CNN-BASED SEGMENT BOUNDARY DETECTION ON NOVEL SSMS WITH IMPROVED INTRA-CLASS SIMILARITY FOR REPETITIONS

Tian CHENG (tian.cheng@aist.go.jp) [1], **Tomoyasu NAKANO** (t.nakano@aist.go.jp) [1], and **Masataka GOTO** (m.goto@aist.go.jp) [1]

[1] **National Institute of Advanced Industrial Science and Technology (AIST)**, Tsukuba, Japan

ABSTRACT

This paper addresses the segment boundary detection task that is to detect boundaries between consecutive segments in musical pieces. In the commonly used Self-Similarity Matrix (SSM) for music structure analysis, we observe two types of segment, homogeneous and repetitive segments indicated by square blocks and diagonal stripes, respectively. The standard SSM with frame-wise similarity is suitable for detecting homogeneous segments. However, it remains a challenge to effectively represent the intra-class similarity for repetitive segments. To tackle this issue, we propose two novel SSMs, RSSM_value and RSSM_index, based on our observation that vertical column vectors during diagonal stripes (repetitive segments) in the SSM are similar to each other after shifting these vectors up or down. For each pair of column vectors in SSM, we compute their cosine similarity while shifting one column vector to find the shift index that yields the maximal similarity value. We then define RSSM_value where each element represents the maximal value and RSSM_index where each element represents the shift index, resulting in clear blocks for repetitive segments. We stack the standard and proposed SSMs and apply a CNN model to detect the segment boundaries. Experimental results show that our method outperforms state-of-the-art methods on three test-only datasets (RWC_pop, Beatles, and SALAMI).

1. INTRODUCTION

The task of Music Structure Analysis (MSA) consists of detecting boundaries between consecutive segments and grouping segments into relevant categories, referred to as boundary detection and segment labelling, respectively. This task has a long history in the field of Music Information Retrieval (MIR) [1–4] and still gains attention [5–10] because structures play an essential role in music understanding and composition. MSA has been used to facilitate other music content analysis tasks such as music similarity estimation [4], beat and downbeat tracking [11, 12], and chord transcription [13]. Structural information has also been used in music generation tasks [6, 14, 15].

Most of the existing MSA methods divide up musical pieces based on the homogeneity within a segment and the repetition of different segments [6]. The Self-Similarity Matrix (SSM) is a commonly used mid-level representation for MSA, and it is generated by computing the similarity between two audio feature vectors for each matrix element. The SSM is considered as a standard tool in MSA because it reveals two main types of segments: homogeneous segments and repetitive segments. A homogeneous segment is indicated by a square block (i.e., a region where matrix elements with high similarity values form a square shape) in the SSM, indicating high similarities between pairs of feature vectors within the segment. On the other hand, the repetition of segments typically leads to diagonal stripes in the SSM [16], with the stripes representing the high sequential similarity between the repetitive segments. In order to effectively divide a musical piece into meaningful segments, an ideal SSM should have high values for matrix elements in the shape of square blocks belonging to the same class (high *intra-class* similarity) [9, 17].

This has motivated researchers to work on representing the diagonal stripes with block-like structures for a better presentation of intra-class similarity for repetitive segments. Previous work achieved block-enhanced SSMs by converting the diagonal stripes in the SSM into blocks by using eigenvalue decompositions [16], spectral clustering [18], and non-negative matrix factor 2-D deconvolution (NMF2D) [19]. Recent deep-learning-based methods first learned features from patches of spectral features to enforce intra-class similarity by using the triplet loss [9, 17, 20, 21] or SSM-based loss [10]. They then used methods based on (classical or trained [10]) checkerboards or clustering to detect segment boundaries in the block SSMs computed on the learned features.

In this paper, we address the segment boundary detection task in MSA. In contrast to the above deep-learning-based methods, we first compute two hand-crafted novel SSMs, RSSM_value and RSSM_index, by using the standard SSM, and then apply a CNN model for boundary detection as shown in Figure 1. We observed that although the vertical column vectors within each repetitive segment are not similar to each other, similar vectors repeatedly appear across different repetitive segments in the SSM, as depicted in the eight blue rectangles in Figure 2. Moreover, a column vector within each repetitive segment (i.e., within consecutive column vectors showing diagonal stripes in each blue rectangle in Figure 2) can be similar to another vector even within the same repetitive segment after shifting the vector up or down thanks to the nature

ISBN 978-3-200-10642-0

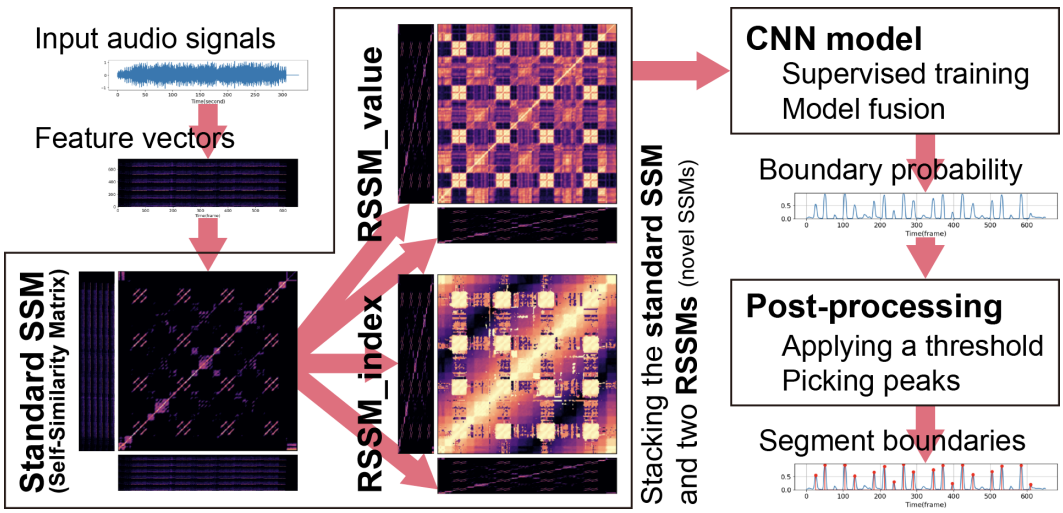


Figure 1: Overview of the proposed method.

of diagonal stripes. Our RSSM_value and RSSM_index are computed from the standard SSM by leveraging these findings and can have much clearer square blocks for repetitive segments. After stacking the proposed RSSMs with the standard SSM, the CNN model for them is trained with a model fusion technique that averages the weights of the models at several smallest validation losses to achieve more robust results. The model was trained and validated on the Harmonix [22] dataset, and it was evaluated on the RWC_pop [23], Beatles [24], and SALAMI [25] datasets. The results showed that it outperformed state-of-the-art methods [9, 10, 17, 20, 21, 26–28].

2. PROPOSED METHOD

On the basis of the standard SSM described in Section 2.1, we define two RSSMs in Section 2.2 to improve the intra-class similarity for repetitive segments. As shown in Figure 1, we apply the CNN model for segment boundary detection (Sections 2.3 and 2.4) to the standard SSM and two RSSMs, followed by the post-processing (Section 2.5).

2.1 SSM Computation

We compute the standard SSM based on the mel-spectrogram of musical pieces by using the Python library `librosa` [29]. First, we load the audio with a sampling rate of 22050 Hz and compute the mel-spectrogram with 128 mel bins and a hop size of 441 (20 ms). We convert the mel-spectrogram into a log amplitude with $\log(1 + mel)$.

Then, we compute mean features at a rate of 0.5 s (we call this a *frame*) with each feature obtained by averaging the above mel-spectral features within 2 s. We obtain the audio feature vector $\boldsymbol{f}_i$ by stacking the mean features in adjacent 6 frames (3 s) for each frame i [26], with $\boldsymbol{f}_i \in \mathbb{R}^{768\times 1}, i \in [1, T]$, where $T = \text{audio\_length}/0.5$ is the the number of frames in the sequence. Finally, we compute the SSM based on the Euclidean distance between pairs of the audio feature vectors:

$$\text{SSM}_{i,j} = \exp(-||\boldsymbol{f}_i - \boldsymbol{f}_j||/\text{b}), \quad (1)$$

where b is the bandwidth parameter. We use the librosa function `librosa.segment.recurrence_matrix` to compute the SSM, with `mode='affinity'` and default setting for the bandwidth b.

To obtain a high contrast SSM, the above exp is computed only when $\boldsymbol{f}_i$ and $\boldsymbol{f}_j$ are similar (close) enough, which is judged by using k-nearest neighbors (see the librosa implementation [29] for details), and we set $\text{SSM}_{i,j} = 0$ otherwise. We smooth the SSM by taking the average of 7 frames along the diagonal direction to reduce noisy short sequences.

An element $\text{SSM}_{i,j}$ of the SSM $\in \mathbb{R}^{T\times T}$ indicates the similarity between $\boldsymbol{f}_i$ and $\boldsymbol{f}_j$, with a value between 0 and 1. If $\text{SSM}_{i,j}$ is close to 1, $\boldsymbol{f}_i$ is similar to $\boldsymbol{f}_j$. For homogeneous segments with *frame-wise similarity*, all pairs of $\boldsymbol{f}_i$ and $\boldsymbol{f}_j$ within each segment are similar, forming a square block in the SSM. For repetitive segments with *sequence-wise similarity*, the first frame of a segment is similar to the first frame of the repetitive segment, and this applies to the second and subsequent frames, forming a diagonal stripe in the SSM.

2.2 RSSMs: Novel SSMs for Repetitive Segments

As part of the preparation for computing two RSSMs from the SSM, we denote a vertical column vector of the SSM by $\boldsymbol{s}_i$, with $\boldsymbol{s}_i = [SSM_{1,i}, SSM_{2,i}, \ldots, SSM_{T,i}]^\intercal$. The column vector $\boldsymbol{s}_i$ represents the similarities between the feature vector $\boldsymbol{f}_i$ and all the other feature vectors $\boldsymbol{f}_j$, $j \in 1..T$. Taking the eight repetitive segments (eight blue rectangles) in Figure 2, for example, the repetition is shown as eight peaks (high similarity values) in each column vector $\boldsymbol{s}_i$ within each segment (blue rectangle). In each segment, the eight peaks form the eight diagonal stripes. The column vectors $\boldsymbol{s}_i$ and $\boldsymbol{s}_j$ within the same segment are similar after matching the eight peaks by vertically shifting $\boldsymbol{s}_j$ by several elements (frames) upwards or downwards. The maximal similarity between adjacent vectors $\boldsymbol{s}_i$ and $\boldsymbol{s}_{i+1}$ within the same segment can be obtained by shifting all elements of $\boldsymbol{s}_{i+1}$ down by one element (i.e., shift index is -1), and also the maximal similarity between $\boldsymbol{s}_i$ and $\boldsymbol{s}_{i+2}$ can be obtained by shifting $\boldsymbol{s}_{i+2}$ down by two elements (i.e., shift index is -2).

ISBN 978-3-200-10642-0

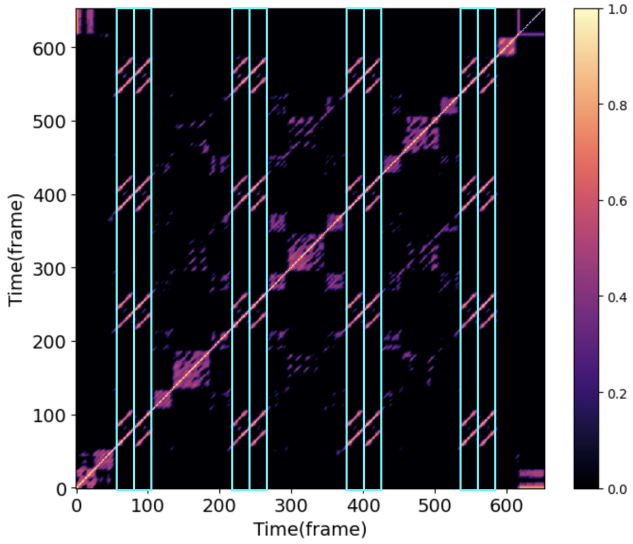


Figure 2: The standard SSM. Eight blue rectangles are added to highlight repetitive segments indicated by diagonal stripes. These repetitions are visible as eight high similarity values in each vertical column vector.

By leveraging them, each element of the two RSSMs, $\text{RSSM\_value}_{i,j}$ and $\text{RSSM\_index}_{i,j}$, is computed by using $\boldsymbol{s}_i$ and $\boldsymbol{s}_j$ (two column vectors of the standard SSM). By using the shift index m ($m \in [-T/2, T/2]$), we vertically shift the column vector $\boldsymbol{s}_j$ by m elements upwards (or downwards when $m < 0$) to generate $\boldsymbol{s}_j^m$, which contains T variations of $\boldsymbol{s}_j$. We then compute the cosine similarity between $\boldsymbol{s}_i$ and each of $\boldsymbol{s}_j^m$ to find the shift index m that gives the maximal similarity value. By using that maximal similarity value ($\max_m$) and its shift index (argmax_m), we define two matrices RSSM_value and RSSM_index as follows:

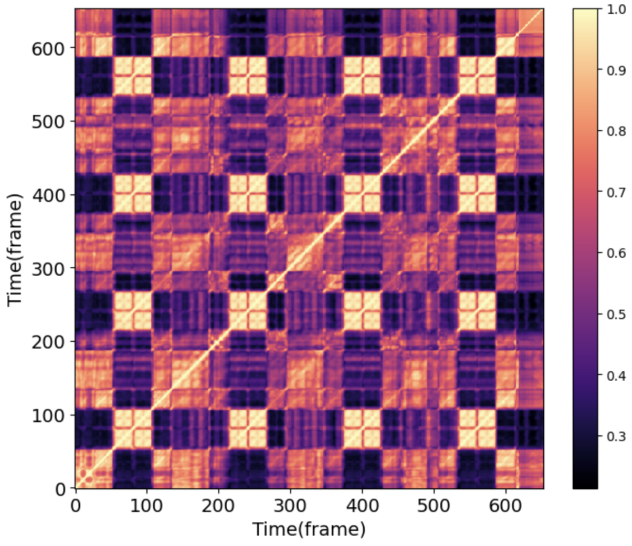
$$\text{RSSM\_value}_{i,j} = \max_m \cos(\boldsymbol{s}_i, \boldsymbol{s}_j^m), \quad (2)$$

$$\text{RSSM\_index}_{i,j} = 1 - |\operatorname{argmax}_m \cos(\boldsymbol{s}_i, \boldsymbol{s}_j^m)|/(T/2), \quad (3)$$

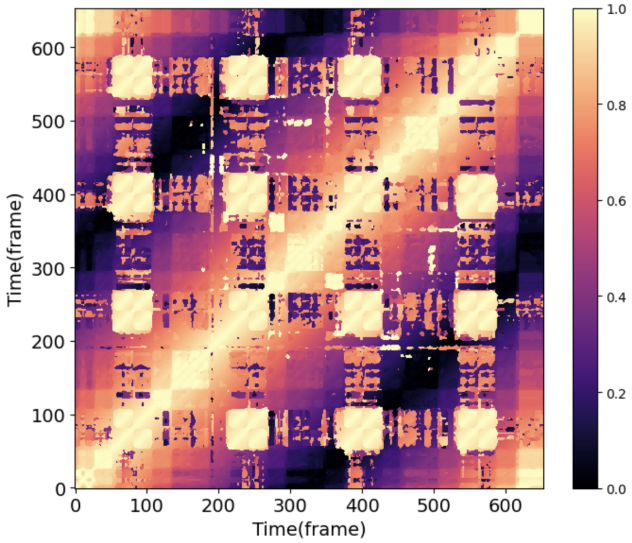
where $\text{RSSM\_value}_{i,j} \in [0, 1]$ and $\text{RSSM\_index}_{i,j} \in [0, 1]$ since the shift index is normalized to the range of 0 to 1. RSSM_index represents the proximity of m to 0.

The differences between the proposed RSSMs and traditional SSM is that in RSSMs, each element is computed based on the similarities between shifted SSM columns with a computational cost of $O(T^3)$; while in SSM or Self-Similarity Lag Matrix (SSLM) [30], each element is computed based on the similarities between spectral features with a computational cost of $O(T \times T)$ and $O(T \times L)$, where L is the lag time in the SSLM. Figure 3 shows the RSSMs corresponding to the standard SSM in Figure 2. The advantage of RSSM_value is that, as shown in Figure 3a, the repetitions represented by the diagonal stripes in the standard SSM are clearly indicated by square blocks in RSSM_value. The reason for the emergence of the square blocks is that if $\cos(\boldsymbol{s}_i, \boldsymbol{s}_{i+1}^{-1})$ is high, then $\cos(\boldsymbol{s}_{i+1}, \boldsymbol{s}_i^1)$ is also high. The same applies to $\cos(\boldsymbol{s}_i, \boldsymbol{s}_{i+2}^{-2})$, $\cos(\boldsymbol{s}_{i+1}, \boldsymbol{s}_{i+2}^{-1})$, and beyond.

However, the homogeneous segments shown as blocks in the standard SSM are blurred in RSSM_value. The advantage of RSSM_index is that, as shown in Figure 3b, both homogeneous and repetitive segments are indicated by the square blocks in RSSM_index. All pairs of $\boldsymbol{s}_i$ and $\boldsymbol{s}_j^0$ within each homogeneous segment are highly similar to each other due to the homogeneity of their audio feature vectors. So, the shift index m giving the maximal $\cos(\boldsymbol{s}_i, \boldsymbol{s}_j^m)$ must be 0, resulting in the high proximity of m to 0. It thus forms high values of a square block in RSSM_index. For each repetitive segment, also, the shift index m giving the maximal $\cos(\boldsymbol{s}_i, \boldsymbol{s}_j^m)$ also tends to be close to 0 due to its locality, resulting in a square block.



(a) RSSM_value (each element represents the maximal similarity value between column vectors of SSM after shifting up or down)



(b) RSSM_index (each element represents the proximity of the shift index (giving the maximal similarity value) to 0)

Figure 3: Proposed RSSMs.

Since the SSM and our original RSSMs have complementary advantages, we leverage all of them by stacking them together. Note that the size of the three matrices is the same.

2.3 CNN Model for Segment Boundary Detection

After stacking the standard SSM, RSSM_value, and RSSM_index to make the three-channel input (each chan-

ISBN 978-3-200-10642-0

	network name (the right is larger in size than the left)			
	Alex	115	117	1111
layer name	filter size	filter size	filter size	filter size
conv2d_0	11×11	11×11	11×11	11×11
conv2d_1	7×7	7×7	7×7	11×11
conv2d_2	5×5	5×5	7×7	7×7
conv2d_3	3×3	5×5	5×5	7×7
conv2d_4	3×3	5×5	5×5	5×5
conv2d_5	3×3	3×3	3×3	3×3

Table 1: Network architectures.

nel corresponds to one of the SSMs), we feed it to a CNN model. The CNN model is adapted from the Alexnet [31] [1], which consists of 6 convolutional layers with filter numbers of 96, 128, 256, 384, 384, and 256, respectively. We compare different filter sizes in the convolutional layers as shown in Table 1.

There is a pooling layer after each of the first 5 convolutional layers with a pooling size of (3, 1) and a stride of (2,1). The output of the 6^{th} convolutional layer has 4 dimensions: [*batch*, *row*, *time*, *channel*]. We add a pooling layer after the 6^{th} convolutional layer to pool along the *row* dimension and obtain an output with 3 dimensions: [*batch*, *time*, *channel*]. To find the best pooling method for the last pooling layer, we compare three different types: "mean" using an average pooling layer, "max" using a max pooling layer, and "both" using (stacking) both of them.

There are 4 fully-connected layers after the convolutional and pooling layers with the output dimensions of 1024, 1024, 256 and 1, respectively. All layers use the 'ReLU' activation functions, except that the last fully-connected layer uses a 'sigmoid' activation function with a dimension of 1 to predict the boundary probability at each time frame (every 0.5 s).

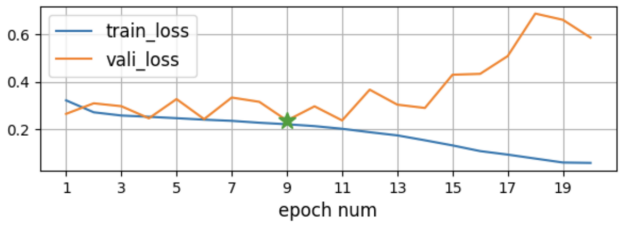
In Section 3.3, we report our ablation study to compare the above network configurations to find the best one.

2.4 Model Training and Fusion

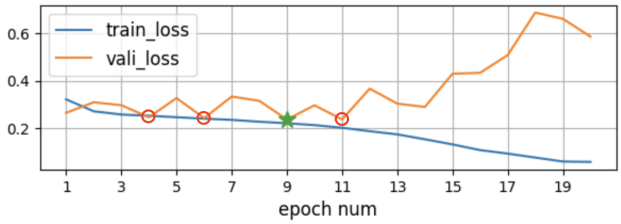
We train the CNN model with annotated data in a supervised manner using the Adam optimiser with a learning rate of 10^{-4}. In order to achieve efficient training with more balanced data, the ground-truth segment boundaries are extended by a range of 7 frames with a rectangular window after ignoring the first and last boundaries [10]. Since the input sizes vary depending on the length of the musical pieces, we use a batch size of 1 in the training.

It is common to select a model with the smallest validation loss to predict the output, as indicated by the star in Figure 4a. However, the final results also depend on the initialisation of the model training and the post-processing on the model output. To make the results more robust, we use a model fusion technique by averaging the weights of the models at several smallest validation losses (as in Figure 4b). This weight averaging can also improve the performance and generalisation of the model without additional computational complexity or training processes [32].

[1] This architecture is simpler than other transformer-based architectures in the recent literature. We chose it because we want to show that a simple CNN is sufficient to produce good results with the novel structure representation, RSSMs. Using more modern architectures may be of help and would be interesting to explore in the future.



(a) Choose a model with the smallest validation (vali) loss



(b) Average the weights of models at the four smallest vali losses

Figure 4: Model fusion by averaging model weights.

In our preliminary experiment, we found that averaging the models at the four smallest validation losses works best.

2.5 Post-Processing

We first apply a threshold to the model output, and then pick the peaks as the segment boundaries by using the python library `scipy.signal.find_peaks`. Because we ignore the first and last boundaries during the training, we append 0 s and the last frame (audio length) to the detected boundaries to obtain the final results.

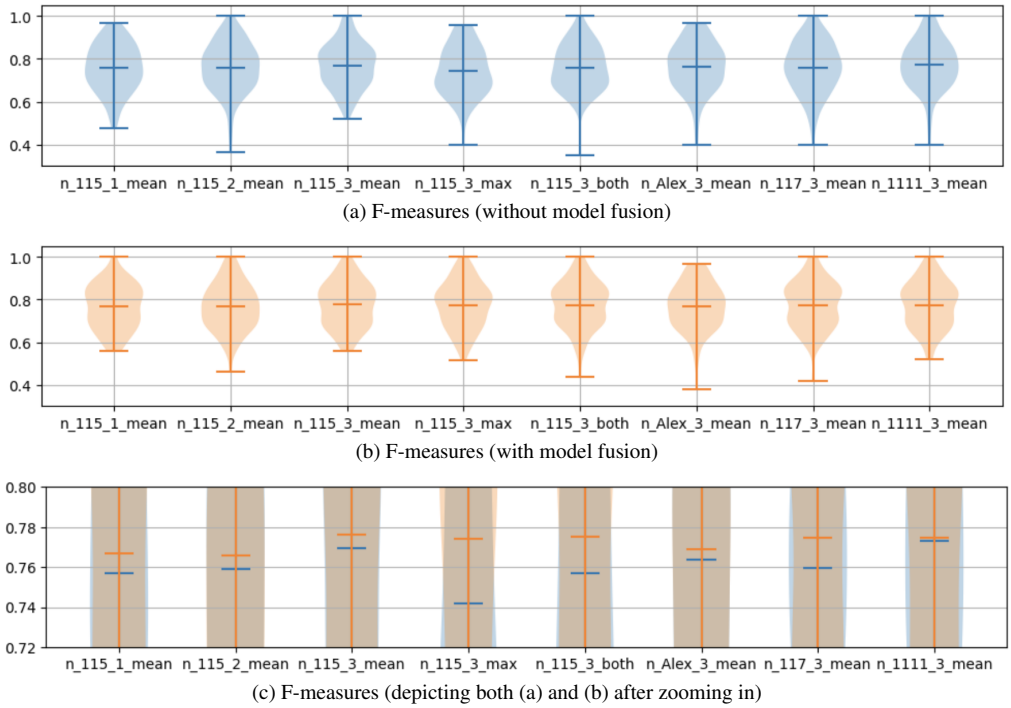
3. EXPERIMENTS

3.1 Datasets

For training, we divided the Harmonix dataset [22] into train, validation (vali), and test subsets (636, 184, and 92 songs, respectively). The model was trained with the Harmonix-train subset and validated with the Harmonix-vali subset. The ablation study was conducted on the Harmonix-test subset.

For testing, we used three datasets: **RWC_pop** (100 tracks of the RWC_pop [23] with AIST annotations [33]), **Beatles** (174 tracks of the Beatles [24] with TUT annotations [34]), and **SALAMI** [25]. For the SALAMI dataset, we considered the two annotations (An1, An2) and the two levels of flat annotations (Upper, Lower) corresponding to the files "textfile{1,2}_{upper,lowercase}.txt" in the annotations. The uppercase annotations ('Upper') are in a large scale (labelled with uppercase letters), mainly corresponding to the segment functions, such as 'intro', 'transition', and 'chorus'. The lowercase annotations ('Lower') are in a small scale (labelled with lowercase letters). We made three SALAMI subsets as in [10]: **SA_pop** (subset of SALAMI tracks with CLASS equal to Popular, with 237/148 songs for An1/An2), **SA_IA** (subset of SALAMI tracks with SOURCE equal to IA (Internet Archive), with 379/220 songs for An1/An2), and **SA_two** (subset of SALAMI tracks with at least two annotations, with 762/762 songs for An1/An2).

ISBN 978-3-200-10642-0



(a) F-measures (without model fusion)

(b) F-measures (with model fusion)

(c) F-measures (depicting both (a) and (b) after zooming in)

Figure 5: Ablation study on the Harmonix-test subset. 'n_X_Y_Z' indicates the network configuration, with 'X' referring to the network name (filter size) as in Table 1, 'Y' referring to the input channel number, and 'Z' referring to the type of the last pooling layer.

3.2 Evaluation Metrics

We evaluated the performance of segment boundary detection by using F-measure with a precision window of 3 s. We used the Python library `mir_eval` [35] with the function `mir_eval.segment.detection`.

3.3 Ablation Study (Finding the Best Configuration)

Figure 5 shows the violin plots for the ablation (comparative) study with the Harmonix-test subset. The label 'n_X_Y_Z' for each plot indicates the network configuration as explained in the figure caption.

First, we confirm the effectiveness of the model fusion. The short horizontal bars in the enlarged figure in Figure 5(c) indicate the average of the violin plots in (a) and (b). By comparing the two horizontal bars in each network configuration in (c), we can find that the orange averages ((b) with model fusion) are always higher than the blue averages ((a) without model fusion) for all the network configurations. This indicates that the use of the model fusion can improve the performance and provide more robust results. If we compare the violin plots between Figure 5 (a) and (b) for each configuration, the orange variances of (b) tend to be smaller than the blue variances of (a). So, the model fusion can help to achieve more stable results for different songs.

Then, we find the best network configuration with the model fusion by comparing the orange F-measure averages of eight different configurations (models) in Figure 5(c). We can see that model 'n_115_3_mean' with our 3-channel SSMs (SSM, RSSM_value, and RSSM_index) worked better than 'n_115_2_mean' with SSM and RSSM_value, confirming the effectiveness of using all the three SSMs. Since model 'n_115_2_mean' is not superior to 'n_115_1_mean' using SSM only, adding RSSM_value alone was not effective, but better generalization was obtained by adding RSSM_value in our preliminary experiments. Since the importance of using the 3 channels (all the three SSMs) is thus confirmed, we identify the best pooling method for the last pooling layer. By comparing 'n_115_3_mean' with 'n_115_3_max' and 'n_115_3_both', we decide to use "mean" (average pooling layer) since the max pooling did not help. Finally, we find the best filter size (network name in Table 1) by comparing four configurations 'n_*_3_mean'. We find that the F-measure was improved by increasing the size from 'n_Alex_3_mean' to 'n_115_3_mean', but no significant improvement observed by further increasing the size to 'n_117_3_mean' or 'n_1111_3_mean'.

With all the results considered, we conclude that the configuration 'n_115_3_mean' is the best and we call it the proposed model 'modelH.' We use 'modelH' for all other experiments below.

3.4 Results (Comparison with Existing Methods)

We evaluated the proposed 'modelH' on three test-only datasets and the results are shown in Table 2. Since the SALAMI dataset has two levels of ground-truth annotations, i.e., uppercase annotation with segments in large scale (Upper) and lowercase annotation with segments in small scale (Lower), we show results with three different thresholds, 0.0, 0.2, and 0.4, for the post-processing. It makes sense that the best results on the uppercase annotation was achieved with 'modelH_0.4' (with the largest

ISBN 978-3-200-10642-0

Method	RWC_pop	Beatles	SA_pop	SA_IA	SA_two	SALAMI Annotation
Proposed: modelH_0.0	0.650	0.592	0.554/0.549	0.443/0.434	0.453/0.462	
Proposed: modelH_0.2	**0.791**	0.736	0.683/0.668	0.588/0.593	0.577/0.588	Upper; An1/An2
Proposed: modelH_0.4	0.757	**0.756**	**0.695/0.672**	**0.600/0.615**	0.608/0.618	
Proposed: modelH_0.0			**0.732/0.762**	**0.692/0.719**	**0.710/0.715**	
Proposed: modelH_0.2			0.578/0.589	0.510/0.569	0.571/0.578	Lower; An1/An2
Proposed: modelH_0.4			0.498/0.490	0.371/0.429	0.453/0.459	
Serra [26]	**0.791**	0.752				
Buisson [9] †					**0.683**	Upper; An*
Grill [27] GS1	0.715				0.623	Upper; An*
McCallum [20] † Unsynch.		0.597		0.497		Not specified in [20]
Beat-synch.		0.648		0.535		
Salamon [28] $\text{DEF}_{\mu^H,\gamma^H}$					0.564	Upper; An*
Wang [17] scluster/D/eu/mul	0.653		0.623		0.553	Upper; An1+An2
Buisson [21] HE_0/HE_1	0.681	0.718			0.597/0.595	Upper; An1/An2
					0.611/0.600	Lower; An1/An2
Peeters [10] †	0.713		0.631/0.624	0.520/0.511	0.521/0.530	Upper; An1/An2
			0.570/0.610	0.547/0.612	0.589/0.589	Lower; An1/An2
Comparison: novelty curve from SSM	0.698	0.600				
Comparison: novelty curve from RSSM_value	0.761	0.688				
Comparison: novelty curve from RSSM_index	0.663	0.591				

Table 2: Results (F-measures) on three test-only datasets. The 'modelH_X' indicates the proposed modelH with a threshold of X in the post-processing step. † indicates the results obtained with the first and last boundaries trimmed. 'An1/An2' means that two results, R1/R2, were obtained separately using two different annotations An1 and An2: R1 from An1 and R2 from An2. 'An1+An2' means that the result was obtained by averaging over (An1+An2) annotated songs. 'An*' means that the annotation number is not specified in the reference.

threshold of 0.4), and the best results on the lowercase annotation was achieved with 'modelH_0.0'.

Table 2 also shows the performances reported for the eight existing methods [9,10,17,20,21,26–28] on the same test-only datasets. On the RWC_pop and Beatles datasets, we can see that our 'modelH' achieved start-of-the-art performances in comparison to the other methods. On the three SALAMI subsets, our 'modelH' similarly outperformed all other methods, except on the SALAMI_two with the uppercase annotation, where the best performance was achieved by the method Bussion [9]. We thus confirmed the effectiveness of the proposed method.

3.5 Comparison with Novelty Curve on RSSMs

Since the proposed RSSMs are so powerful as explained in Section 2.2 and demonstrated in the experimental results above, the simple traditional detection method based on the checkerboard kernel [1] without using the CNN model might be able to detect segment boundaries well if it is applied to the RSSMs. We therefore applied a Gaussian checkerboard kernel with a length of 30 frames to obtain the novelty curve by using an existing implementation. [2] We smoothed the novelty curve by using a moving average filter of 9 frames, and then picked the peaks from the smoothed novelty curve as boundaries.

At the bottom of Table 2 are the results of this detection on the RWC_pop and Beatles datasets when it was applied to each of the SSM, RSSM_value, and RSSM_index. Here, the novelty curve from the RSSM_value was the best. Surprisingly, its F-measure of 0.761 on the RWC_pop was better than those of several existing methods [3], showing the advantage of RSSM_value having clear square blocks. On the other hand, the results of RSSM_index were worse because its elements around the diagonal of the matrix naturally tend to have high values, as shown in Figure 3b, which is not suitable for the checkerboard kernel. However, it is certainly effective when used with the CNN-based model, evidenced by the comparison between 'n_115_2_mean' and 'n_115_3_mean' in the ablation study (Figure 5).

3.6 A case study

To better understand the proposed method, we further analyze an example of the input SSMs and the output boundary probability of the CNN model (modelH) in Figure 6. As can be seen in Figure 6, high peaks of the boundary probability correspond to the ground-truth segment boundaries. Since the boundary probability is predicted from the input SSMs, if there are clear clues for boundaries, such as edges of square blocks, in some SSMs, it is easier to detect them, as shown by high peaks above 0.5 in Figure 6b. If such clear clues cannot be seen, it is more difficult to detect, as shown by peak "C" in SSM (Figure 6a top). We expect that if future work could add different SSMs based on other features, clearer clues could be obtained even for such boundaries. Other difficult cases are the peaks "A" and "B". Although their heights are similar, "A" is a false positive error, while "B" is the correct boundary. The ambiguity between these peaks is not due to the input SSMs and could be avoided by enforcing regularity on segment duration [6, 36].

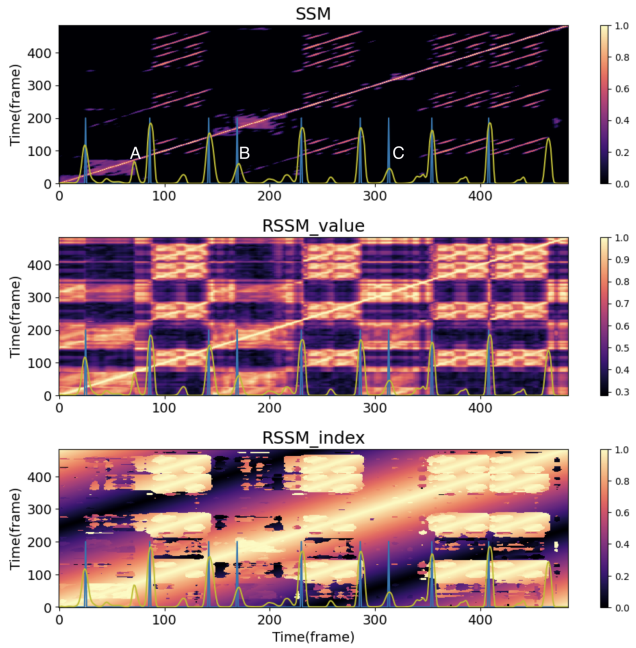
4. CONCLUSIONS

In this paper, we proposed the two novel RSSMs to highlight repetitive segments for the segment boundary detection task. The contributions of this paper can be summarized as follows. First, although the key idea behind RSSMs is simple, to the best of our knowledge, this is the first work to propose them. Second, we confirmed that

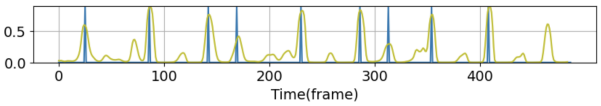
[2] https://www.audiolabs-erlangen.de/resources/MIR/FMP/C4/C4S4_NoveltySegmentation.html

[3] Please note that in Serra [26] the novelty curve is calculated based on the difference between successive structural features, which is different from the checkerboard method used here.

ISBN 978-3-200-10642-0



(a) Input (three SSMs) and output (boundary probability) of CNN



(b) The boundary probability and the ground truth

Figure 6: Example of the input and output of the CNN model 'modelH' (for a piece from the Harmonix-test subset). The green curves overlaid at the bottom of each SSM/RSSM in (a) and the curve in (b) are all the same and represent the boundary probability. The vertical blue lines indicate the ground-truth segment boundaries.

the proposed RSSMs help to improve the performance in the ablation study. Third, we showed that the proposed method based on RSSMs and CNN can achieve better F-measures than the state-of-the-art methods on the three test-only datasets, which have been conventionally used for the evaluation of this task.

As we showed in Section 3.6, detecting a boundary becomes difficult when there are no clear clues in the SSMs. In the future, we could address this issue by integrating SSMs computed from different features such as chroma features and learned features from deep models. In addition, since the RSSMs are so effective in the boundary detection task, our future work will also include a verification of their effectiveness in the segment labelling task [10, 17, 19, 21, 26, 28].

Acknowledgments

This work was supported in part by JST CREST Grant Number JPMJCR20D4, Japan.

5. REFERENCES

[1] J. Foote, "Automatic audio segmentation using a measure of audio novelty," in *Proc. of the IEEE Int. Conf. on Multimedia and Expo (ICME)*, 2000, pp. 452–455.

[2] M. Goto, "A chorus-section detecting method for musical audio signals," in *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2003, pp. 437–440.

[3] R. B. Dannenberg and M. Goto, *Music Structure Analysis from Acoustic Signals*. New York, NY: Springer New York, 2008, pp. 305–331.

[4] J. P. Bello, "Measuring structural similarity in music," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 7, pp. 2013–2025, 2011.

[5] O. Nieto and J. P. . Bello, "Systematic exploration of computational music structure research," in *Proc. of the 17th Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2016.

[6] O. Nieto, G. J. Mysore, C. Wang, J. B. L. Smith, J. Schlüter, T. Grill, and B. McFee, "Audio-based music structure analysis: Current trends, open challenges, and applications," *Transactions of the International Society for Music Information Retrieval*, vol. 3, no. 1, 2020.

[7] J.-C. Wang, Y.-N. Hung, and J. B. L. Smith, "To catch a chorus, verse, intro, or anything else: Analyzing a song with structural functions," in *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2022.

[8] A. Marmoret, J. E. Cohen, and F. Bimbot, "Barwise music structure analysis with the correlation block-matching segmentation algorithm," *Transactions of the International Society for Music Information Retrieval*, vol. 6, no. 1, pp. 167–185, 2023.

[9] M. Buisson, B. McFee, S. Essid, and H. C. Crayencour, "A repetition-based triplet mining approach for music segmentation," in *Proc. of the 24th Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2023, pp. 417–424.

[10] G. Peeters, "Self-similarity-based and novelty-based loss for music structure analysis," in *Proc. of the 24th Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2023, pp. 749–756.

[11] M. Fuentes, B. McFee, H. C. Crayencour, S. Essid, and J. P. Bello, "A music structure informed downbeat tracking system using skip-chain conditional random fields and deep learning," in *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.

[12] T. Kim and J. Nam, "All-in-one metrical and functional structure analysis with neighborhood attentions on demixed audio," in *Proc. of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2023.

[13] M. Mauch, K. C. Noland, and S. Dixon, "Using musical structure to enhance automatic chord transcription," in *Proc. of the 10th Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2009.

ISBN 978-3-200-10642-0

[14] S. Dai, Z. Jin, C. Gomes, and R. B. Dannenberg, "Controllable deep melody generation via hierarchical music structure representation," in *Proc. of the 22th Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2021, pp. 417–424.

[15] M. Agarwal, C. Wang, and G. Richard, "Structure-informed positional encoding for music generation," in *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2024.

[16] H. Grohganz, M. Clausen, N. Jiang, and M. Müller, "Converting path structures into block structures using eigenvalue decompositions of self-similarity matrices," in *Proc. of the 14th Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2013, pp. 209–214.

[17] J. C. Wang, J. B. L. Smith, W. T. Lu, and X. Song, "Supervised metric learning for music structure features," in *Proc. of the 22th Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2021.

[18] B. McFee and D. P. W. Ellis, "Analyzing song structure with spectral clustering," in *Proc. of the 15th Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2014.

[19] T. Cheng, J. B. L. Smith, and M. Goto, "Music structure boundary detection and labelling by a deconvolution of path-enhanced self-similarity matrix," in *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 106–110.

[20] M. C. McCallum, "Unsupervised learning of deep features for music segmentation," in *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.

[21] M. Buisson, B. McFee, S. Essid, and H. C. Crayencour, "Learning multi-level representations for hierarchical music structure analysis," in *Proc. of the 23th Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2022.

[22] O. Nieto, M. McCallum, M. E. P. Davies, A. Robertson, A. Stark, and E. Egozy, "The Harmonix Set: Beats, downbeats, and functional segment annotations of western popular music," in *Proc. of the 20th Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2019.

[23] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, "RWC Music Database: Music genre database and musical instrument sound database," in *Proc. of the 4th Int. Conf. on Music Information Retrieval (ISMIR)*, 2003, pp. 229–230.

[24] M. Mauch, C. Cannam, M. Davies, S. Dixon, C. Harte, S. Kolozali, D. Tidhar, and M. Sandler, "OMRAS2 Metadata Project 2009," in *Proc. of the 10th Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2009.

[25] J. B. L. Smith, J. Burgoyne, I. Fujinaga, D. Roure, and J. S. Downie, "Design and creation of a large-scale database of structural annotations," in *Proc. of the 12th Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2011.

[26] J. Serra, M. Müller, P. Grosche, and J. L. Arcos, "Unsupervised detection of music boundaries by time series structure features," in *Proc. of the 26th AAAI Conf. on Artificial Intelligence*, 2012, pp. 1613–1619.

[27] T. Grill and J. Schlüter, "Music boundary detection using neural networks on combined features and two-level annotations," in *Proc. of the 16th Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2015.

[28] J. Salamon, O. Nieto, and N. J. Bryan, "Deep embeddings and section fusion improve music segmentation," in *Proc. of the 22th Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2021.

[29] 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 *Proc. of 14th Python in Science Conf.*, vol. 8, 2015.

[30] T. Grill and J. Schlüter, "Music boundary detection using neural networks on spectrograms and self-similarity lag matrices," in *Proc. of the 23rd European Signal Processing Conf. (EUSIPCO)*, 2015, p. 1296–1300.

[31] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, vol. 25, 2012.

[32] W. Li, Y. Peng, M. Zhang, L. Ding, H. Hu, and L. Shen, "Deep model fusion: A survey," *arXiv preprint arXiv:2309.15698*, 2023.

[33] M. Goto, "AIST Annotation for the RWC Music Database," in *Proc. of the 7th Int. Conf. on Music Information Retrieval (ISMIR)*, 2006, pp. 359–360.

[34] J. Paulus, "Improving Markov model based music piece structure labelling with acoustic information," in *Proc. of the 11th Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2010, pp. 303–308.

[35] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. P. W. Ellis, "mir_eval: A transparent implementation of common MIR metrics," in *Proc. of the 15th Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2014.

[36] G. Sargent, F. Bimbot, and E. Vincent, "A regularity-constrained viterbi algorithm and its application to the structural segmentation of songs," in *Proc. of the 12th Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2011.

ISBN 978-3-200-10642-0