

DETECTION OF STREAM SEGMENTS IN SYMBOLIC MUSICAL DATA

Dimitris
Rafailidis*

Alexandros
Nanopoulos*

Emilios
Cambouropoulos[#]

Yannis
Manolopoulos*

* Dept of Computer Science, Aristotle Univ. of Thessaloniki, {draf, ananopou, manolopo}@csd.auth.gr

[#] Dept. of Music Studies, Aristotle University of Thessaloniki, emilios@mus.auth.gr

ABSTRACT

A listener is thought to be able to organise musical notes into groups within musical streams/voices. A *stream segment* is a relatively short coherent sequence of tones that is separated horizontally from co-sounding streams and, vertically from neighbouring musical sequences. This paper presents a novel algorithm that discovers musical stream segments in symbolic musical data. The proposed algorithm makes use of a single set of fundamental auditory principles for the concurrent horizontal and vertical segregation of a given musical texture into stream segments. The algorithm is tested against a small manually-annotated dataset of musical excerpts, and results are analysed; it is shown that the technique is promising.

1. INTRODUCTION

Voice separation algorithms [6, 8, 12, 13, 14, 16, 17, 18] attempt to model computationally the segregation of polyphonic music into separate voices; music segmentation algorithms [1, 4, 5, 7, 15] on the other hand, segment music voices/streams into smaller coherent groups. A common assumption underlying computational models of musical structure is that, firstly voices (or at least the melody) have to be identified and, then, segmentation can be applied to individual voices. For instance, Temperley [17] states that ‘once the voices of a texture are identified, the grouping problem becomes more tractable. ... polyphonic grouping cannot be done without first grouping the notes into contrapunctal lines.’ (pp.81, 83).

In the current paper, the concept of *stream segment* is introduced, i.e. a relatively small number of tones grouped together into a coherent ‘whole’ perceived independently from other adjacent tones. Such stream segments may be organised into longer streams in case streams are relatively stable for a period of time, or may remain independent structural units. The advantage of adopting the concept of stream segments is that they are meaningful in any type of music, not only when music has a relatively ‘fixed’ number of voices, as in fugues, choral music, string quartets (see example of stream segments in Figure 1). The proposed algorithm makes use of a single set of auditory principles for the concurrent horizontal and vertical segregation of a musical texture into stream segments.

An algorithm capable of detecting stream segments can be very useful as it enables the organisation of tones into coherent groups that are musically meaningful, allowing thus more efficient and higher quality analytic processing. Most music analytic methodologies (from traditional harmonic analysis to pc-set theory and semiotic analysis) rely on an initial breaking down of the music into relevant (hierarchical) groups/segments/spans (we would say stream segments). In terms of MIR, for instance, a melodic pattern should be identified not spread across different voices or melodic boundaries (perceptually implausible), but within meaningful musical units, or cover-song detection algorithms may be enhanced if they can identify pertinent structural similarities between stream segments (e.g. between melodic segments rather than accompanimental segments).

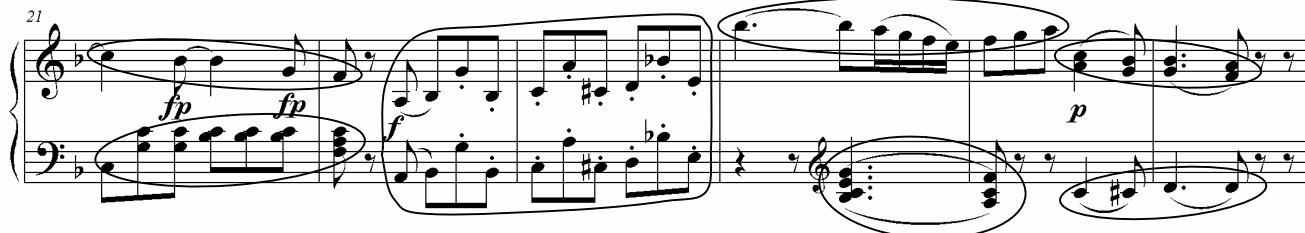


Figure 1 Excerpt from Mozart’s Sonata K332, Allegro Assai. Potential stream segments are circled (other options: third segment broken into two pseudopolyphonic voices, and last two segments seen as a single homophonic segment).

2. STREAM SEGMENTS

Most voice separation algorithms such as algorithms [6, 8, 12, 13, 14, 16, 17, 18] model computationally the segregation of polyphonic music into separate voices. Such algorithms commonly assume that ‘voice’ is a monophonic sequence of successive non-overlapping musical tones. Karydis et al. [12] adopt a perceptual view of musical ‘voice’ that corresponds to the notion of auditory stream and develop a computational model that automatically splits a musical score into different voices/streams (fewer voices than the maximum number of notes in the greatest chord can be detected manually in [13]).

An underlying assumption of such algorithms is that a fixed number of voices/streams evolve throughout an individual piece of music (sometimes a voice may pause for a certain period and, then, reappear again later - see more on voice and stream in [3]). Most of these algorithms are tested against groundtruth that consists of musical pieces that have a ‘fixed’ number of voices/streams, such as fugues, choral music, string quartets, songs (melody and accompaniment).

The above assumption, however, is limiting. There exists a significant amount of music that is not composed of a steady number of voices/streams. In many musical works, homophonic, polyphonic, heterophonic elements are mixed together not allowing a listener to trace musical streams throughout the duration of a piece. This fact has led us to hypothesize a novel music theoretic concept, which we will refer to as a *stream segment* (we are not aware of any equivalent music theoretic notion). A stream segment is a relatively short coherent sequence of tones that is separated horizontally from co-sounding streams and, vertically from neighbouring musical sequences.

Grouping relies essentially on fundamental cognitive mechanisms that enable a listener to perceive individual entities as *gestalts/wholes* [2, 9, 10]. Such mechanisms are based on the principles of proximity and similarity: proximal or similar entities in terms of time, space, pitch, dynamics, timbre to be grouped perceptually together. Such principles are applied locally (e.g. a large pitch interval in-between smaller ones signifies a potential local boundary) or at a higher-level (e.g. a repeating pitch pattern gives rise to a higher level structural unit delimited by pattern boundaries) [4]. In this paper we will discuss segmentation only in relation to the application of gestalt principles at the local level.

In the temporal domain, the most important perceptual principles are the following:

T1 – Synchronous Note Principle: ‘Notes with synchronous onsets and same IOIs (durations) tend to be merged into a single sonority.’ ([12], p.446)

T2 – Principle of Temporal Continuity: ‘Continuous or recurring rather than brief or intermittent sound sources’ evoke strong auditory streams ([10], p.12).

In essence, these two principles indicate that concurrent sounds and sounds following closely one another tend to be merged into the same group/stream, and that asynchronous overlapping tones and sounds occurring at a distance from one another tend to be perceived as belonging to different groups.

In the pitch domain, the most important principles are:

P1 – Principle of Tonal Fusion: The perceptual independence of concurrent tones is weakened when they are separated by intervals (in decreasing order: unisons, octaves, perfect fifths...) that promote tonal fusion [10].

P2 – Pitch Co-modulation Principle: ‘The perceptual union of concurrent tones is encouraged when pitch motions are positively correlated.’ ([10], p.31)

P3 – Pitch Proximity Principle: ‘The coherence of an auditory stream is maintained by close pitch proximity in successive tones within the stream.’ ([10], p.24)

The first two of these principles apply in the case of synchronous concurrent tones, whereas the third principle applies for successive non-overlapping tones.

By combining the principles we get the following:

a. (T1 & P1 & P2) Synchronous notes tend to be merged; merging is stronger when the notes are separated by intervals that promote tonal fusion and when the pitch motions of concurrent notes are positively correlated,

b. (T2 & P3) Successive notes tend to be grouped together when they follow each other closely and the pitch intervals separating them are relatively small.

In all other cases, groupings are discouraged and potential segmentation boundaries are introduced.

3. THE ALGORITHM

Following the above discussion a stream segment detection algorithm has been developed. The algorithm is based on the *k*-nearest neighbor clustering algorithm, where the distances between notes are computed based on four of the above auditory perceptual organisation principles (only the Tonal Fusion principle is not taken into account at this stage). The algorithm accepts as input a musical piece in symbolic format (quantised piano-roll) and outputs a list of musical stream segments.

The stream segment detection algorithm is illustrated in Figure 3. Procedure `Segment` gets the list of notes *N* of the examined piece, and three parameters, *k* (number of nearest neighbors), *cL* (chain length; see below), and *w* (window length; see below). It first computes the distance matrix *D* between each pair of notes. For the computation of *D*, we examine three cases: (i) If two notes, *n_i* and *n_j*, fully coincide, they are checked by a procedure called `verifyCoincide`, which examines whether the vertical structure in the temporal neighborhood of these notes is strong or not (this case examines notes in accordance to the Synchronous Note and the Pitch Co-Modulation principles).

In particular, for each coinciding notes n_i and n_j , `verifyCoincide` forms two separate ‘chains’ centered at n_i and n_j , which contain preceding and following notes that coincide with each other and have the smallest absolute pitch difference with the center notes. Procedure `verifyCoincide` tries to form chains with length up to cL . Figure 2a illustrates an example of two such chains with length 3, centered at notes n_2 and n_5 . If two such chains can be formed this means that there exists vertical homophonic structure, as several nearby notes coincide. Additionally, `verifyCoincide` examines the *direction* in the movement of chains, by checking the deviation in the pitch differences between the corresponding chain elements. This way, cases like the one depicted in Figure 2b, where the two chains containing concurrent notes move in non-parallel directions, are distinguished and merging is avoided according to the Pitch Co-modulation Principle.

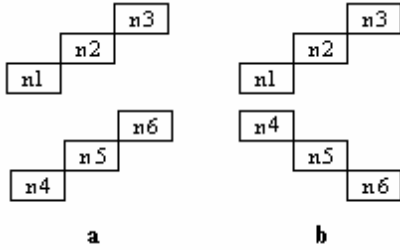


Figure 2 Parallel (a) and non-parallel (b) note chains.

(ii) The second case in the computation of distance matrix D , examines asynchronous notes that overlap in time; such notes are considered to have infinite distance. (iii) The final case examines notes that neither overlap nor coincide and their distance is set equal to their normalized pitch difference (`normPD`) plus their normalized IOI (`normIOI`), divided by two (in accordance to the Temporal Continuity and Pitch Proximity Principles). A normalized pitch difference (IOI, respectively) is taken by dividing with the maximum occurring pitch difference (IOI, respectively) between notes occurring within the same time window of length less than w .

Next, clustering is performed with procedure `kNNCluster`. This procedure implements the k -nearest neighbor clustering algorithm [11], which finds for each note, the list of its k -nearest neighbors (sorted in increasing order of distance). The `kNNCluster` procedure is able to detect clusters of various shapes and sizes. It assigns to each pair of notes n_i and n_j , a mutual neighborhood value (MVN) that is equal to $p + q$, where p is the position of n_j in the list of n_i and q is the position of n_i in the list of n_j . If n_j or n_i do not belong to the list of each other, then MVN equals infinity. Therefore, if MVN is small, then the corresponding notes are close to each other. For this reason, the procedure joins in the same cluster, notes with $MVN = 2, 3, \dots, 2k$.

The main characteristic of our stream segment detection algorithm (procedure `Segment`) is that it places a pair of

objects (notes) in the same cluster, if they are mutual k -nearest neighbors. Due to this characteristic, as will be verified experimentally, this algorithm is able to detect segments of various shapes and sizes, as it is based only on the coherency within clusters. In contrast to several other clustering algorithms, the proposed algorithm does not require that the number of final clusters be determined a priori. This is very suitable for the case of musical stream segment detection as the number of clusters is difficult to be defined correctly in advance. The complexity of the algorithm is $O(N^2)$ for N notes.

```

Procedure Segment( $N, k, cL, w$ )
  foreach  $n_i \in N$ 
    foreach  $n_j \in N$ 
      if coincide( $n_i, n_j, cL$ )
        if verifyCoincide( $n_i, n_j$ )
           $D(n_i, n_j) = 0$ ;
        else if nonoverlap( $n_i, n_j$ )
           $D(n_i, n_j) = 0.5 * (\text{normPD}(n_i, n_j) + \text{normIOI}(n_i, n_j))$ ;
        else
           $D(n_i, n_j) = \infty$ ;
    kNNCluster( $N, D, k$ );
end;

Procedure kNNCluster( $N, D, k$ )
  foreach  $n_i \in N$ 
     $L(n_i) = \text{kNN}(k, N, D)$ ;
  foreach  $n_i \in N$ 
    foreach  $n_j \in N$ 
      if  $n_j \in L(n_i)$  and  $n_i \in L(n_j)$ 
         $MVN(n_i, n_j) = \text{pos}(n_j, L(n_i)) + \text{pos}(n_i, L(n_j))$ ;
      else
         $MVN(n_i, n_j) = \infty$ ;
    for  $m = 2$  to  $2 * k$ 
      find all  $n_i, n_j \in N$  with  $MVN(n_i, n_j) == m$ 
      assign  $n_i, n_j$  to the same cluster
  end;

```

Figure 3 Algorithmic description of the proposed method.

Regarding the tuning of parameters, for several types of musical pieces, like sonatas, mazurkas, waltzes, parameter cL takes small values (not more than 2), whereas for others, like fugues, it takes relatively high values (around 10). The reason is that cL is responsible for detecting vertical homophonic structures (like sonatas, mazurkas, waltzes), a relatively low value suffices. In contrast, for genres for which this does not hold (polyphonic music like fugues), a higher value is required to avoid false drops.

The tuning of parameter k depends on the characteristics of each particular piece, thus has to be tuned accordingly. In our experiments, values within the range 1 to 5 were adequate. In contrast, when considering the stream segment detection algorithm for voice/stream separation (for instance, in Figure 6 the algorithm finds two streams, i.e. melody and accompaniment), higher k values should be used. The reason is that, in order to detect entire voices, which can be considered as concatenations of

smaller segments, a higher k value enables the algorithm to combine more small segments into larger ones, and eventually to voices/streams. For the detection of smaller, localized segments, small values of k , as those mentioned previously, have to be used. Finally, the length of window w is set constant to a value equal to half a second, in order to capture local variance of pitch and IOI differences.

4. EXPERIMENTS AND RESULTS

The proposed algorithm has been tested on a small set of musical extracts for piano.¹ The dataset has been annotated by a music theory research student that was instructed to indicate non-overlapping low-level groups of notes that may be perceived as ‘wholes’ (not to indicate phrases, themes) - the example of figure 1 was discussed before preparing the groundtruth. This small dataset acted as groundtruth for testing the algorithm in this first experiment. Seven extracts were selected from different musical styles: from the openings of Bach’s Fugue No.14 in F# major, WTCI, BWV859, Chopin’s Mazurkas Op6, No.2 & Op.7, No.5 and Waltz Op.69, No.2 and Beethoven’s Sonatas Op.2, No.1, Op.13 (Pathétique) & Op.31, No.3 (overall 1500 notes). The small size of this dataset allowed a more detailed qualitative analysis as well (see discussion below and Figures 4, 5, 6 and 7), in addition to a quantitative analysis.

We studied experimentally the performance of the proposed method (henceforth denoted as kNNClust). For comparison purposes, we also examined a segmentation method that is based on a hierarchical clustering algorithm (henceforth denoted as HierarchicalClust). We considered the following variations of hierarchical clustering: single link, complete link, and Ward’s. We found significant differences in the performance of the variations. Thus, we examined all of them, and for each measurement we present the best result among them. Hierarchical methods run on the same distance matrix produced by Procedure Segment, whereas the number of clusters was tuned to give the best results for these methods.

For each music piece and for each segmentation method, we measured accuracy against the groundtruth in terms of the *Jaccard Coefficient*. Let f_{11} denote the number of notes that belong to the same segment in the groundtruth and have been assigned to the same segment by the segmentation method. In the same manner, f_{10} denotes the number of notes that belong to the same segment in the groundtruth but have not been assigned to the same segment by the segmentation method, whereas f_{01} denotes the number of notes that do not belong to the same segment in the groundtruth but have been assigned to the same segment by the segmentation method. The Jaccard Coefficient is computed as the ratio: $f_{11}/(f_{10}+f_{01}+f_{11})$

¹ These pieces were downloaded in Melisma format from the Kern collection (<http://kern.humdrum.org>)

Segments vary significantly in size, i.e., the number of notes they contain. Jaccard Coefficient tends to penalize excessively incorrect assignments of notes to larger segments and, in contrast, it tends to underestimate errors in smaller segments. To cope for this factor and, thus, to treat equally segments of various sizes, we examined a local Jaccard Coefficient, by computing the f_{11} , f_{10} , and f_{01} numbers only for notes that belong within a time-window. As mentioned, in the proposed method we use a time window to measure normalized pitch and IOI distances. Therefore, we naturally select the length of this window when measuring the local Jaccard Coefficient. Comparison results between kNNClust and HierarchicalClust in terms of the local Jaccard Coefficient are presented in Table 1. Clearly, due to its special characteristics, the proposed method kNNClust compares favorably against HierarchicalClust.

Title	kNNClust	HierClust
Beethoven, Sonata, Op31, No3	0.96	0.82
Beethoven, Sonata, Op2 No1	0.82	0.59
Beethoven, Sonata, Op13, No8	0.84	0.60
Chopin, Mazurka, Op7, No5	0.86	0.62
Chopin, Mazurka, Op6, No2	0.83	0.75
Chopin, Waltz Op.69, No.2	0.94	0.51
Bach, Fugue No14 BWV859	0.84	0.73

Table 1 Accuracy (0 worst, 1 best) by local Jaccard Coefficient.

The algorithm was also tested on a dataset of complete pieces similar to the dataset used as groundtruth in [12]. In this dataset only voices/streams are annotated - not stream segments. A different measure was, therefore, used to evaluate the performance of the algorithm, that rated how well the detected stream segments fall within the voices/streams of the groundtruth, i.e. stream segments are penalised if they are shared across different streams (stream segments are detected for each piece using similar parametric settings as in the previous test). We measured the *voice-crossing penalty* as the ratio of the number of note pairs that were assigned to the same segment whilst they belong to different voices, divided by the number of note pairs that were assigned to the same segment. The voice-crossing penalty is presented in Table 2. In all cases, the measured error is low.

Title	kNNClust
Bach, Fugue No. 7 in E-flat major, BWV 852	0.09
Bach, Fugue No. 11 in F major, BWV 856	0.10
Bach, Fugue No. 1 in C major, BWV 846	0.14
Bach, Fugue No. 14 in F#minor, BWV 859	0.09
Joplin, Harmony Club Waltz	0.02
Chopin, Mazurka in C Major, Op7, No5	0.00
Chopin, Mazurka in C-sharp Minor, Op6, No2	0.02
Chopin, Waltz in D-flat Major, Op. 64, No. 1	0.09

Table 2 Results for voice-crossing penalty (0 best, 1 worst).

The results were examined in detail (qualitative analysis) in order to understand the kinds of mistakes produced by the algorithm. In the example of Figure 4, the algorithm detects coherent groups of notes such as melodic segments, harmonic accompanimental fragments, homophonic passages. Sometimes the algorithm over-segments, e.g. right hand in mm. 1-2, 10-11, 18-19 (these can be seen as single segments due to motivic similarity), or left hand in mm. 8, 18-19 (isolated chords). In m.10, the first note should be grouped with the following notes due to motivic repetition (NB. the current algorithm does not incorporate procedures for detecting parallelism). In the example of Figure 5, it is interesting that the algorithm groups together in stream segments the rhythmically independent notes that are ‘sandwiched’ in-between the homophonic top and lower notes (note that the algorithm ‘mistakenly’ does not group the top notes with the lower notes at the end of the passage). In Figure 6, the accompaniment should be segmented at the beginning of m.9 (not beginning of m.8) due to the fact that mm.9-12 are essentially a repetition of mm.5-8 (note that the algorithm does not detect repetitions). There are also mistakes at the edges of stream segments as in the ending of the first segment. Finally, in Figure 7, the algorithm ‘correctly’ groups together different contrapuntal lines in m.10 (perceptually they may be seen as a single stream), but makes serious mistakes in m.9 where independent voices are merged in the same segments (this type of mistake is under investigation).

5. CONCLUSIONS AND FUTURE WORK

This paper introduces the notion of *stream segment*, i.e., a relatively short coherent sequence of tones that is separated horizontally from co-sounding streams and, vertically from neighbouring musical sequences. A novel algorithm has been proposed that discovers musical stream segments in symbolic musical data based on a set of fundamental auditory principles. The algorithm has been tested against a small manually-annotated dataset of musical excerpts, and the results have been analysed; it was shown that the technique generates promising results.

The proposed concept is new in the sense that it views streaming and segmentation as linked to the same fundamental perceptual principles and that a single algorithm can generate ‘stream segments’. Further study, however, is required to make this concept clearer and to provide a crisp definition. Links to appropriate music theoretic notions are important, along with detailed theoretical and empirical analyses of a large number of actual musical examples. Such analyses can form the necessary ground truth on which algorithms are tested. Ambiguity, overlapping between stream segments, hierarchic segmentations have to be considered in the future studies of stream segments. We hope that this paper will initiate further research on this topic.

6. REFERENCES

- [1] Barry, D., Gainza, M., & Coyle, E. (2007) Music Structure Segmentation using the Azimugram in conjunction with Principal Component Analysis, *Proc. 123rd Audio Engineering Society Convention*, Jacob Javitz Centre, Manhattan, New York.
- [2] Bregman, A (1990) *Auditory Scene Analysis: The Perceptual Organisation of Sound*. The MIT Press, Cambridge (Ma).
- [3] Cambouropoulos, E. (2006) ‘Voice’ Separation: theoretical, perceptual and computational perspectives. In *Proceedings of ICMPC’06*, 22-23 August, Bologna, Italy.
- [4] Cambouropoulos E. (2006) Musical Parallelism and Melodic Segmentation: A Computational Approach. *Music Perception* 23(3):249-269.
- [5] Cambouropoulos E. (2001). The *Local Boundary Detection Model (LBDM)* and its application in the study of expressive timing. In *Proc. of ICMC01*, Havana, Cuba.
- [6] Cambouropoulos, E. (2000) From MIDI to Traditional Musical Notation. In *Proceedings of the AAAI Workshop on Artificial Intelligence and Music*, Austin, Texas.
- [7] Chew, E. and Wu, X. (2004) Separating voices in polyphonic music: A contig mapping approach. In *Computer Music Modeling and Retrieval: Second International Symposium (CMMR 2004)*, pp. 1-20.
- [8] Conklin, D. and Anagnostopoulou, C. (2006). Segmental Pattern Discovery in Music. *INFORMS Journal of Computing*, 18(3), pp. 285-293.
- [9] Deutsch, D. (1999) Grouping Mechanisms in Music. In D. Deutsch (ed.), *The Psychology of Music* (revised version). Academic Press, San Diego.
- [10] Gowda, K.C. and G. Krishna, G. (1978) Agglomerative clustering using the concept of mutual nearest neighborhood. *Pattern Recognition*, 10:105-112.
- [11] Huron, D. (2001) Tone and Voice: A Derivation of the Rules of Voice-Leading from Perceptual Principles. *Music Perception*, 19(1):1-64.
- [12] Karydis, I., Nanopoulos, A., Papadopoulos, A.N. & Cambouropoulos, E., (2007) VISA: The Voice Integration/Segregation Algorithm. In *Proceedings of ISMIR’07*, pp. 445-448, Vienna, Austria
- [13] Kilian J. and Hoos H. (2002) Voice Separation: A Local Optimisation Approach. In *Proceedings of ISMIR’02*, pp.39-46.
- [14] Kirilin, P.B. and Utgoff, P.E. (2005) VoiSe: Learning to Segregate Voices in Explicit and Implicit Polyphony. In *Proceedings of ISMIR’05*, Queen Mary, Univ. of London, pp. 552-557.
- [15] Levy, M. and Sandler, M. (2006). Extraction of High-Level Musical Structure from Audio Data and its Application to Thumbnail Generation. In *Proc. ICASSP 2006*
- [16] Madsen, S. T. and Widmer, G. (2006) Separating Voices in MIDI. In *Proceedings ICMPC06*, Bologna, Italy.
- [17] Szeto, W.M. and Wong, M.H. (2003) A Stream Segregation Algorithm for Polyphonic Music Databases. In *Proceedings of IDEAS’03*.
- [18] Temperley, D. (2001) *The Cognition of Basic Musical Structures*. The MIT Press, Cambridge (Ma).

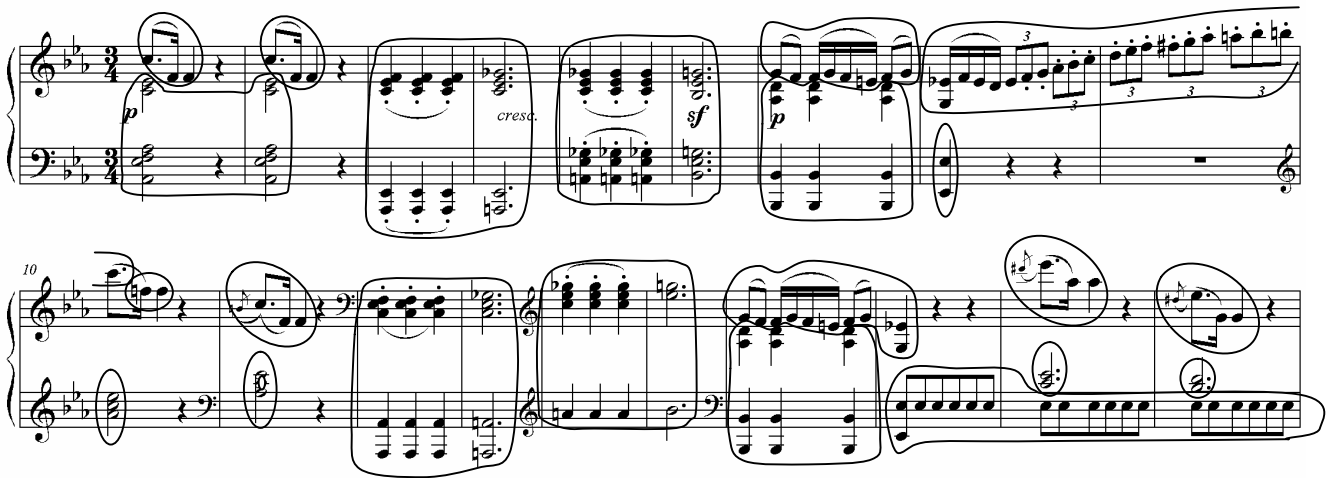


Figure 4 Stream segments detected by algorithm in the opening of Beethoven's Sonata Op.31, No.3 (see text for details).

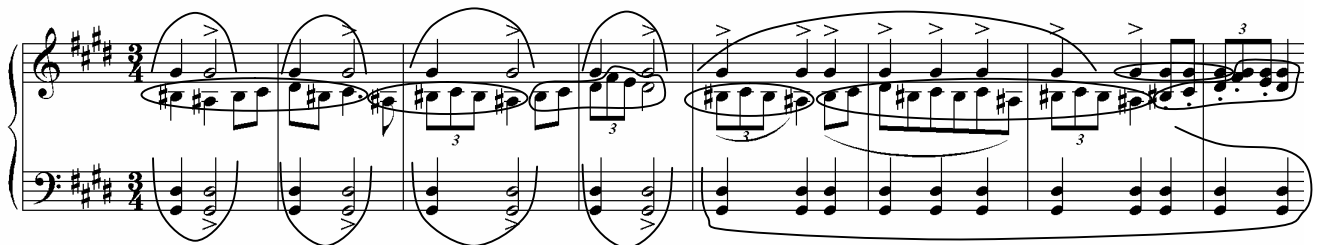


Figure 5 Stream segments detected by algorithm in the opening of Chopin's Mazurka Op.6, No.2 (see text for details)

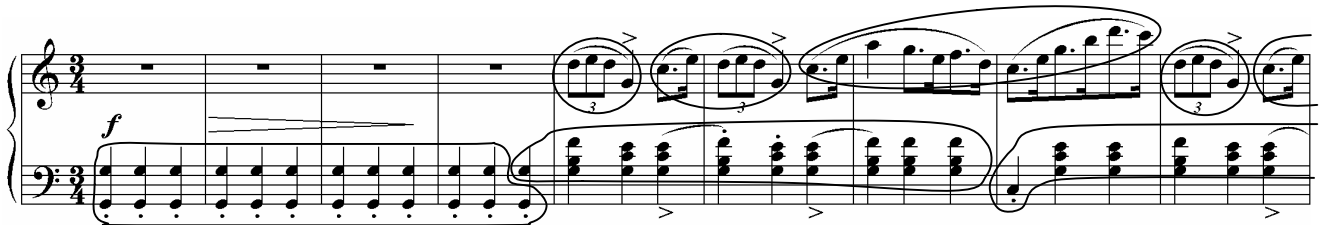


Figure 6 Stream segments detected by algorithm in the opening of Chopin's Mazurka Op.7, No.5 (see text for details)



Figure 7 Stream segments detected by algorithm in the opening of Bach's Fugue in F# major No.14, WTCL, BWV859 (see text for further details)