SCMIR: A SUPERCOLLIDER MUSIC INFORMATION RETRIEVAL LIBRARY

Nick Collins

Department of Informatics
University of Sussex, Brighton, UK
N.Collins@sussex.ac.uk

ABSTRACT

The SuperCollider Music Information Retrieval (SCMIR) library is an extension set for the SuperCollider audio programming language that facilitates automatic analysis of audio files. The framework supports common music information retrieval technologies, including for batch processing across sound file collections. The library takes advantage of scsynth's Non-Real-Time mode and machine listening plug-ins for fast feature extraction, as well as SuperCollider language invocation of auxiliary native externals for some intensive calculations. Features can be normalized/standardized, grouped by detected beat and onset locations as well as arbitrary imposed segmentations. Similarity matrices support multiple distance metrics, novelty curve calculation through checkerboard kernel, and dynamic time warping best match path discovery. Applications include automatic structure analysis of pieces, inter-piece comparison, and as a front-end to machine learning operations via SC classes or Weka.

1. INTRODUCTION

The field of **Music Information Retrieval** (MIR) [6, 7, 11, 14, 2, 4] has grown extensively in the last decade, providing a new impetus to research and commerce in computer music. MIR research is contributing new insights to sound analysis and the handling of large audio data sets, and a healthy research effort here can have bonuses for interactive music system designers.

A number of support tools for MIR researchers have been created and used for many projects. These include MATLAB libraries such as MIRToolbox [8], the C++ codebases of Marsyas [15] and the C Library for Audio and Music (CLAM) [1], as well as the jMIR Java libraries [10] and open source tools developed for the OMRAS2 project [2].[1]

Why then create a new set of MIR tools based around the SuperCollider (SC) [9, 16] synth and language? The author has actually used all of the systems mentioned in the previous paragraph, including for published projects (see for example [3]), and each provides a worthwhile contribution. Nonetheless, integration with SC's own facilities is more difficult for these third party systems, and a native solution allows a tighter leveraging of the SuperCollider language in invoking tasks and using their results. For example, the C++ codebase of SuperCollider offers its own large set of analysis UGens which it would be helpful to use for feature extraction operations.[2] Seeking ways to utilise MIR research in creative musical activity – rather than just listener oriented music services – the SuperCollider language provides a flexible platform familiar to many digital musicians. As a complete realtime system allowing interactive exploration, a close connection to sound synthesis and musical interaction can be established, providing an intriguing bridge between computer music creation and MIR.

SCMIR is available freely with full source code under GNU GPL 3, from
http://www.cogs.susx.ac.uk/users/nc81/code.html .

2. CAPABILITIES

As illustration of typical workflow with the library, Figure 1 demonstrates a sequence of commands, leading from feature extraction, through beat tracking, collection of features by beats, calculating a similarity matrix, to a novelty curve based sectional partition of an audio file (the later two stages are packaged within a convenience command, findSections). The running time for these operations over the 3 minute sound file here is 4.5 seconds, 40 times realtime.

Since version 0.6 of SCMIR, methods calls can be made directly in the main thread of the SuperCollider language. This may give the appearance of beachballing the application (when it is really hard at work), so it is also possible to wrap a set of SCMIR calls (including across a large database) in a Routine via {}.fork; this puts the commands on a separate thread, and allows posting of status messages to the Post window during calculation.

We now say a little more about the library capabilities with respect to the areas highlighted above.

[1] There are also online web services offered by universities and companies (such as services through the EchoNest API), but these are limited in application for large audio databases, where work on a local machine is much more flexible and uninhibited by upload and download limits, and third party APIs not so open to researcher tweaking (it is impossible to modify the signal analysis routines provided by Tristan Jehan in EchoNest, for example).

[2] Conversely, the evaluation of new SC machine listening UGens is facilitated by SCMIR as a feature extraction framework.

```
(
  f = SCMIRAudioFile("/mirdata/pixiesivebeentired.wav", [[MFCC, 13], [Chromagram, 12]]);
  //for sound file to analyze, set features to extract as 13 MFCC coefficients + 12 chroma

  f.extractFeatures();
  f.extractBeats();
  f.gatherFeaturesByBeats; //after this operation, featuredata will be beat locked
  b = f.findSections(0,1,20);
  //0 = cosine metric, 1 unit = 1 beat, checkerboard kernel 20 units a side
)
```

Figure 1. Example client code in SuperCollider invoking library commands. After this routine completes, global variable b contains an array of estimated section boundary locations.

2.1. Feature Extraction

The SuperCollider machine listening UGens supported for feature extraction include ZCR and RMS time domain features, alongside MFCC, Loudness (perceptual loudness model), Tartini (pitch tracker), SpecCentroid, SpecPcile, SpecFlatness, FFTCrest, FFTSpread, FFTSlope, and the Onsets UGen in raw detection function mode. These UGens are found in vanilla SC, in plug-in packs for SuperCollider, and additionally, two machine listening plugins have been written and included with SCMIR: Chromagram (allowing any n-note tuning at given reference frequency) and SensoryDissonance (the algorithm follows Sethares' sum of roughnesses for pairwise partials [13]).

As well as specifying a set of features to be extracted, a normalization scheme can be declared. Two main schemes are supported; normalization by max and min values to put features into the range 0.0 to 1.0, and statistical standardization (subtract mean, divide by standard deviation, to produce zero mean, standard deviation of 1). By default, normalization is mainly feature-wise, through it can be defined over blocks of related features, such as MFCCs or chroma. The findGlobalFeatureNorms() function allows global feature ranges and statistics to be pre-calculated over a corpus (and saved if needed for later use); otherwise, normalization is with respect to values over the single sound file currently being analyzed.

Features are taken by default over windows of 1024 samples[3] at 44100 Hz sampling rate, with no overlap, giving a frame rate of around 43Hz; 512 overlap is also supported for a doubling of frame rate and higher time resolution (at the expense of doubling calculation time). Multiple or single feature trails can be plotted from the library using SuperCollider's Pen drawing functionality.

2.2. Segmentation and Beat Tracking

Feature vectors generated per frame can be collected based on arbitrary segmentations decided by the user, or obtained by further algorithmic means. The collection is via a (featurewise) mean, or a max, taken over each window of frames to gather. Onset detection via the Onsets UGen can be used to find segment locations. Beat tracking is also available via the built-in BeatTrack SC UGen, or the BeatRoot Java external [5]. This facility allows beat locked feature data to be used for later analysis, or a stepping stone to classification operations on events; it also provides a general mechanism for non-realtime high speed discovery of onsets and beat locations for any audio file in SuperCollider.

2.3. Similarity Matrices

External programs[4] invoked from the language enable fast calculation of similarity matrices and novelty curves. A choice of cosine, Manhattan and Euclidean metrics is available. Figure 2 shows a plot of a self similarity matrix for Trevor Wishart's electroacoustic opus *Vox 5*, a six minute work. Similarity matrices can also be calculated between two sound files, where rectangular similarity matrices are supported as well as square.[5]. Novelty curves can be generated from convolution along the matrix diagonal with a checkerboard kernel of required size [4]. Potential section boundaries can be located in a subsequent peak picking stage. The peak picker searches for candidate local maxima, with a points scoring scheme for peaks with respect to their local surroundings, with a user determined threshold on points scores. For comparison of sound files, sequence comparison is supported via Dynamic Time Warping over a similarity matrix, to get a best match score and path. This can provide a potential alignment, or a measure of sequence proximity.

2.4. Further Notes

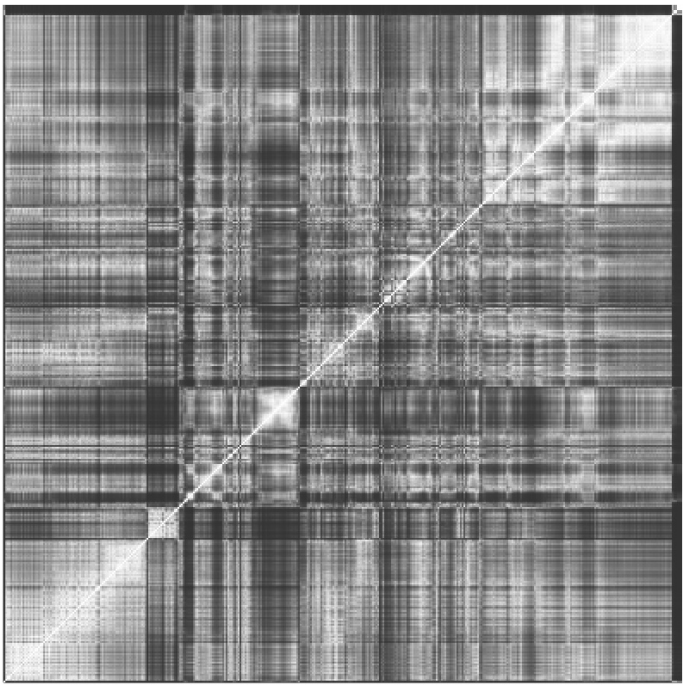
The SCMIR system runs from the SuperCollider language, invoking external programs, including the SuperCollider synthesis server scsynth in Non-Real-Time (NRT) mode, as needed.[6] Externally running unix programs are either piped (taking over the main thread), or otherwise in a side

[3] Larger window sizes with overlap are used as needed, for instance, 4096 FFTs for chromagrams with 1024 hop size, and 2048 for spectral envelope features with 1024 hop size

[4] C source code for these programs, and precompiled binaries for OS X, are provided with the library.

[5] See the SCMIRSimilarityMatrix help file for examples of using the similarity matrix calculations even without audio files, on arbitrary input data.

[6] Pending the lack of a current language plug-ins system for SuperCollider, psuedo-language plug-ins via unix processes are perfectly adequate for fast calculation.



```
(
  f = SCMIRAudioFile("wishartvox5.wav", [[MFCC, 10],[Loudness],[SpecCentroid],[SpecPcile, 0.95],
[SpecPcile, 0.8],[SpecFlatness],[FFTCrest],[FFTCrest, 0, 2000], [FFTCrest, 2000, 10000],
[FFTSpread],[FFTSlope],[SensoryDissonance],[Onsets,\rcomplex]]);
  //extraction of a large variety of features, with associated parameters

  f.extractFeatures();
  d = f.similarityMatrix(40, 0); //combine 40 frames per unit, metric 0 = cosine
  f.plotSimilarity(d,2,19); //plot at twice size, with brightness factor 19
)
```

Figure 2. Self similarity matrix for Trevor Wishart's *Vox 5* (1986), calculated using the SCMIR code above

thread polled for the finishing of their process ID.[7] In order to efficiently move feature data from scsynth's running machine listening plug-ins back to the language, a further plug-in, FeatureSave, was written, wrapping file write commands. SCMIR tidies up after itself, deleting temporary files, and a temporary directory can be set.

SCMIR operations typically lead to collections of data, such as feature signals, or a similarity matrix, which are provided as SC collection objects or instances of SCMIRSimilarityMatrix for ease of use. Save and load of SCMIRAudioFiles (including associated extracted feature trails), and global normalization templates, is provided via the use of binary archive files. Feature data can also be output as an ARFF file for external use with the Weka data mining Java application [17].[8]

The library design has already gone through various iterations; at the time of writing, the version number is 0.6. It is interesting to see the paradigms of feature gathering and similarity matrix recur in MIR at various time levels — for example, taking a similarity matrix over frames, or over beats, or over sections — the current version of the library is flexible enough to cope with extracting similarity matrices with respect to these.

3. APPLICATIONS

The library has many practical uses. It can provide a front-end for feature extraction operations, such as plotting the psychoacoustic loudness across a track, or obtaining a feature vector sequence of Mel Frequency Cepstral Coefficients. The packaging of onset detection as a faster than realtime offline facility helps in automatic segmentation operations, or as a prelude to rhythmic analysis. Beat tracking can be further used for analysis or mark-up of tracks. It is straight forward to start to build concatenative synthesis engines based on such feature extraction facilities; the same features extracted offline, are also available for real-time extraction, since SuperCollider's machine listening UGens are all inherently ready for interactive realtime use. SuperCollider also has available various machine learning UGens and language side facilities (such as Neural Nets, SOM, reinforcement learning, etc.) which can be used to create classifiers based on feature

[7] for those curious as to the exact mechanisms, see the class method Meta_SCMIR:external

[8] http://www.cs.waikato.ac.nz/ml/weka/

data. Examples are provided with the SCMIR download of genre recognition via calls to Weka from SuperCollider, and artist recognition via the NeuralNet SuperCollider class and external combination. A further file gives help for invoking Weka from the command line via SuperCollider's unix command call facilities.[9]

SCMIR has application to automatic analysis; the author has been experimenting in applying it to electroacoustic music as well as popular music. Section boundary identification is of interest in automatic discovery of formal structure. The Wishart piece *Vox 5*, for example, has been hand annotated, and the machine segmentations of section compared. In particular, early section boundaries in the piece have matched consistently well (though some later boundaries have proved more difficult to get human and machine to agree on). Whilst MIR analysis is not yet a replacement for human listening, there is definite potential here to assist analysts with interesting suggestions, as other authors have already demonstrated [12]. The analysis engine also has potential as an automatic critic for algorithmic composition works which integrate an enhanced level of machine listening technology.

4. CONCLUSION

We can improve relations between feature-matching synthesis, machine listening in interactive systems, and MIR tasks, across realtime and non-realtime tasks, by prioritising adoption of MIR technology in interactive systems.

The SCMIR library brings music information retrieval technologies to the heart of a well known existing computer music programming language. Although the library is by no means complete, and many facilities remain to be handled (from HMMs for sequence modeling to further native machine learning capabilities), a critical mass of features is already in place to enable some interesting new musical analysis capabilities for musical systems, and as building blocks to support artistic engagement with MIR.

5. REFERENCES

[1] X. Amatriain, "CLAM: A framework for audio and music application development," *IEEE Software*, vol. 24, no. 1, pp. 82–85, 2007.

[2] M. Casey, R. Veltkamp, M. Goto, M. Leman, C. Rhodes, and M. Slaney, "Content-based music information retrieval: Current directions and future challenges," *Proceedings of the IEEE*, vol. 96, no. 4, pp. 668–696, April 2008.

[3] N. Collins, "Computational analysis of musical influence: A musicological case study using mir tools," in *Proceedings of the International Symposium on Music Information Retrieval*, Utrecht, August 2010.

[4] R. Dannenberg and M. Goto, "Music structure analysis from acoustic signals," *Handbook of Signal Processing in Acoustics*, vol. 1, pp. 305–331, 2009.

[5] S. Dixon, "Evaluation of the audio beat tracking system beatroot," *Journal of New Music Research*, vol. 36, no. 1, pp. 39–50, 2007.

[6] J. S. Downie, "Music information retrieval," *Annual Review of Information Science and Technology*, vol. 37, pp. 295–340, 2003.

[7] D. P. Ellis, "Extracting information from music audio," *Communications of the ACM*, vol. 49, no. 8, pp. 32–37, 2006.

[8] O. Lartillot and P. Toiviainen, "A MATLAB toolbox for musical feature extraction from audio," in *International Conference on Digital Audio Effects (DAFx)*, Bordeaux, France, September 2007.

[9] J. McCartney, "Rethinking the computer music language: SuperCollider," *Computer Music Journal*, vol. 26, no. 4, pp. 61–8, 2002.

[10] D. McEnnis, C. McKay, and I. Fujinaga, "jAudio: Additions and improvements," in *Proceedings of the International Symposium on Music Information Retrieval*, 2006.

[11] N. Orio, "Music retrieval: A tutorial and review," *Foundations and Trends in Information Retrieval*, vol. 1, no. 1, pp. 1–90, 2006.

[12] T. H. Park, Z. Li, and W. Wu, "EASY does it: The Electro-Acoustic music analYsis toolbox," in *Proceedings of the International Symposium on Music Information Retrieval*, Kobe, Japan, 2009.

[13] W. A. Sethares, "Consonance based spectral mappings," *Computer Music Journal*, vol. 22, no. 1, pp. 56–72, 1998.

[14] M. Slaney, D. P. W. Ellis, M. Sandler, M. Goto, and M. M. Goodwin, "Introduction to the special issue on music information retrieval," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 16, no. 2, pp. 253–254, 2008.

[15] G. Tzanetakis and P. Cook, "Marsyas: a framework for audio analysis," *Organised Sound*, vol. 4, pp. 169–175, 2000.

[16] S. Wilson, D. Cottle, and N. Collins, Eds., *The SuperCollider Book*. Cambridge, MA: MIT Press, 2011.

[17] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques (2nd Ed)*. San Francisco: Morgan Kaufmann Publishers, 2005.

[9] At the time of writing, Naive Bayes and Gaussian Mixture Model classes have also been created for native SuperCollider ready for a future SCMIR 0.7 update.