Typed arrays: For most analysis blocks, the source data is read into a JavaScript Typed Array. These allow multiple 'views' on the memory, allowing for shallow copies. Therefore data is stored efficiently as the same memory space is referred to twice, rather than copied, saving system memory. JavaScript performs all floating point calculations in double precision, therefore the only cost to using double is the increased memory footprint and conversion between single and double.
Output: The library supports simple derivation of multiple output formats. In JS-Xtract, data is returned as a JavaScript Object which can be converted to a JSON string or other data store such as XML for transmission.

2.4 Implementation

When using the `JSXtract` object, multiple instances of items such as the DCT, MFCC and Wavelets can be managed by the class. Calling `new jsXtract();` will build an object containing the initialisers and references to the function calls. Features can then be extracted using `obj.features.xtract_[Feature_Name]`. The Float32Array and Float64Array objects have the following two prototype functions applied:
`xtract_get_data_frames` and
`xtract_process_frame_data` to enable the frame decomposition and frame iteration on any floating point array data type, such as those returned for the Web Audio API buffer. Alternatively, for C-like procedural function calls, `xtract_[Feature_Name]` can be used, which is aligned with LibXtract syntax.

Feature	FF	C	S	E
Tonality	0.791	0.712	0.411	0.277
SSD	1.022	1.094	0.415	0.334
SD	0.819	0.627	0.377	0.248
ASDF	2,095	4,280	5,453	28,185
AMDF	2,319	4,252	2,476	3,102
DCT	56,395	184,412	50,618	142,830

Table 1: Feature performance in *ns* on up-to-date (July-2016) desktop browsers, where FF: Firefox, C: Chrome, S: Safari, E: Edge

Feature	iPhone	iPad	Nexus	Linx
Tonality	0.621	1.287	1.324	1.046
SSD	1.108	1.429	1.741	1.048
SD	0.517	1.263	1.192	0.959
ASDF	7,246	18,779	9,096	45,319
AMDF	4,491	9,971	8,824	8,305
DCT	64,297	169,967	315,077	347,625

Table 2: Feature performance in *ns* on mobile browsers

3. PERFORMANCE

To demonstrate the performance of the library, feature extraction was performed on a sine wave polluted with Gaussian white noise, which was 1024 samples long. Each feature was iterated 3,000 times on 41 computer - browser pairs. The fastest and slowest 3 features per call are presented in Table 1 for each of the major desktop browsers, and in Table 2 for mobile platforms.

Millisecond accurate timestamps are obtained through the W3C High Resolution Time Level 2, [4] where most major desktop and mobile browsers, support the use of the API natively. [5] A timestamp is taken before and after the iterations, giving the total time to execute the functions.

Firefox showed the best overall performance, although the slowest for the scalar features. Chrome and Edge both showed unstable performance for the DCT calculations whilst Firefox and Safari proved consistent results for the vector features (slowest three). The results show the library outperforms other JavaScript feature extraction libraries for real-time performance, and exhibits relative consistency across platforms when using scalar features. Given the efficiency, the library has the capacity to support real-time audio applications on both desktop and mobile interfaces.

[4] At time of publication, latest draft 25th February 2016
[5] See `http://caniuse.com/#feat=high-resolution-time`

4. REFERENCES

[1] Paul Adenot, Chris Wilson, and Chris Rogers. Web Audio API. *W3C, October*, 10, 2013.

[2] Jamie Bullock. Libxtract: A lightweight library for audio feature extraction. In *Proceedings of the International Computer Music Conference*, volume 43, 2007.

[3] Sebastian Ewert. Chroma toolbox: Matlab implementations for extracting variants of chroma-based audio features. In *Proc. ISMIR*, 2011.

[4] Ian Hickson. Web workers, 2015. Available at `http://www.w3.org/TR/workers/`.

[5] Nicholas Jillings, David Moffat, Brecht De Man, Joshua D Reiss, and Ryan Stables. Web audio evaluation tool: A framework for subjective assessment of audio. In *The 2nd Web Audio Conference (WAC)*. Georgia, US, 2016.

[6] Bangalore S Manjunath, Philippe Salembier, and Thomas Sikora. *Introduction to MPEG-7: multimedia content description interface*, volume 1. John Wiley & Sons, 2002.

[7] Benoit Mathieu, Slim Essid, Thomas Fillon, Jacques Prado, and Gaël Richard. Yaafe, an easy to use and efficient audio feature extraction software. In *ISMIR*, pages 441–446, 2010.

[8] Cory McKay, Ichiro Fujinaga, and Philippe Depalle. jaudio: A feature extraction library. In *Proceedings of the International Conference on Music Information Retrieval*, pages 600–3, 2005.

[9] David Moffat, David Ronan, and Joshua D Reiss. An evaluation of audio feature extraction toolboxes. In *International Conference on Digital Audio Effects (DAFx)*, 2016.

[10] Brian CJ Moore and Brian R Glasberg. Suggested formulae for calculating auditory-filter bandwidths and excitation patterns. *The Journal of the Acoustical Society of America*, 74(3):750–753, 1983.

[11] Geoffroy Peeters, Bruno L Giordano, Patrick Susini, Nicolas Misdariis, and Stephen McAdams. The timbre toolbox: Extracting audio descriptors from musical signals. *The Journal of the Acoustical Society of America*, 130(5):2902–2916, 2011.

[12] Hugh Rawlinson, Nevo Segal, and Jakub Fiala. Meyda: an audio feature extraction library for the web audio api. In *The 1st Web Audio Conference (WAC)*. Paris, Fr, 2015.

[13] Ryan Stables, Sean Enderby, Brecht De Man, György Fazekas, and Joshua Reiss. Safe: A system for the extraction and retrieval of semantic audio descriptors. In *15th International Society for Music Information Retrieval Conference (ISMIR 2014)*, 2014.

[14] Hugues Vinet, Perfecto Herrera, and François Pachet. The cuidado project. In *International Conference on Music Information Retrieval*, pages 197–203, 2002.