SM Toolbox: MATLAB Implementations for Computing and Enhancing Similarity Matrices

Meinard Müller[1], Nanzhu Jiang[1], and Harald Grohganz[2]

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

[2]*Institut für Informatik III, Universität Bonn, 53117 Bonn, Germany*

Correspondence should be addressed to Meinard Müller (`meinard.mueller@audiolabs-erlangen.de`)

ABSTRACT

The concept of similarity matrices (SMs) has been widely used for a multitude of music analysis and retrieval tasks including audio structure analysis or version identification. For such tasks, the improvement of structural properties of the similarity matrix at an early state of the processing pipeline has turned out to be of crucial importance. In this paper, we present the SM toolbox, which contains MATLAB implementations for computing and enhancing similarity matrices in various ways. Furthermore, our toolbox includes a number of additional tools for parsing, navigation, and visualization synchronized with audio playback. Finally, we provide the code for a recently proposed audio thumbnailing procedure that demonstrates the applicability and importance of enhancement concepts. Providing MATLAB implementations on a website under a GNU-GPL license and including many illustrative examples, our aim is to foster research and education in music information retrieval.

1. INTRODUCTION

The fundamental concept of similarity matrices is central for the analysis of many kinds of time series. Generally, one starts with a feature space $\mathscr{F}$ containing the elements of the time series under consideration as well as a similarity measure $\mathbf{s} : \mathscr{F} \times \mathscr{F} \to \mathbb{R}$ that allows for comparing these elements. Then given two time series $X = (x_1, x_2, \ldots, x_N)$ and $Y = (y_1, y_2, \ldots, y_M)$, the *similarity matrix* $\mathscr{S} \in \mathbb{R}^{N \times M}$ is defined by

$$\mathscr{S}(n,m) = \mathbf{s}(x_n, x_m),$$

where $x_n, y_m \in \mathscr{F}$, $n \in [1:N] = \{1, 2, \ldots, N\}$ and $m \in [1:M]$. Typically, the value $\mathscr{S}(n,m)$ is high (dark color in Figure 1) if the two elements x_n and y_m are similar, otherwise $\mathscr{S}(n,m)$ is low (light color). Instead of a similarity measure, one often uses a cost or distance measure which then results in a so-called *cost matrix* or *distance matrix*. Since such matrices can easily be converted into similarity matrices (e. g. by taking negative values), we only consider in the following the case of similarity matrices.

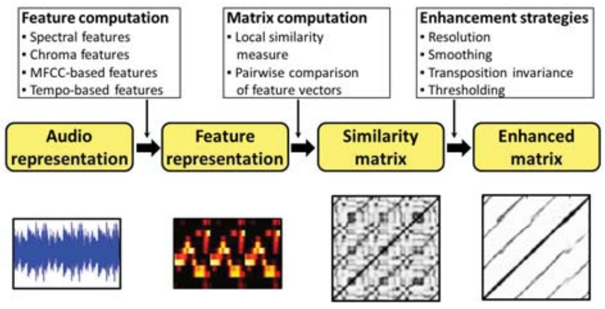

Fig. 1: Overview of the similarity matrix computation.

In the case that the sequences X and Y coincide, the resulting matrix is often referred to as *self-similarity matrix* (SSM). Such matrices have been used under the name *recurrence plot* for the analysis of chaotic systems [4]. Later, Foote [6] introduced self-similarity matrices to the music domain in order to visualize the time structure of a given audio recording. Since then similarity matrices and their relatives have been widely used for various music analysis and retrieval tasks including audio structure analysis [3, 18], structure-based audio retrieval [2], audio thumbnailing [1, 7, 16], music synchronization [5] and version or cover song identification [21, 22].

In the music context, the first step for computing a similarity matrix is to convert the given audio representations into suitable feature representations, which emphasize different musical aspects such as harmony, tempo, or timbre. The properties of the resulting similarity matrix crucially depend on the respective feature type as well as on the underlying similarity measure used to compare the features. Furthermore, many different smoothing, thresholding, and other strategies have been proposed for enhancing certain structural properties of a similarity matrix while suppressing unwanted, noise-like artifacts [18]. This leads to a large number of variants of similarity matrices, which may show quite different behaviors in the context of a specific music analysis task.

In this paper, we introduce a toolbox, the *SM toolbox*, which is released under a GNU-GPL license at [23]. This toolbox contains MATLAB implementations for computing and modifying certain properties of similarity matrices, see also Figure 1 for an overview and Figure 2 for examples. In particular, it contains functions for enhancing path-like structures that are important in repetition-based music structure analysis. Furthermore, our toolbox includes a number of additional tools for parsing, navigation, and visualization synchronized with audio playback.

Note that our toolbox is specialized with a focus on similarity matrices as they are used in applications such as structure analysis and thumbnailing. There are a number of general toolboxes for processing music and audio data. In particular, we want to mention the comprehensive MATLAB toolbox provided by Lartillot et al. [9, 10, 11] called *MIRtoolbox*. This toolbox offers a large number of functions for the extraction of audio features that refer to different musical aspects such as tonality, rhythm, and structure. The *MIRtoolbox* also supplies a basic function (called *mirsimatrix*) for computing similarity matrices. Our toolbox largely extends this functionality by providing various enhancement and visualization strategies.

The remainder of this paper is organized as follows. In Section 2, we give a short summary on the various enhancement strategies and discuss the role of the most important parameters that can be used to modify the matrices' characteristics. Then, in Section 3, we describe the functions of the toolbox. Note that we do not claim that our toolbox is comprehensive in any way. Instead, we focus on some specific techniques that are general enough to illustrate the importance of structural enhancements. Furthermore, as an example application, we have also included code for a recently proposed audio thumbnailing procedure [16], see Section 4. Finally, in Section 5, we conclude this paper with some general remarks on the suitability of the various enhancement steps depending on the specific application context.

2. MATRIX ENHANCEMENT

In this section, we give an overview of the various enhancement strategies contained in the SM toolbox. Even though all enhancement strategies are implemented for general similarity matrices, we consider in the following only the case of self-similarity matrices for the sake of simplicity. As illustration, Figure 2 shows various variants of self-similarity matrices for an audio recording consisting of four parts $A_1A_2BA_3$. In this example, A_2 is a modulation of A_1 transposed by one semitone upwards, whereas A_3 is a repetition of A_1, however played much faster.

2.1. Feature Representation

In the first step, the waveform-based audio recordings are transformed into suitable feature representations, which capture specific acoustic and musical properties. As detailed in [18], the suitability of a feature type largely depends on the respective application. For example, MFCC-based and related spectral-based features may be suitable to capture aspects such as instrumentation and timbre. Other features based on onset and novelty curves or tempograms are used to capture beat, tempo, and rhythmic information. Finally, chroma-based audio features, which relate to harmonic and melodic properties, have turned out to be a powerful tool for many music analysis and retrieval tasks. Each 12-dimensional chroma vector describes a signal's local energy distribution over an analysis window (frame) across the 12 pitch classes of the equal tempered scale (ignoring octave information). Hence, the resulting feature space is $\mathscr{F} = \mathbb{R}^{12}$. As an example, we use in the following a chroma variant referred to as `CENS` features[1], which

[1]A MATLAB implementation of CENS features is part of the Chroma Toolbox, which is freely available at `http://www.mpi-inf.mpg.de/resources/MIR/chromatoolbox/`

come along with two parameters: a length parameter $w \in \mathbb{N}$ controlling the size of the analysis window (frame length) and a downsampling parameter d controlling the feature rate, see [15] for details.

2.2. Similarity Measure

As mentioned in the introduction, one requires a notion of similarity (or dissimilarity, distance, cost) for a quantitative comparison of two elements $x, y \in \mathscr{F}$. In the case of $\mathscr{F} = \mathbb{R}^D$ being a Euclidean space of dimension D, typical measures are based on the ℓ^p-norm defined by $\|x\|_p = \left(\sum_{i=1}^{D} |x(i)|^p\right)^{1/p}$ for a vector $x = (x(1), x(2), \ldots, x(D))^\mathrm{T}$. Then, for example, a similarity measure $\mathbf{s}$ may be obtained by setting $\mathbf{s}(x,y) = a - \|x-y\|_p^b$ for constants $a \in \mathbb{R}$ and $b \in \mathbb{N}$. In the following, we only consider the case $p = 2$ and assume that x and y are normalized with respect to this norm. Then, using $a = 2$ and $b = 2$, the measure $\mathbf{s}$ boils down to the inner product $\langle x|y\rangle$ (up to a factor of two), which measures the cosine of the angle between x and y.

2.3. Smoothing

One important property of similarity matrices is the appearance of paths of high similarity that are parallel to the main diagonal [18, 20]. Each such path encodes the similarity of two segments that are obtained by projecting the path onto the horizontal and vertical axes, respectively. The identification and extraction of such paths is the main step in many music analysis applications. However, due to musical and acoustic variations, the path structure is often very noisy and hard to extract.

To some extent, such noise can be reduced simply by using longer analysis windows in the feature computation step and adjusting the feature rate. To further enhance the path structure, one general strategy is to apply some kind of smoothing filter along the direction of the main diagonal, resulting in an emphasis of diagonal information in $\mathscr{S}$ and a denoising of other structures, see [1, 17, 19, 22] and Figure 2b. Such a filtering process is closely related to the concept of *time-delay embedding*, which has been widely used for the analysis of dynamical systems [13]. A simple filtering along the main diagonal only works well if there are no relative tempo differences between the segments to be compared. However, this assumption is often violated for music, where a part may be repeated with a faster or slower tempo. To deal with such tempo difference, a multiple filtering approach has been suggested in [17], where a similarity matrix is filtered along various directions that lie in a neighborhood of the direction defined by the main diagonal. Each such direction corresponds to a tempo difference and results in a separate filtered similarity matrix. The final similarity matrix is obtained by taking the cell-wise maximum over all these matrices. In this way, the path structure is also enhanced in the presence of local tempo variations as illustrated in Figure 2c.

In our implementation, we have simulated the multiple filtering approach by an efficient procedure that is based on a combination of feature and matrix resampling steps and simple diagonal smoothing. All operations can be expressed by full matrix operations, which are efficiently realized in MATLAB. Two main parameters are provided for controlling the smoothing quality: a smoothing length parameter ℓ and discrete set Θ of relative tempo differences, see Section 3 for more explanations.

The implemented smoothing filter is realized to smooth in the forward direction, which results in a fading out of the paths in particular when using a large length parameter. To avoid this fading out, one can use a forward-backward option, which applies the filter also in backward direction. The final similarity matrix is then obtained by taking the cell-wise maximum over the forward-smoothed and backward-smoothed matrices, see Figure 2d.

2.4. Transposition Invariance

It is often the case that certain musical parts are repeated in a transposed form as the part A_2 in our example. Such transpositions can be simulated by cyclically shifting chroma vectors [7]. In [14] this idea was used to construct *transposition-invariant* similarity matrices. To this end, one chroma feature sequence is left unaltered whereas the other chroma feature sequence is cyclically shifted along the chroma dimension in the twelve possible ways. Then, for each shifted version, a similarity matrix is computed, and the final similarity matrix is obtained by taking the cell-wise maximum over the twelve matrices. In this way, the repetitive structure is revealed even in the presence of key transpositions (Figure 2e). Furthermore, storing the maximizing shift index for each cell results in another matrix referred to as *transposition index matrix*, which displays the harmonic relations within the music recording (Figure 2f). For example, this

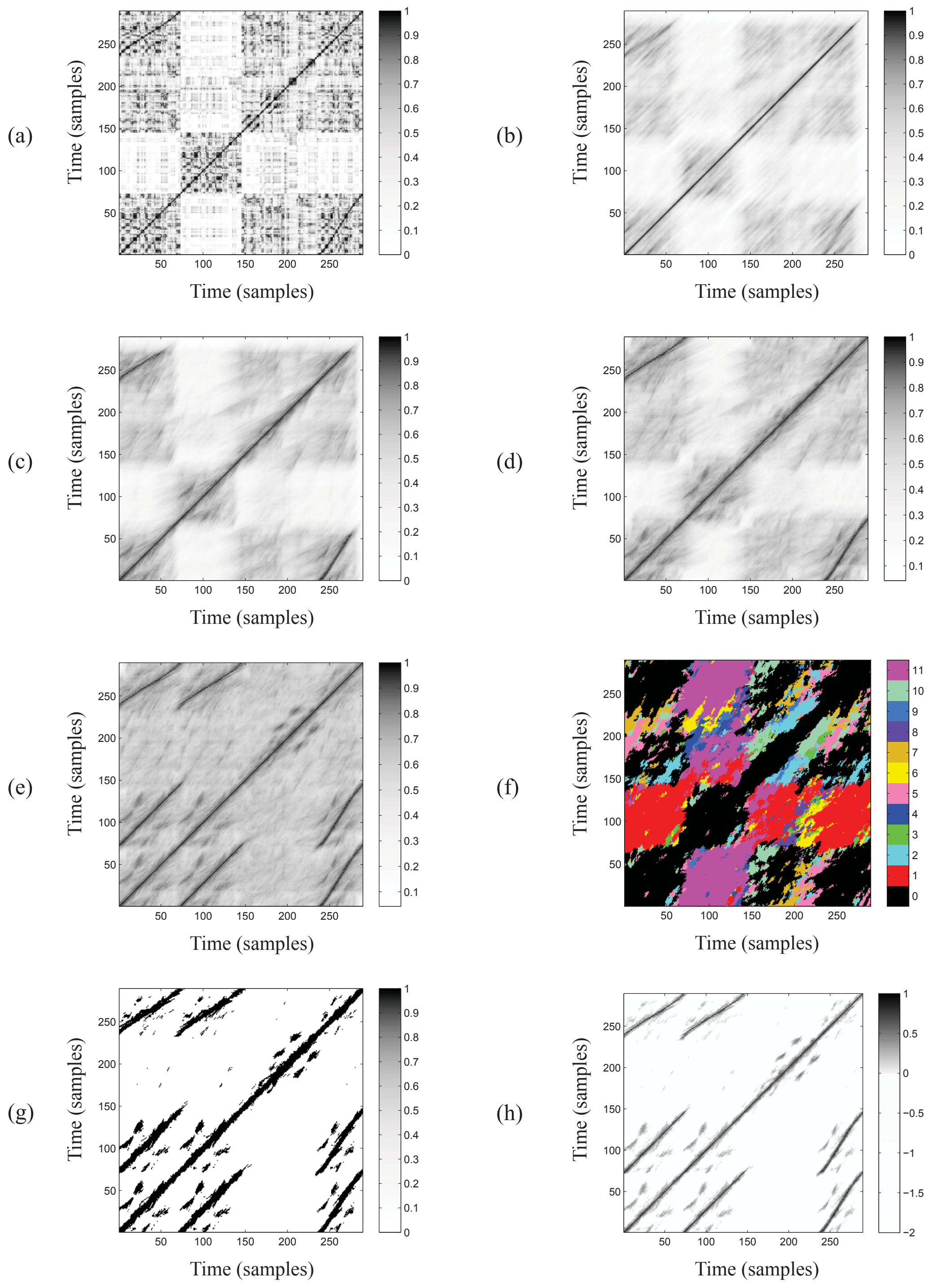

Fig. 2: Variants of similarity matrices for the same audio recording. The figures are generated using the code shown in Table 2. **(a)** Original SSM using `CENS` features of 2 Hz resolution. **(b)** SSM after applying diagonal smoothing. **(c)** SSM after applying tempo-invariant smoothing. **(d)** SSM after appyling forward-backward smoothing. **(e)** Transposition-invariant SSM. **(f)** Transposition index matrix. **(g)** SSM after thresholding and binarization. **(h)** SSM after thresholding, scaling, and applying a penalty parameter.

matrix reveals that the A_2-segment is indeed transposed by one semitone upwards relative to the A_1-segment.

In our implementation, we have provided a parameter Γ for specifying the chroma indices to be considered in the cyclic shifts. For example, $\Gamma = [0]$ leads to the original similarity matrix, whereas $\Gamma = [0:11]$ leads to the transposition-invariant version. At this point, we want to note that introducing transposition-invariance by cell-wise maximization over several matrices may increase the noise-level in the resulting similarity matrix. Therefore, the transposition-invariant matrix should be computed on the basis of smoothed matrices, since the smoothing typically goes along with a suppression of unwanted noise.

2.5. Thresholding

In many music analysis applications, similarity matrices are further processed by suppressing all values that fall below a given threshold. On the one hand, such a step often leads to a substantial reduction of the noise while leaving only the most significant structures. On the other hand, weaker but still relevant information may be lost. Actually, the thresholding strategy may have a significant impact on the final results and has to be carefully chosen in the context of the considered application.

In its simplest form, one can apply a global thresholding strategy. In our implementation, providing a threshold parameter $\tau > 0$, all values $\mathscr{S}(n,m)$ of a given similarity matrix $\mathscr{S}$ below τ are set to zero. Also binarization of the similarity matrix can be applied by setting all values above the threshold to one and all others to zero, see Figure 2g. Instead of binarization, one may perform a scaling where the range $[\tau : \mu]$ is linearly scaled to $[0:1]$ in the case that $\mu = \max_{n,m}\{\mathscr{S}(n,m)\} > \tau$, otherwise all entries are set to zero. Sometimes it may be beneficial to introduce an additional penalty parameter $\delta \leq 0$ and setting all original values below the threshold to the value δ. The global threshold τ can be chosen also in a relative fashion using the parameter ρ by keeping $\rho \cdot 100\%$ of the cells having the highest value, see (Figure 2h). Finally, as described in [22], thresholding can also be performed using a more local strategy by thresholding in a column- and row-wise fashion. To this end, for each cell (n,m) the value $\mathscr{S}(n,m)$ is kept if it is among the $\rho \cdot 100\%$ of the largest cells in row n and at the same time among the $\rho \cdot 100\%$ of the largest cells in column m, all other values are set to zero.

3. TOOLBOX

The matrix enhancement components as described in Section 2 form the core of our SM toolbox, which is freely available at the website [23] under a GNU-GPL license. Table 1 gives an overview of the main MATLAB functions along with the most important parameters. Note that there are many more parameters and additional functions not discussed in this paper.

To demonstrate how our toolbox can be applied, we now discuss the code example shown in Table 2, which is also contained in the toolbox as function `demoSMtoolbox.m`. Our example starts in lines 1–8 with computing a suitable chroma-based feature representation for the given audio recording. The used functions are part of the Chroma Toolbox [15]. Note that these features only serve as an example and any other feature representation may be used equally well in the following steps. The call to the function `wav_to_audio`, which is a simple wrapper around MATLAB's `wavread.m`, converts the input WAV file into a mono version at a sampling rate of 22050 Hz. Next, `Pitch` and `CENS` features are computed, where the struct `paramPitch` and the struct `paramCENS` are used to pass optional parameters to the feature extraction function. If some parameters or the whole struct are not set manually, then meaningful default settings are used. This is a general principle, which applies for the chroma toolbox as well as for the SM toolbox. In the current settings, the resulting `CENS` features have a feature resolution of 2 Hz, see [15] for details.

In lines 10–49, the various self-similarity matrices as shown in Figure 2 are computed, where different parameter settings that are encoded by the struct `paramSM` are used. First, in line 10 a self-similarity matrix `S` is computed by comparing `f_CENS` with itself. Note that one may also input two different feature sequences resulting in a more general similarity matrix. Furthermore, note that no parameters are specified in the function call `features_to_SM`. As a result, the function-internal default settings are used, where a simple inner product is used as similarity measure and no matrix enhancement is applied. The matrix is visualized by the function `visualizeSM` (line 12) using a colormap that is specified by the parameter `paramVis.colormapPreset` (line 11), see Figure 2a.

Next, various enhancement strategies are activated by setting the corresponding parameter values. In line 14,

Filename	Main parameters	Description
`wav_to_audio.m`	–	Import of WAV files and conversion to expected audio format.
`audio_to_pitch_via_FB.m`	winLenSTMSP	Extraction of pitch features from audio data.
`pitch_to_CENS.m`	winLenSmooth $\widehat{=} w$, downsampSmooth $\widehat{=} d$	Derivation of `CENS` features from `Pitch` features.
`features_to_SM`	smoothLenSM $\widehat{=} \ell$, forwardBackward	Smoothing of SM.
	(tempoRelMin, tempoRelMax, tempoNum) $\widehat{=} \Theta$	Application of tempo invariance.
	circShift $\widehat{=} \Gamma$	Application of transposition invariance.
`threshSM`	threshTechnique, threshValue $\widehat{=} \tau$ or $\widehat{=} \rho$	Application of different thresholding techniques.
	penality $\widehat{=} \delta$, applyBinarize, applyScale	Application of binarization or scaling.
`visualizeSM`	colormapPreset, print, figureName, imageRange	Visualation of similarity matrix.
`visualizeTransIndex`	colormapPreset, print, figureName	Visualization of transposition index matrix.
`makePlotPlayable`	featureRate, fs	Synchronized playback of audio file along with a plotted figure.
`SSM_to_scapePlotFitness`	stepSize, stepWeight	Computation of fitness scape plot from self-similarity matrix.
`scapePlotFitness_to_thumbnail`	lowerBound	Computation of thumbnail segment from fitness scape plot.
`thumbnailSSM_to_pathFamily`		Computation of induced segment family from thumbnail.
`visualizeScapePlot`	featureRate, print, figureName	Visualization of fitness scape plot.
`visualizePathFamilySSM`	featureRate, showSegInduced, showThumbnail	Visualization of similarity matrix and path family.
`visualizeSegFamily`	print, figureName	Visualization of segment family.

Table 1: Overview of the main MATLAB functions contained in SM toolbox [23] and the most important parameters. The first three functions for feature extraction are contained in the Chroma Toolbox [15].

the smoothing length parameter ℓ is set to 20 (given in feature samples), which corresponds to 10 s of the original audio when using a feature rate of 2 Hz, see Figure 2b. In lines 18–20, the discrete set Θ used for tempo-invariant smoothing is defined by three different parameters: `tempoRelMin` specifies the minimal relative tempo difference contained in Θ, `tempoRelMax` the maximal relative tempo difference contained in Θ, and `tempoNum` the actual number of elements of Θ (using a logarithmic sampling between `tempoRelMin` and `tempoRelMax` for the intermediate values). The resulting matrix is again visualized by line 22, see Figure 2c. In line 24, the forward-backward smoothing is activated, see Figure 2d. Then, in line 28, the transposition-invariance using all twelve possible cyclic chroma shifts is activated. In line 25, the self-similarity matrix `S` as well as the transposition index matrix `I` are returned and visualized in the next two lines, see Figure 2e/f. Finally, the similarity matrix is further processed by applying the thresholding function `threshSM`. Various thresholding strategies specified by the parameter `threshTechnique` are available. In lines 33–36, a simple global thresholding (`threshTechnique=1`) using an absolute threshold $\tau = 0.75$ (`threshValue=0.75`) and binarization (`applyBinarize=1`) is applied, see Figure 2g. Similarly, in lines 39–44, relative thresholding (`threshTechnique=2`) with $\rho = 0.15$ (`threshValue=0.15`) as well as with scaling and penalty is applied, see Figure 2h. Note that depending on the thresholding technique, the parameter `threshValue` is interpreted either as absolute threshold or as relative threshold. Finally, note that further parameters can be specified for the visualization function including a `print` option to save the generated figure as `.eps` file, see lines 45–49.

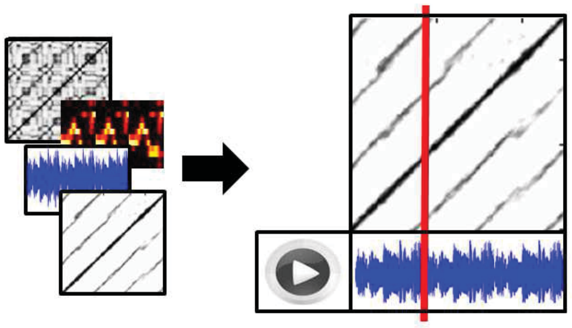

Fig. 3: Tool for visualization synchronized with audio playback.

```
 1  clear;close all;
 2  filename='Test_AABA.wav';
 3  f_audio=wav_to_audio('','data_music/',filename);
 4  paramPitch.winLenSTMSP=4410;
 5  f_pitch=audio_to_pitch_via_FB(f_audio,paramPitch);
 6  paramCENS.winLenSmooth=11;
 7  paramCENS.downsampSmooth=5;
 8  f_CENS=pitch_to_CENS(f_pitch,paramCENS);
 9
10  S=features_to_SM(f_CENS,f_CENS);
11  paramVis.colormapPreset=2;
12  visualizeSM(S,paramVis);
13
14  paramSM.smoothLenSM=20;
15  S=features_to_SM(f_CENS,f_CENS,paramSM);
16  visualize_SM(S,paramVis);
17
18  paramSM.tempoRelMin=0.5;
19  paramSM.tempoRelMax=2;
20  paramSM.tempoNum=7;
21  S=features_to_SM(f_CENS,f_CENS,paramSM);
22  visualizeSM(S,paramVis);
23
24  paramSM.forwardBackward=1;
25  S=features_to_SM(f_CENS,f_CENS,paramSM);
26  visualizeSM(S,paramVis);
27
28  paramSM.circShift=[0:11];
29  [S,I]=features_to_SM(f_CENS,f_CENS,paramSM);
30  visualizeSM(S,paramVis);
31  visualizeTransIndex(I);
32
33  paramThres.threshTechnique=1;
34  paramThres.threshValue=0.75;
35  paramThres.applyBinarize=1;
36  S_thres=threshSM(S,paramThres);
37  visualizeSM(S_thres,paramVis);
38
39  paramThres.threshTechnique=2;
40  paramThres.threshValue=0.15;
41  paramThres.applyBinarize=0;
42  paramThres.applyScale=1;
43  paramThres.penalty=-2;
44  S_final=threshSM(S,paramThres);
45  paramVis.imageRange=[-2,1];
46  paramVis.colormapPreset=3;
47  paramVis.print=1;
48  paramVis.figureName='SM_final';
49  handleFigure=visualizeSM(S_final,paramVis);
50
51  parameterMPP.fs=22050;
52  parameterMPP.featureRate=2;
53  makePlotPlayable(f_audio,handleFigure,parameterMPP);
```

Table 2: Code example generating matrices shown in Figure 2.

As another feature of our toolbox, we provide a function `makePlotPlayable` that allows a user to playback the original audio file synchronized to any kind of feature or matrix representation derived from this audio file, see Figure 3 for a schematic illustration of this functionality. As an example, lines 51 – 53 show how to call this function, where the audio signal and the handle of the figure (as returned by the function `visualizeSM` in line 49) need to be specified. Furthermore, parameters for the sampling rate of the audio signal and the feature rate used in the representation of the figure are specified. During playback of the audio, the function indicates in the figure the corresponding position by a moving vertical line. Vice versa, by left clicking on any position of the figure's time line allows for jumping to the corresponding position in the audio signal (whereas a right click stops the playback). This simple functionality is of great help for analyzing and better understanding the properties of a feature representation in a musically informed way.

4. THUMBNAILING APPLICATION

As an illustrative application, our toolbox also contains the MATLAB code for a recently proposed audio thumbnailing procedure [16]. For this task, the goal is to find the most representative and repetitive segment of a given audio recording. Based on a suitable self-similarity matrix, the procedure in [16] computes for each audio segment a fitness value that expresses how well the given segment explains other related segments (also called induced segments) in the audio recording. These relations are expressed by a so-called path family over the given segment. The thumbnail is then defined as the fitness-maximizing segment. Furthermore, a triangular scape plot representation is computed, which shows the fitness of all segments and yields a compact high-level view of the structural properties of the entire audio recording.

Table 1 shows the main functions implementing this procedure. Starting with a self-similarity matrix (as computed in line 44 of Table 2), the function `SSM_to_scapePlotFitness` is used to derive the fitness scape plot. Here, various step size and weighting parameters can be used to adjust the procedure. As a result, one obtains a fitness scape plot which can be visualized by `visualizeScapePlot`, see Figure 4a. Furthermore, using the fitness scape plot as input, the function `scapePlotFitness_to_thumbnail` outputs the thumbnail. Then, together with the SSM, the function `thumbnailSSM_to_pathFamily` computes the corresponding path family and the induced segment family, which can be visualized by the function `visualizePathFamilySSM`, see Figure 4b. Finally, the function `visualizeSegFamily` can be used for

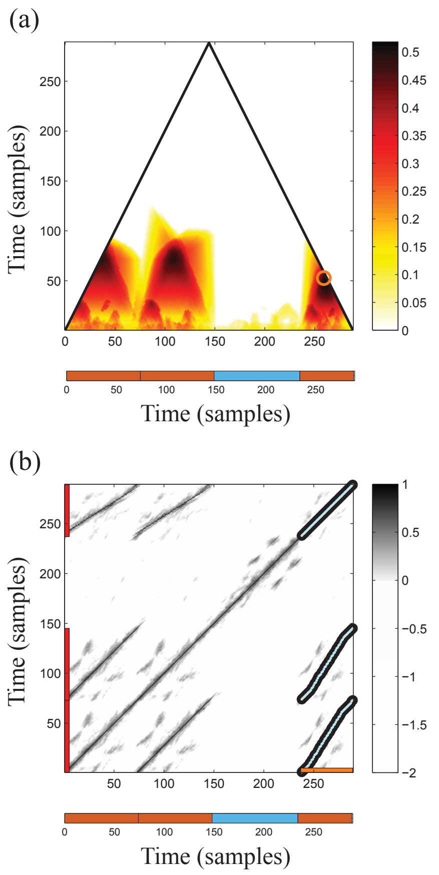

Fig. 4: Thumbnailing application. **(a)** Fitness scape plot and ground-truth segmentation. **(b)** SSM with thumbnail (shown on horizontal axis), path family (cyan), induced segment family (shown on vertical axis) and ground-truth segmentation.

visualizing any segment family, e. g., the ground-truth segmentation as shown in Figure 4c.

Besides these main functions for the thumbnailing application, the SM toolbox contains a number of required sub-functions as well as optimized C++ code for computing path families compiled as a `mex`-file (which can be called from MATLAB). Furthermore, the toolbox contains additional demo files for more complex audio recordings.

5. DISCUSSION

As noted before, many variants of similarity matrices based on different features, similarity measures, and enhancements have been suggested in the MIR literature for analyzing, comparing, structuring, and retrieving audio material. At this point, we want to emphasize that there is no single variant that works best in all situations and the requirements of the used similarity matrix very much depends on the specific application in mind. For example, for many tasks related to cover song identification [21] or audio structure analysis [3, 7, 18] audio-based chroma features at a feature resolution of roughly 2 Hz have turned out to be a meaningful choice. Obviously, such resolutions are much too coarse when considering tasks such as high-resolution music synchronization [5]. When considering segmentation and classification tasks based on, e. g., timbre rather than harmony, one needs to use different features such as MFCCs [6]. Also the smoothing variant very much depends on the application. As noted in [18], similarity matrices typically contain path-like structures (accounting for repetition-based properties) and block-like structures (accounting for homogeneity-based properties). The smoothing variants discussed in this paper enhance path-like structures, but destroy block-like structures. This is not always wanted. For example, when performing homogeneity-based structure analysis [8, 12], one requires different smoothing techniques that enhance the block structure. Generally speaking, smoothing decreases the noise level in similarity matrices, thus introducing additional robustness to the overall procedure. On the downside, valuable structural information may be smoothed out and lost for the subsequent analysis. Another quite obvious but important remark is that one should only apply enhancement strategies if they are actually needed. For example, when performing structure analysis for music with constant tempo (which is often the case for popular music) a simple diagonal smoothing may do the job and tempo-invariance is not needed. Actually, applying tempo-invariance in this situation may even introduce unwanted artifacts. Similarly, if one does not expect any modulations, transposition-invariance should not be used. The reason is that achieving invariance also comes at some cost. For example, computing tempo- and transposition-invariant similarity matrices as done in this toolbox, the noise of the individual matrices may penetrate to the final matrix obtained by maximization, which may increase the overall noise level. Thus, such strategies need to be applied with care and also the order in which enhancement strategies are applied may have a significant impact on the final results.

As a general goal of this paper and our toolbox, we want to raise the awareness of such issues. Also, providing cleaned-up example code, we hope that our toolbox may inspire future research in music information retrieval and may serve as illustrative material in education.

Acknowledgments: This work has been supported by the German Research Foundation (DFG CL 64/8-1, DFG MU 2682/5-1). The International Audio Laboratories Erlangen are a joint institution of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and the Fraunhofer-Institut für Integrierte Schaltungen IIS.

6. REFERENCES

[1] M. A. Bartsch and G. H. Wakefield. Audio thumbnailing of popular music using chroma-based representations. *IEEE Transactions on Multimedia*, 7(1):96–104, 2005.

[2] J. P. Bello. Measuring structural similarity in music. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(7):2013–2025, September 2011.

[3] R. B. Dannenberg and M. Goto. Music structure analysis from acoustic signals. In D. Havelock, S. Kuwano, and M. Vorländer, editors, *Handbook of Signal Processing in Acoustics*, volume 1, pages 305–331. Springer, New York, NY, USA, 2008.

[4] J.-P. Eckmann, S. O. Kamphorst, and D. Ruelle. Recurrence plots of dynamical systems. *Europhysics Letters*, 4(9):973–977, 1987.

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

[6] J. Foote. Visualizing music and audio using self-similarity. In *Proceedings of the ACM International Conference on Multimedia*, pages 77–80, Orlando, FL, USA, 1999.

[7] M. Goto. A chorus section detection method for musical audio signals and its application to a music listening station. *IEEE Transactions on Audio, Speech and Language Processing*, 14(5):1783–1794, 2006.

[8] F. Kaiser and T. Sikora. Music structure discovery in popular music using non-negative matrix factorization. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 429–434, Utrecht, The Netherlands, 2010.

[9] O. Lartillot. MIRtoolbox 1.5, User's Manual. `https://www.jyu.fi/hum/laitokset/musiikki/en/research/coe/materials/mirtoolbox/MIRtoolbox1.5Guide/`, Retrieved 10.09.2013, 2013.

[10] O. Lartillot and P. Toiviainen. A matlab toolbox for musical feature extraction from audio. In *Proceedings of the 10th International Conference on Digital Audio Effects (DAFx)*, Bordeaux, France, 2007.

[11] O. Lartillot and P. Toiviainen. MIR in Matlab (II): A toolbox for musical feature extraction from audio. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, pages 127–130, Vienna, Austria, 2007.

[12] M. Levy and M. Sandler. Structural segmentation of musical audio by constrained clustering. *IEEE Transactions on Audio, Speech and Language Processing*, 16(2):318–326, 2008.

[13] N. Marwan, M. C. Romano, M. Thiel, and J. Kurths. Recurrence plots for the analysis of complex systems. *Physics Reports*, 438(5-6):237–329, 2007.

[14] M. Müller and M. Clausen. Transposition-invariant self-similarity matrices. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR)*, pages 47–50, Vienna, Austria, 2007.

[15] M. Müller and S. Ewert. Chroma Toolbox: MATLAB implementations for extracting variants of chroma-based audio features. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 215–220, Miami, FL, USA, 2011.

[16] M. Müller, N. Jiang, and P. Grosche. A robust fitness measure for capturing repetitions in music recordings with applications to audio thumbnailing. *IEEE Transactions on Audio, Speech & Language Processing*, 21(3):531–543, 2013.

[17] M. Müller and F. Kurth. Enhancing similarity matrices for music audio analysis. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 437–440,

Toulouse, France, 2006.

[18] J. Paulus, M. Müller, and A. P. Klapuri. Audio-based music structure analysis. In *Proceedings of the 11th International Conference on Music Information Retrieval (ISMIR)*, pages 625–636, Utrecht, The Netherlands, 2010.

[19] G. Peeters. Sequence representation of music structure using higher-order similarity matrix and maximum-likelihood approach. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, pages 35–40, Vienna, Austria, 2007.

[20] G. Peeters, A. L. Burthe, and X. Rodet. Toward automatic music audio summary generation from signal analysis. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, Paris, France, 2002.

[21] J. Serrà, E. Gómez, P. Herrera, and X. Serra. Chroma binary similarity and local alignment applied to cover song identification. *IEEE Transactions on Audio, Speech and Language Processing*, 16:1138–1151, 2008.

[22] J. Serrà, X. Serra, and R. G. Andrzejak. Cross recurrence quantification for cover song identification. *New Journal of Physics*, 11(9):093017, 2009.

[23] SM Toolbox. `http://www.audiolabs-erlangen.de/resources/MIR/SMtoolbox/`, Retrieved 10.09.2013.