International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:4, No:1, 2010

MTSSM - A Framework for Multi-Track Segmentation of Symbolic Music

Brigitte Rafael and Stefan M. Oertl

***Abstract*—Music segmentation is a key issue in music information retrieval (MIR) as it provides an insight into the internal structure of a composition. Structural information about a composition can improve several tasks related to MIR such as searching and browsing large music collections, visualizing musical structure, lyric alignment, and music summarization. The authors of this paper present the MTSSM framework, a two-layer framework for the multi-track segmentation of symbolic music. The strength of this framework lies in the combination of existing methods for local track segmentation and the application of global structure information spanning via multiple tracks. The first layer of the MTSSM uses various string matching techniques to detect the best candidate segmentations for each track of a multi-track composition independently. The second layer combines all single track results and determines the best segmentation for each track in respect to the global structure of the composition.**

***Index Terms*—Pattern Recognition, Music Information Retrieval, Machine Learning.**

I. Introduction

DURING the last years plenty of research has been done in the field of music information retrieval (MIR), also including various aspects of music segmentation [1], [2]. Music segmentation targets at the identification of boundaries between structurally relevant parts of a composition to enable or improve MIR-related tasks such as effective searching and browsing within large music collections [3] as well as the compression, classification and visualisation of music [4], music summarization [5], lyric alignment [6], and the development of recommendation systems. Current approaches are based on a similarity matrix [7], [8], [9], hidden Markov models [1], or the application of the shortest path algorithm [2]. The most common approach aims at detecting structure boundaries with the aid of a novelty score which is described in [6] and [7]. Methods using that score are limited to compositions following certain rules and principles as they require the existence of domain knowledge (extraopus). To the contrary, analyzing music based on its self-similarity is not dependent on this kind of a priori knowledge but focuses on the information provided within the piece itself (intraopus). In consequence, methods using self-similarity can be applied to a broader musical spectrum and are therefore chosen for the MTSSM. The local segmentation methods of the MTSSM make use of repetitions to detect segments and then cluster similar segments to create segment groups (i.e., collections of several nonoverlapping segments that fulfill a given similarity condition). However, as the MTSSM is a modular framework, it can be extended to apply methods based on novelty scores as well.

Music audio formats like MP3, WAV, or MPEG dominate digital music collections, therefore creating a focus on audio data for most past and current projects. There are, however, also music formats representing music in symbolic form. The most popular one is the MIDI (Musical Instrument Digital Interface) format which is used in the MTSSM framework. An advantage of MIDI compared to digital audio is the availability of separate tracks. Instead of having all instruments merged into one common data stream, the information in MIDI files is separated into single tracks. Nevertheless current approaches in MIR working on symbolic music do not take advantage of the single track data but still merge all data and try to achieve one global segmentation for the whole composition. There are, however, compositions where local track structures differ from each other (because of polyrhythmic tracks or time offsets between local patterns). Achieving local segmentations for each track can provide additional information about the global structure of the composition. If one global segmentation is needed, the local segmentations can still be weighted and summed up to form a global segmentation.

In this paper the authors present a framework that combines single track segmentations to find the best segmentation for each track in respect to the global structure of a multi-track composition. Following the introduction, the second section gives an overview of the architecture of the MTSSM framework. The third section introduces the reader to the application of string matching techniques for pattern matching in symbolic music data since they are used for the segmentation of single tracks. The fourth section demonstrates the combination of single track segmentations to achieve a multi-track segmentation for each track. An outlook on future work concludes the paper.

II. MTSSM

The MTSSM is a modular framework that allows the application of existing segmentation methods for the segmentation of single tracks and includes their global combination to provide information about the geometrical structure of a composition. The algorithm of the framework is not limited to one local segmentation method but offers the possibility to apply various methods to each track.

Fig. 1 shows the architecture of the MTSSM. The first layer of the framework creates several candidate segmentations for each track by applying local segmentation methods. All results are evaluated to produce a local score for each candidate segmentation. The second layer of the MTSSM performs a

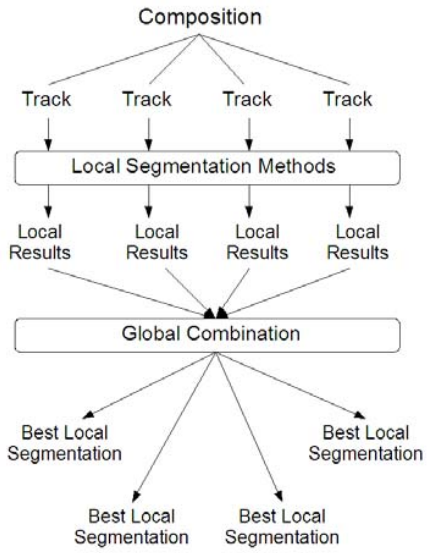

Fig. 1. Structure of the MTSSM-framework

pairwise comparison of all local results belonging to different tracks and calculates a global score for each local segmentation based on the correlation to segmentations of other tracks. As a final step the algorithm chooses the best segmentation for each track resulting from a combination of the local and global score. The following sections give a more detailed description of the MTSSM's components.

III. Local Segmentation Methods

MIDI data provides exact information about pitches and rhythms of a composition. Consequently, each music sequence can be expressed as one or several strings allowing the application of string matching techniques for pattern recognition [10], [11], [12], [13]. Since string matching techniques are very common in MIR, they have also been chosen for the local segmentation component of the MTSSM. Fig. 2 illustrates the architecture of this component. An arbitrary number of methods can be applied to each track to produce its local candidate segmentations. For the first version of the MTSSM the authors chose to apply one exact string matching method using a correlative matrix and one approximate string matching method using dynamic programming. Both methods are described in section III-A and III-B, respectively.

There are several possibilities to transform a musical sequence into a string. Most authors choose a numerical representation since using letters leaves room for ambiguities (e.g., `C#` and `Db` could be seen as different symbols although representing the same pitch enharmonically) and does not provide information about the current octave. Expressing pitches with their MIDI pitch values (e.g., 60 for Middle C) avoids ambiguities. To get an octave-invariant representation the modulo operator can be applied to all pitches.

String matching algorithms are available in a broad range for analyzing music data represented by strings. Charras et al.

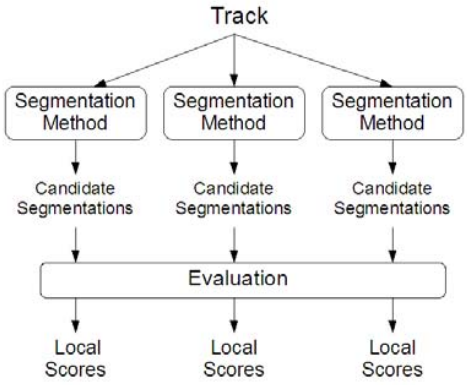

Fig. 2. Local Segmentation of each track

[14] give a good introduction to several of them. Common approaches for exact matching are described in [15], [16], [17]. Exact matching is limited to the detection of exact repetitions. In many compositions approximate patterns are more frequent than exact ones. Their detection is also important for most applications so the focus in music information retrieval has switched to approximate string matching [18]. Some approaches combine exact and approximate string matching in an effort to benefit from the advantages of both methods [17]. The MTSSM framework uses both exact and approximate string matching, taking advantage of the details given by the MIDI representation.

MIDI data offers a wide variety of attributes to be extracted from a music sequence. Some basic features include exact pitch values, relative and absolute onset and offset times of notes, and velocity values. From these features many more note characteristics can be derived: pitch intervals, pitch contour, relative and absolut durations of notes, polyphonic notes, positions and durations of rests, etc. All these features can be used for pattern matching. However, as some attributes (e.g., pitch, onset and offset times) can be expressed in different ways, the best representation–or a combination of representations–has to be selected for each feature. The authors of this paper have chosen to use a combination of exact pitch values, pitch intervals, pitch contour, and note durations measured in beats (accepting a tolerance interval for the comparison of durations).

Fig. 3 illustrates an extract from the notes of one track. The upper part shows a graphical representation of notes as it might be familiar to the reader. The lower part is similar to the representation above. It also contains five staff lines and an additional line to indicate Middle C. Notes are displayed as boxes. Box widths depend on the durations of the corresponding notes. Lines above notes indicate an increment of the pitch value by one semitone. Rests are represented as lighter boxes. Vertical lines correspond to the vertical lines in the upper picture and represent bar changes. Rectangles below notes symbolize a sample segmentation. Segments of the same shade of gray belong to the same segment groups.

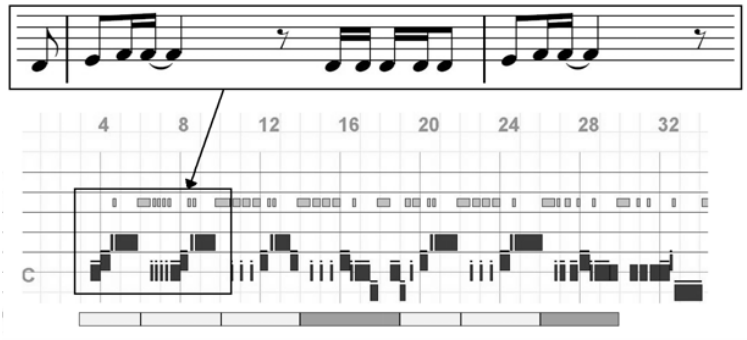

Fig. 3. Graphical representation of a MIDI track

	C	G	E	G	C	G	E	D	G	C	G	D	G	C
C	-	0	0	0	1	0	0	0	0	1	0	0	0	1
G		-	0	1	0	2	0	0	1	0	2	0	1	0
E			-	0	0	0	3	0	0	0	0	0	0	0
G				-	0	1	0	0	1	0	1	0	1	0
C					-	0	0	0	0	2	0	0	0	2
G						-	0	0	1	0	3	0	1	0
E							-	0	0	0	0	0	0	0
D								-	0	0	0	1	0	0
G									-	0	1	0	2	0
C										-	0	0	0	3
G											-	0	1	0
E												-	0	0
G													-	0
C														-

	C	G	E	G	C	G	E	D	G	C	G	D	G	C
C	-	0	0	0	1	0	0	0	0	1	0	0	0	1
G		-	1	0	0	2	1	0	1	0	2	1	1	0
E			-	0	0	1	3	0	0	0	1	3	0	0
G				-	1	1	0	0	1	1	1	0	4	1
C					-	0	0	0	1	2	0	0	1	5
G						-	1	0	1	0	3	1	1	0
E							-	0	0	0	0	0	0	0
D								-	0	0	0	2	0	0
G									-	1	1	0	1	1
C										-	0	0	1	2
G											-		1	0
E												-	0	0
G													-	1
C														-

Fig. 4. Correlative matrix–standard and parametrized version

A. Parameterized Exact String Matching

The exact string matching part is based on the repetition of exact sequences. A common method to find repeating patterns is the similarity matrix where each note or time frame is compared to all other notes or time frames of the composition, respectively. In a note-based matrix each cell $[i, j]$ contains a value representing the similarity between the ith and the jth note.

Hsu et al. [19] describe how to employ a correlative matrix to find all exact patterns within a musical sequence. The correlative matrix builds up patterns by considering the previous cells. If the ith and the jth notes have the same pitch and $i, j > 0$ then the new value for cell $[i, j]$ results from the value of cell $[i - 1, j - 1] + 1$. If the pitches are identical but $i = 0$ or $j = 0$ the value is set to 1. In the end the value in each cell gives the longest possible pattern ending at the corresponding note. The left table in Fig. 4 gives the correlative matrix for the music sequence of Fig. 5. The gray boxes indicate that there are three possible patterns of length 3 with two occurences each.

The authors use a parameterized version of the correlative matrix. Whereas in [19] patterns only consist of notes with identical pitch values, the authors' approach also accepts notes with corresponding pitch intervals or a similar combination of pitch contour and note durations. As a result, the algorithm detects transpositional invariant patterns, as well. In the parameterized correlative matrix the value of a cell $[i, j]$ is increased if the similarity between the ith and the jth note (calculated from pitch, contour and duration values) exceeds the similarity threshold. The right table in Fig. 4 gives the parametrized correlative matrix for the music sequence of Fig. 5. The gray boxes indicate that there is one possible pattern of length 3 with three and one with two occurences as well as a pattern of length 5 with two occurences.

Derived from the result of the correlative matrix the algorithm creates pattern groups by clustering identical patterns. These pattern groups are evaluated and the best pattern group is chosen. The quality of a pattern group depends on its size (= the number of patterns in the group) and its pattern length (= the number of notes within one pattern). Patterns must not overlap so all overlapping patterns are removed from the other pattern groups. The algorithm repeats these evaluation, selection, and deletion steps until no more pattern groups are left. The result of the parameterized exact matching is a list of pattern groups that form a nonoverlapping segmentation of the composition. The patterns within the groups can be

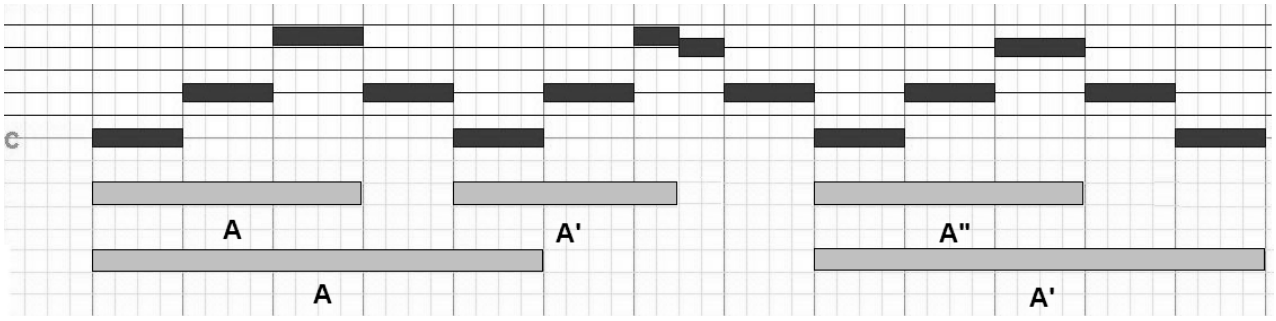

Fig. 5. Segmentation with exact matching

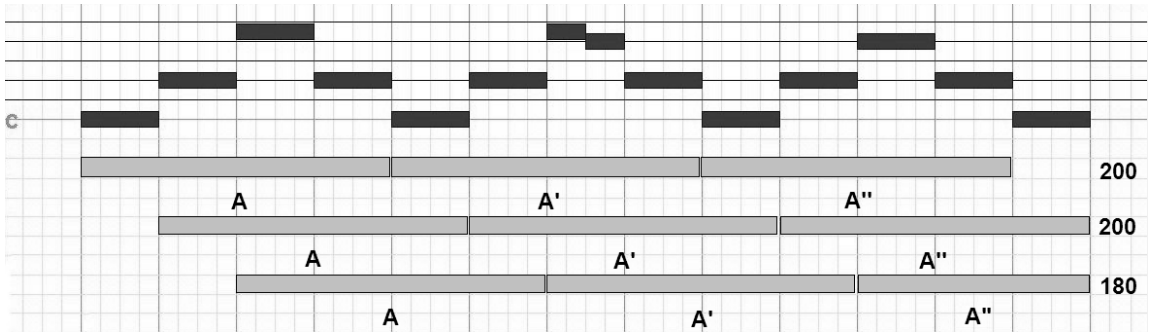

Fig. 6. Segmentation with approximate matching

further processed (e.g., combined or extended) to increase the covering range of the segmentation. Fig. 5 shows some possible pattern groups found by the exact string matching algorithm. Patterns labeled with the same letter belong to the same pattern group. Apostrophes indicate that patterns within a group are not identical but just similar (like A, A', and A").

B. Approximate String Matching

The exact string matching algorithm achieves good results for structured tracks with a high number of (almost) identical repetitions of a pattern. Its weakness is the incapacity to detect pattern repetitons that do not have the same number of notes as notes have been inserted into or deleted from any of the pattern occurences. Therefore, the authors also use an approximate string matching algorithm for the segmentation of tracks with less similar repetitions. The approximate string matching algorithm borrows the concept of dynamic programming from the field of bioinformatics which has already been introduced to MIR [20]. Dynamic programming targets at the alignment of two musical sequences, thus allowing insertion, deletion, and replacement of notes. It uses a matrix to assign scores to matches or mismatches between pairs of notes, and a gap penalty for matching a note in one sequence to a gap in the other (which is the result of deletion and insertion). A traceback algorithm then finds the best alignment with the highest score.

The approximate string matching algorithm performs a fragmentation of a track into several segments of the same beat length. It compares all possible segment pairs by means of dynamic programming and calculates similarity scores in pairs. If the similarity score of two segments exceeds a certain threshold, the segments are assigned to the same segment group. As a result the algorithm creates a list of segment groups with nonoverlapping segments for each fragmentation. This process is repeated for various segment lengths, shifting the start position forward to each possible beat. Segment lengths range from one beat up to half the duration of the composition. In the end the algorithm evaluates each list and calculates a total score for each potential segmentation to find the optimal segmentation for the track. Fig. 6 provides an example for several results for one track achieved by the approximate string matching algorithm plus their total scores.

C. Evaluation of Local Segmentation Results

The evaluation of the local segmentation results retrieved from the local segmentation methods in the previous step is crucial to the success of the algorithm. The evaluation method has to determine the quality of each local candidate segmentation. As the reader can see in Fig. 1, the evaluation component of the MTSSM is modular and independent of the other components so it can be exchanged or extended for experimentation with various evaluation methods.

The evaluation procedure has to assess several criteria to calculate the local score of a segmentation. An example for such an equation to calculate the local score could look like this:

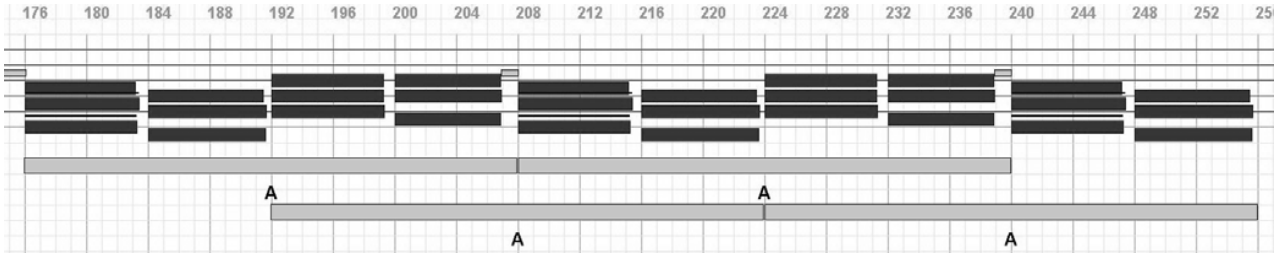

Fig. 7. Multiple segmentations for one track

$$
\begin{aligned}
score = & \sum_{g \in G} |seg_{ident}(g)| * w_{ident} \\
& + \sum_{g \in G} |seg_{sim}(g)| * w_{sim} \\
& + |seg_{notes}| * w_{notes} \\
& + |seg_{bars}| * w_{bars} \\
& + |seg_{reg}| * w_{reg} \\
& + \sum_{g \in G} |seg_{duration}(g)| * w_{duration} \\
& - |groups_{duration}| * w_{duration_{group}} \\
& - |seg_{clip}| * w_{clip} \\
& - |seg_{short}| * w_{short} \\
& - \sum_{g \in G} |diversity(g)| * w_{div}
\end{aligned}
\tag{1}
$$

Variables used in the evaluation function are explained in Table I.

w_{ident}, w_{sim}, w_{notes}, w_{bars}, w_{reg}, $w_{duration}$, $w_{duration_{group}}$, w_{clip}, w_{short}, and w_{div} are parameters for weighting the factors of the evaluation function. They have been tested and fine-tuned by composers and music experts. The evaluation concludes the first layer of the MTSSM resulting in a list of candidate segmentations for each track along with their local scores.

IV. Global Combination of Single Track Segmentations

The result of the first layer of the MTSSM is a list of multiple candidate segmentations for each track. The segmentation with the highest score is the optimal local segmentation for the respective track. From a global view, however, also candidates having a lower score might turn out as the best segmentation if they fit into the global structure of the composition better than the best single track segmentation. If more than one candidate share the highest score like in the example in Fig. 7, one of them can be chosen using global structure information retrieved by the candidate segmentations of other tracks. Fig. 7 shows a track with two potential segmentations that contain the same number of identical segments of the same length so their evaluation results in the same single track score.

The second layer of the MTSSM (displayed in Fig. 8) calculates a global (geometric) score for each candidate segmentation of each track by comparing it to all candidate segmentations of all other tracks. Given a composition with three tracks, A, B, and C, each list of track A is compared to all lists of B as well as to all lists of C. For each pair of lists the geometric score is calculated and assigned to the respective lists.

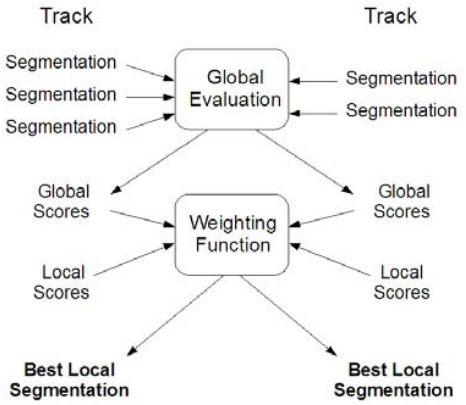

Fig. 8. Global combination of local segmentations

The global evaluation component searches for corresponding segment groups between two lists. The score increases with the number of segments of a segment group in the first list that occur at the same positions as segments of one segment group in the second list. The longer the common regions of such segments, the better the score. An even higher score is reached if two segments have the same starting and ending points. Fig. 9 gives three examples for corresponding segment groups. Rectancles around segments indicate pairs of segments of different tracks that occur at the same time. The lists of the first example have a high score already since all segments of the segment group in list A have corresponding segments of one segment group in list B. The second example results in an even higher geometric score since the overlapping regions have the same durations as in the first example and the segments of lists A and B also share the same ending points. The highest geometric score is computed for the last example. Although segments themselves are shorther and yield lower local scores for lists A and B, the durations of the overlapping regions are the same as in the other examples and starting as well as ending points are shared by all segment pairs resulting in a high geometric score.

TABLE I
VARIABLES OF THE EVALUATION FUNCTION

Variable name	Variable description
G	set of all segment groups (each group is a set itself, containing all segments within the group)
S	set of all segments of the segmentation
N	set of all notes of the track
$seg_{ident}(g)$	set of identical segments in segment group g $s \in g \vert \exists s' \in g : s = s'$
$seg_{sim}(g)$	set of similar segments in segment group g $s \in g \vert \exists s' \in g : s != s' \wedge sim(s,s') >= threshold_{sim}$
$sim(s,s')$	calculates the similarity value between two segments
$threshold_{sim}$	similarity threshold: segments with a similarity value above the threshold are similar
seg_{notes}	set of segments where start points coincide with note starts: $s \in S \vert \exists n \in N : start(n) = start(s)$
seg_{bars}	set of segments that start at the first beat of a bar: $s \in S \vert start(s) \mod beatsPerBar = 0$
seg_{reg}	set of segments with regular distances to other segments: $s2 \in S \vert start(s2) - start(s1) = meanDistance$ where $s1$ and $s2$ are succeeding segments and $meanDistance$ is the mean distance between succeding segments
$seg_{duration}(g)$	set of segments with the mean segment duration of g: $s \in g \vert duration(s) = meanDuration(g)$
$meanDuration(g)$	mean duration of all segments in segment group g
$groups_{duration}$	set of segment groups that have a different mean segment duration than the other segment groups $g \in G \vert meanDuration(g) != meanDuration(G)$
seg_{clip}	set of segments where segment boundaries clip notes $s \in S \vert \exists n \in N : (start(n) < start(s) \wedge end(n) > start(s)) \vee (start(n) < end(s) \wedge end(n) > end(s))$
seg_{short}	set of segments that do not contain enough notes to be valid: $s \in S \vert \vert notes(s) \vert < minNotesForSegment$
$diversity(g)$	diversity between the segments of segment group g: $\sum_{s \in g} \vert seg_{div} \vert / \vert g \vert$
$seg_{div}(s,g)$	number of segments in g that are not identical to s: $s \in g \vert s! = s'$

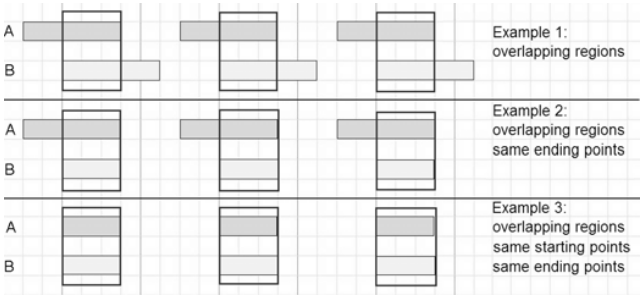

Fig. 9. Geometric comparison of segmentations

Fig. 10 shows extracts of sample segmentations for two tracks. After the removal of bad segmentations in the filtering process, the first track has just one potential segmentation. The two candidate segmentations of the second track (the same as in Fig. 7) have the same local score but the first list has a higher global score and is therefore chosen as the better one. Whereas in this example there is only one solution for one of the tracks and also the other one has two lists only, in general segmentation problems are more complex. Compositions on average consist of 6 to 10 tracks with each track comprising up to 10 or more candidate solutions. Still the MTSSM finds the best solution for each track by weighting each candidate solution by its local and global score.

With a set of the best geometric segmentations a hierarchical structure can be created for each track using top-down and bottom-up methods. A hierarchical structure provides structural information on various granularity levels. The upper hierarchy levels contain longer segments (e.g., verse and chorus) while the segments on lower levels are reduced to rather short repeating patterns. Geometrical comparison is used here as well to get the best segment structures on each hierarchy level. Fig. 11 shows a sample of a hierarchical structure for the first track of Fig. 10. Again, it is only a very simple structure as the hierarchical structure for a track can consist of up to five hierarchy levels or more.

V. CONCLUSION

The MTSSM framework is a modular framework for the multi-track segmentation of symbolic music. Due to its modularity it allows the application of various existing segmentation methods for local track segmentation as well as the replacement of the local and global evaluation components and the weighting function, resulting in a flexible test environment. The main strength of the MTSSM is the incorporation of global structure information spanning over multiple tracks. As a result, new information about the structure of a composition is obtained.

The framework presented in this paper already achieves good results for a large pool of test compositions. Considering both exact and approximate string matching techniques, segmentations for tracks with various characteristics can be found. For tracks allowing multiple segmentations with similar scores, the framework finds the best solution in respect to the geometry of the composition.

An advantage of the framework presented in this paper is the embedding of each track's segmentation into the global structure of the composition. However, if a track does not fit into the global structure of a composition (e.g., if it is polyrhythmic), its local segmentation is preserved and it is not adapted to the global structure. The final score is a combination of local and global score, thus paying respect to strong local scores. As a consequence, the result of the framework is a segmentation for each track instead of one segmentation for the whole

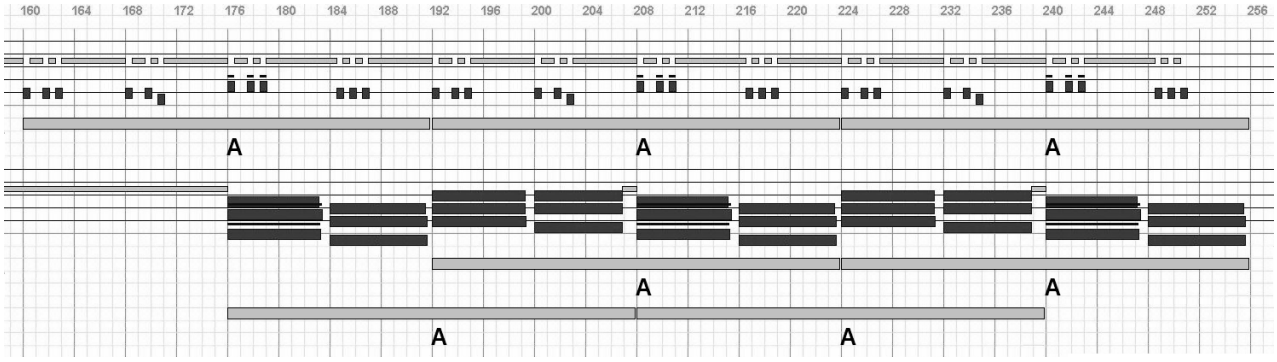

Fig. 10. Multi-track combination of segmentations

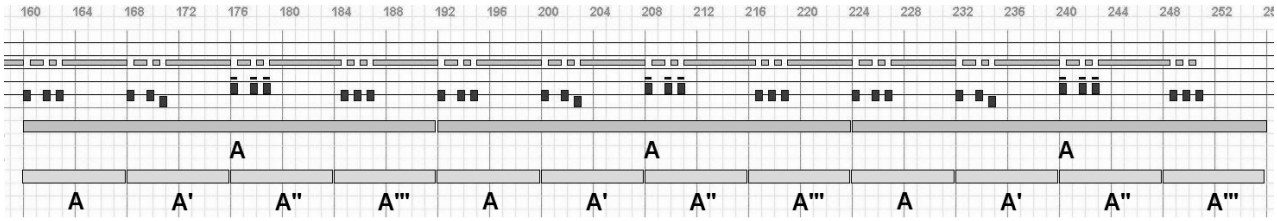

Fig. 11. Hierarchical segment structure

composition. Its results can therefore not be compared to existing results as the latter consider segmentations of whole compositions only. All results of the framework have been validated by professional composers and musicologists.

Future work will concentrate on optimizing the first layer of the approach to decrease the runtime of the algorithm. The concept of dynamic programming described in section III-B is very powerful but also time and memory consuming. Other approximate string matching techniques have to be tested to get the best trade-off between runtime and segmentation quality.

REFERENCES

[1] M. Levy, K. Noland, and M. Sandler, "A comparison of timbral and harmonic music segmentation algorithms," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 4, 2007, pp. 1433–1436.

[2] K. Jensen, "Multiple scale music segmentation using rhythm, timbre, and harmony," *EURASIP Journal on Applied Signal Processing*, vol. 2007, no. 1, 2007.

[3] S. Downie and M. Nelson, "Evaluation of a simple and effective music information retrieval method," in *Proceedings of the ACM International Conference on Research and Development in Information Retrieval (SIGIR),*, 2000, pp. 73–80.

[4] E. Isaacson, "What you see is what you get: On visualizing music," in *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, 2005, pp. 389–395.

[5] M. Cooper and J. Foote, "Summarizing popular music via structural similarity analysis," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2003, pp. 127–130.

[6] K. Lee and M. Cremer, "Segmentation-based lyrics-audio alignment using dynamic programming," in *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR)*, 2008, pp. 395–400.

[7] E. Peiszer, "Automatic audio segmentation: Segment boundary and structure detection in popular music," Master's thesis, Vienna University of Technology, Vienna, Austria, 2007.

[8] J. Paulus and A. Klapuri, "Music structure analysis by finding repeated parts," in *Proceedings of the 1st ACM Workshop on Audio and Music Computing for Multimedia (AMCMM)*. ACM, 2006, pp. 59–68.

[9] M. Mueller and S. Ewert, "Joint structure analysis with applications to music annotation and synchronization," in *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR)*, 2008, pp. 389–394.

[10] S. Perttu, "Combinatorial pattern matching in musical sequences," 2000.

[11] M. Crochmore, C. S. Iliopoulos, T. Lecroq, and Y. J. Pinzon, "Approximate string matching in musical sequences," in *Proceedings Prague Stringology Club (PSC)*, 2001, pp. 26–36.

[12] T. Crawford, "String-matching techniques for musical similarity and melodic recognition," *Computing in Musicology*, pp. 73–100, 1998.

[13] V. Mkinen, G. Navarro, and E. Ukkonen, "Transposition invariant string matching," *Journal of Algorithms*, vol. 56, pp. 124–153, 2005.

[14] C. Charras and T. Lecrog, *Handbook of Exact String Matching Algorithms*. King's College Publications, 2004.

[15] M. Crochemore, "An optimal algorithm for computing the repetitions in a word," *Information Processing Letters*, vol. 12, pp. 244–250, 1981.

[16] C. S. Iliopoulo, D. W. G. Moore, and K. Park, "Covering a string," *Algorithmica*, vol. 16, pp. 288–297, 1996.

[17] E. Cambouropoulos, "Musical parallelism and melodic segmentation: A computational approach," *Music Perception*, vol. 23, no. 3, pp. 249–269, 2006.

[18] E. Cambouropulos, M. Crochemore, C. S. Iliopoulos, L. Mouchard, and Y. J. Pinzon, "Algorithms for computing approximate repetitions in musical sequences," *International Journal of Computer Mathematics*, vol. 79, no. 11, pp. 1135–1148, 2002.

[19] J. Hsu, C. Liu, and A. Chen, "Discovering nontrivial repeating patterns in music data," in *IEEE Transactions on Multimedia*, vol. 3, 2001, pp. 311–325.

[20] T. Jehan, "Hierarchical multi-class self similarities," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2005, pp. 311–314.