Comparing Voice and Stream Segmentation Algorithms

Nicolas Guiomard-Kagan, Mathieu Giraud, Richard Groult, Florence Levé

COMPARING VOICE AND STREAM SEGMENTATION ALGORITHMS

Nicolas Guiomard-Kagan MIS, U. Picardie Jules Verne Amiens, France

Mathieu Giraud CRIStAL (CNRS, U. Lille) Lille, France

Richard Groult MIS, U. Picardie Jules Verne (UPJV) Amiens, France

Florence Levé MIS, U. Picardie Jules Verne (UPJV) Amiens, France

{nicolas, mathieu, richard, florence}@algomus.fr

ABSTRACT

Voice and stream segmentation algorithms group notes from polyphonic data into relevant units, providing a better understanding of a musical score. Voice segmentation algorithms usually extract voices from the beginning to the end of the piece, whereas stream segmentation algorithms identify smaller segments. In both cases, the goal can be to obtain mostly monophonic units, but streams with polyphonic data are also relevant. These algorithms usually cluster contiguous notes with close pitches. We propose an independent evaluation of four of these algorithms (Temperley, Chew and Wu, Ishigaki *et al.*, and Rafailidis *et al.*) using several evaluation metrics. We benchmark the algorithms on a corpus containing the 48 fugues of *Well-Tempered Clavier* by J. S. Bach as well as 97 files of popular music containing actual polyphonic information. We discuss how to compare together voice and stream segmentation algorithms, and discuss their strengths and weaknesses.

1. INTRODUCTION

Polyphony, as opposed to monophony, is a music created by simultaneous notes (see Figure 1) coming from several instruments or even from a single polyphonic instrument, such as the piano or the guitar. Polyphony usually implies chords and harmony, and sometimes counterpoint when the melody lines are independent.

Voice and stream segmentation algorithms group notes from polyphonic symbolic data into layers, providing a better understanding of a musical score. These algorithms make inference and matching for relevant patterns easier. They are often based on perceptive rules as studied by Huron [7] or Deutsch [6]. Chew and Wu gathered these rules into four principles [2]:

- *(p1)* Voices are monophonic;

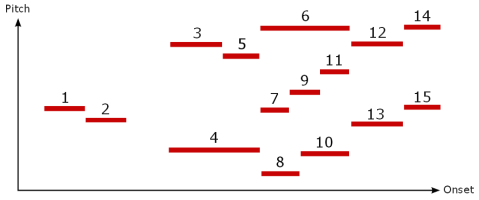

Figure 1: In this piano-roll representation, each segment describes a note. The horizontal axis represents time and the vertical axis represents the pitch.

- *(p2)* At least once, all voices must be played simultaneously;
- *(p3)* Intervals are minimized between successive notes in the same stream or voice (pitch proximity);
- *(p4)* Voices tend not to cross.

Voice segmentation algorithms extract voices from the beginning to the end of the piece. Usually, the resulting voices are monophonic (*p1*) and, at some point, all the voices do appear (*p2*). The algorithms described by Chew and Wu [2] and Ishigaki *et al.* [9] first identify *contigs* of notes, then link these contigs. These algorithms will be discussed later. De Valk *et al.* [5] proposed a machine learning model with a neural network to separate voices in lute tablatures. The study of Kirlin and Utgoff [13] uses another machine learning model to separate voices, taking in consideration both actual polyphony and *implicit* polyphony, such as the one obtained with arpeggios.

Stream segmentation algorithms identify segments generally smaller than complete voices. A *stream* is a group of coherent notes, usually respecting principles such as *p3* and *p4*. Temperley's algorithm [17] extracts streams with respect to several constraints. Rafailidis *et al.*'s algorithm [16], based on an earlier work by [11], uses a k-nearest neighbors clustering technique on individual notes. Both algorithms will be discussed in Sections 3.1 and 3.2. The study by Madsen and Widmer [15], inspired by Temperley [17], allows crossing voices. The method of Kilian and Hoos [12] starts by cutting the input score into sections called *slices* such that all the notes of a slice overlap; Then, an optimization method involving several evaluation

functions is applied to divide and combine the slices into voices; The output voices can contains chords.

Depending on the algorithms, the predicted streams can thus be small or large. However, such algorithms do predict groups of notes, especially contiguous relevant notes, and may thus be compared against full voice segmentation algorithms. De Nooijer *et al.* [4] made a comparison by humans of several voice and stream separation algorithms for melody finding.

In this paper, we independently evaluate some of these algorithms, benchmarking in the same framework voice and stream segmentation algorithms. We compare some simple and efficient algorithms that were described in the litterature [2, 9, 16] and added the algorithm in [17] for which an implementation was freely available. Our corpus includes Bach's fugues (on which many algorithms were evaluated) but also pop music containing polyphonic material made of several monophonic tracks. The next two sections detail these algorithms. Section 4 presents the evaluation corpus, code, and methods. Section 5 details the results and discusses them.

2. VOICE SEPARATION ALGORITHMS

2.1 Baseline

To compare the different algorithms, we use a very simple reference algorithm, based on the knowledge of the total number of voices (*p2*). The *baseline algorithm* assigns a reference pitch for each voice to be predicted, then assigns each note to the voice which has the closest reference pitch (Figure 2).

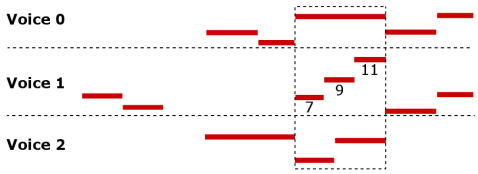

Figure 2: The baseline algorithm assigns each note to the voice having the closest reference pitch. This reference pitch is computed by averaging pitches on segments having the highest number of simultaneous notes. Here the middle voice, Voice 1, has a reference pitch that is the average of the pitches of notes 7, 9 and 11.

2.2 CW

The CW algorithm separates voices by using the four principles (*p1, p2, p3, p4*) [2].

Contigs. The first step splits the input data into blocks such that the number of notes played at the same time during one block does not change. Moreover, when a note crosses the border of two blocks and stops or starts to sound inside a block, the block is split in two at this time. The obtained blocks are called *contigs* (Figure 3). By construction, the number of played notes inside a contig is constant. Notes are grouped from the lowest to the highest pitch in *voice fragments* (Figure 3).

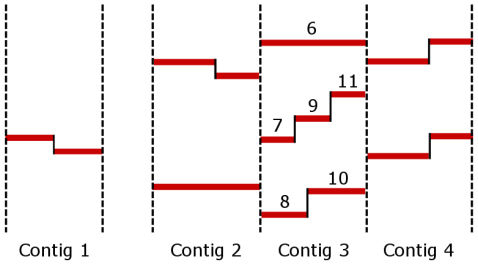

Figure 3: Four contigs: Contig 3 contains three fragments, $\{6\}$, $\{7, 9, 11\}$ and $\{8, 10\}$.

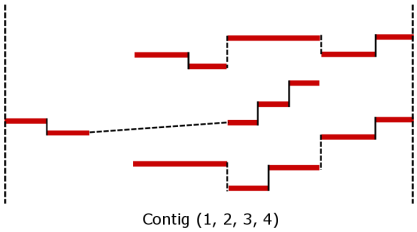

Figure 4: Connection Policy: All fragments are connected with respect to *p3*.

Connection Policy. The second step links together fragments from distinct contigs (see Figure 4). The contigs containing the maximal number of voices are called *maximal voice contigs* (*p2*). The connection starts from these maximal contigs: Since the voices tend not to cross, the order of the voices attributed to fragments of these contigs has a strong probability to be the good one (*p2* and *p4*).

Given two fragments in contiguous contigs, CW defines a *connection weight*, depending on n_1, the last note of the left fragment, and on n_2, the first note of the right fragment. If n_1 and n_2 are two parts of the same note, this weight is $-K$, where K is a large integer, otherwise the weight is the absolute difference between the pitches of the two notes (*p3*). The fragments connected between two contigs are the ones which minimize the total connection weight (Figure 5).

2.3 CW-Prioritized

Ishigaki *et al.* [9] proposed a modification of CW algorithm in the merging step between the contigs. Their key observation is that the *entry of a voice* is often non ambiguous, contrary to the exit of a voice that can be a "fade out" which is difficult to precisely locate. Instead of starting from maximal voice contigs, they thus choose to start

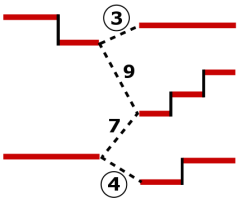

Figure 5: Connection between contigs: The selected links are the ones minimizing the total weight (3 + 4 = 7).

only from adjacent contigs with an *increasing* number of voices. For example in Figure 3, the procedure starts by merging Contig 1 with Contig 2. The choice of merged fragments is identical to the method described in CW algorithm. After the merge of all fragments of two adjacent contigs c_1 and c_2, we get a new contig containing the same number of voices than in c_2 (see Figure 6).

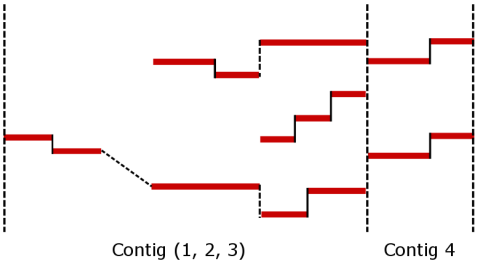

Figure 6: Contig combining: Contigs 1, 2, then 3 are combined, resulting in a Contig 1+2+3 with 3 voices.

The procedure described above is reiterated as long as two adjacent contigs have an increasing number of voices. If at the end of this procedure, there is more than one contig, they are merged by the original CW connection policy.

3. STREAM SEGMENTATION ALGORITHMS

We also study *stream segmentation* algorithms, which do not segment a score into voices but into streams that may include overlapping notes. Streams can be melodic fragments, but also can cluster related notes, such as chords. A voice can be thus split into several streams, and a stream can cluster notes from different voices.

3.1 Streamer

The algorithm proposed by Temperley extracts streams while respecting several constraints [17]. The first constraint is pitch proximity: two contiguous notes with close pitches are placed in the same stream (*p3*). The second constraint is temporal: when there is a long rest between two notes, the second note is put into a new stream (Figure 7). The last principle allows the duplication of a note in two voices (provided that the streams do not cross, *p4*).

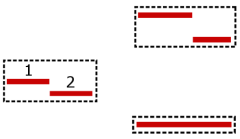

Figure 7: Due to the rest after note 2, Streamer assigns notes 1 and 2 to a stream that does not include any other notes.

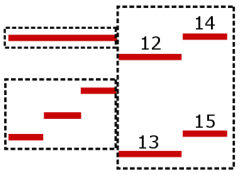

Figure 8: Stream Segment assigns notes 12, 13, 14, and 15 in a same stream. The notes 13-15 can be seen as a transposition of notes 12-14, forming a succession of chords.

3.2 Stream Segment

The algorithm by Rafailidis *et al.* [16] clusters notes based on a k-nearest-neighbors clustering. The algorithm first computes a distance matrix, which indicates for each possible pair of notes whether they are likely to belong to the same stream. The distance between two notes is computed according to their synchronicity (Figure 8), pitch and onset proximity (among others criteria); then for each note, the list of its k-nearest neighbors is established.

3.3 CW-Contigs

We finally note that the first step of the CW algorithm (contig creation) can be considered as a stream segmentation algorithm. We call this first step CW-Contigs. For example, on the Figure 3, this method creates 8 streams corresponding to the 8 voice fragments of the four contigs.

4. EVALUATION CORPUS AND METRICS

4.1 Evaluation corpus

Usually these algorithms are evaluated on classical music, in particular on counterpoint music such as *fugues*, where the superposition of melodic lines gives a beautiful harmony. As a fugue is made up several voices, this naturally constitutes a good benchmark to evaluate voice separation algorithms [2, 5, 9–11, 15–17]. We thus evaluated the algorithms on the 48 fugues of the two books of the *Well-Tempered Clavier* by J.-S. Bach [1] .

We also wanted to evaluate other forms of polyphonic writing. The problem is to have a ground truth for this task. From a set of 2290 MIDI files of popular music, we formed a corpus suitable for the evaluation of these algorithms. We focused on MIDI tracks (and not on MIDI channels). We kept only "monophonic" tracks (where at most one note is played at any time) of sufficient length (at least 20 % of the length of the longest track). We deleted the tracks corresponding to the drums. We considered each remaining

[1] `.krn` files downloaded from `kern.ccarh.org` [8]

corpus	wtc-i	wtc-ii	pop
files	24	24	97
voices	3.5	3.4	3.0
notes	1041	1071	874

Table 1: Files, and average number of voices and notes for each corpus.

track as an independant voice. Finally, we kept 97 MIDI files with at least 2 voices, composed on average of 3.0 voices (see Table 1).

4.2 Evaluation code

We implemented the algorithms CW-Contigs, CW, CW-Prioritized and Stream Segment, using a Python framework based on music21 [3]. The Streamer algorithm [2] was run with default parameters. As it quantizes input files, the offset and duration of notes in the output are slightly different from the ones in our original files: We thus had to associate notes to the correct ones.

4.3 Evaluation metrics

4.3.1 Note-based evaluation.

A first evaluation is to ask whether the voices are correctly predicted. The *note precision (NPR)* is the ratio between the number of notes correctly predicted (in the good voice) over the total number of notes. On one voice, this measure is the same than the *average voice consistency (AVC)* defined by [2]. However on a piece or on a corpus, we compute the ratio on the total number of notes, instead of averaging ratios as in [2]. Especially in the pop corpus, the distribution of notes is not equal in all pieces and all voices, and this measure better reflects the ability of the algorithm to assign the good voice to each note.

Computing NPR requires to assert *which voice in the prediction corresponds to a given voice of the ground truth.* In a fugue, there may be a formal way to exactly define the voices and number them, from the lowest one to the highest one. But, in the general case, this correspondance is not always obvious. By construction, the two voice segmentation algorithms studied here predict a number of voices equal to the maximal number of voices, whereas the stream segmentation algorithms have no limit for the number of streams. In the general case, one solution is to compare each voice predicted by the algorithm with *the most similar voice of the ground truth*, for example taking the voice of the ground truth sharing the highest number of notes with the predicted voice.

Note-based evaluation tends to deeply penalize some errors in the middle of the scores: When a voice is split in two, half of the notes will be counted as false even if the algorithm did "only one" mistake. Moreover, this is not a fair way to evaluate stream segmentation algorithms, as they may predict (many) more streams than the number of voices. We thus use two other metrics, that better measure the ability of the algorithms to gather notes into voices, even when a voice of the ground truth is mapped to several predicted voices. These metrics do not require to make the correspondence between predicted voices and voices of the truth.

[2] downloaded from `www.link.cs.cmu.edu/melisma`

4.3.2 Transition-based evaluation.

The result of voice or stream segmentation methods can be seen as a set of *transitions*, that are pairs of successive notes in a same predicted voice or stream. We compare these transitions against the transitions defined by the ground truth, and compute usual precision and recall ratios.

The *transition precision (TR-prec)* (called *soundness* by [13]) is the ratio of correctly assigned transitions over the number of transitions in the predicted voices. This is related to *fragment consistency* defined in [2] – but the fragment consistency takes only into account the links between the contigs, and not all the transitions. The *transition recall (TR-rec)* (called *completeness* by [13]) is the ratio of correctly assigned transitions over the number of transitions in the truth. This is again related to *voice consistency* of [2].

For each piece, we compute these ratio on all the voices – taking the number of correct transitions inside *all* the voices, and computing the ratio over the number of transitions inside either *all* the predicted voices or *all* the truth. When the number of voices in the ground truth and in the prediction are equal, the TR-prec and TR-rec ratios are thus equal: we simply call this measure *TR*. Figure 12, at the end of the paper, details an example of NPR and TR values for the six algorithms.

4.3.3 Information-based evaluation.

Finally, we propose to adapt techniques proposed to evaluate music segmentation, seen as an assignation of a label to every audio (or symbolic) frame [1, 14]. Lukashevich defines two scores, S_o and S_u, based on normalized entropy, reporting how an algorithm may over-segment (S_o) or under-segment (S_u) a piece compared to the ground truth. The scores reflect how much information there is in the output of the algorithm compared to the ground truth (S_o) or conversely (S_u). Here, we use the same metrics for voice or stream segmentation: both the ground truth and the output of any algorithm can be considered as an assignation of label to every note. On the probability distribution of these labels, we then compute the entropies $H(\text{predicted}|\text{truth})$ and $H(\text{truth}|\text{predicted})$, that become S_o and S_u after normalization [14]. As these scores are based on notes rather than on transitions, they enable to measure whether the clusters are coherent, even in situations when two simultaneous voices are merged in a same stream (giving thus bad TR ratios).

	wtc-i					wtc-ii					pop				
	avg.	NPR	TR	S_o	S_u	avg.	NPR	TR	S_o	S_u	avg.	NPR	TR	S_o	S_u
Baseline	3.5	71.4%	63.7%	0.48	0.48	3.4	71.9%	62.6%	0.45	0.45	3	**89.5%**	87.1%	**0.77**	0.75
CW	3.5	**83%**	95.9%	**0.73**	0.73	3.4	**87.8%**	95.6%	0.73	0.73	3	84.6%	88.7%	0.76	**0.76**
CW-Prioritized	3.5	82.5%	**97.4%**	0.72	0.72	3.4	86.5%	**97.1%**	**0.74**	0.74	3	64.8%	**89.4%**	0.51	0.5
	avg.	TR-prec	TR-rec	S_o	S_u	avg.	TR-prec	TR-rec	S_o	S_u					
Streamer	16	75.6%	68.3%	0.46	0.62	15.4	75.6%	65.2%	0.42	0.57					
Stream Segment	191.1	76.5%	62.1%	0.23	0.79	214	77.4%	61.9%	0.21	0.79					
CW-Contigs	226.2	**99.4%**	**86.7%**	0.34	**0.98**	282.3	**99.4%**	**86.8%**	0.34	**0.98**					

Table 2: Results on the fugues and on the pop corpora. "avg." is the average number of voices or streams predicted.

5. RESULTS AND DISCUSSION

We evaluated the six algorithms on the 48 fugues of *Well-Tempered Clavier* by J. S. Bach, and moreover the voice separation algorithms were evaluated on the 97 pop files. Table 2 details the results.

5.1 Results

Note and transition-based evaluation. Between 80 % and 90 % of the notes are assigned correctly to the right voice by at least one of the voice separation algorithms. The results confirm that these NPR metric is not very meaningful. The baseline has good NPRs, and on the pop corpus, the baseline NPR is even better than CW and CW-Prioritized. Compared to the baseline algorithm, all algorithms output longer fragments (see Figure 9). As expected, the transition ratio (TR) metrics are better to benchmark the ability of the algorithms to gather relevant notes in the same voice: all the algorithms have better TR metrics than the baseline.

The three stream segmentation algorithms predict more streams that the number of voices in the ground truth, hence low TR-rec ratios. The TR-prec ratios are higher, better than the baseline, and the CW-Contigs has an excellent TR-prec ratio.

Information-based evaluation. An extreme case is perfect prediction, with NPR = TR = 100%, $S_o = 1$ and $S_u = 1$ (like in Bach's Fugue in E minor BWV 855 for both CW and CW-Prioritized). In a pop song (allsaints-bootiecall) where two voices play mostly same notes, the baseline algorithm merges all notes in the same voice, so NPR and TR are close to 50%, but S_o is close to 1 and S_u close to 0.

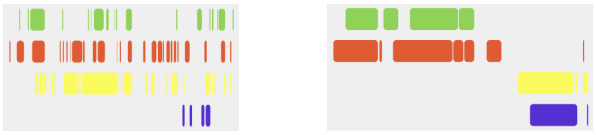

Figure 9: Notes attributed to the wrong voice with the baseline (left) and CW (right) algorithms on Bach's Fugue #2 – book II (in C minor, BWV 871). When CW makes errors, the voices are kept in a same predicted voice.

In the general case, S_u is correlated with TR-prec, and S_o with TR-rec. As expected, in stream segments algorithms, S_u is better than S_o. Note that the Stream Segment has not the best TR-prec ratio (sometimes, it merges notes that are in separate voices), but it has a quite good S_u score among all the algorithms (when it merges notes from separate voices, it tends to put in the same stream all notes that are in related voices). The best S_u scores are obtained by the CW-Contigs, confirming the fact that the contig creation is a very good method that makes almost no error.

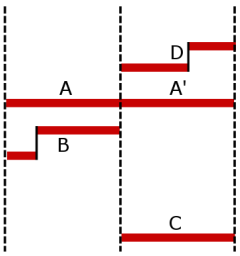

Figure 10: A note spanning two contigs is split in A and A'. CW and CW-Prioritized link the fragments $(A + A')$, $(B + C)$, keeping A in the same voice. The original implementation of Ishigaki *et al.* links the fragments $(A + D)$, (B, A'), duplicating the whole note $A + A'$.

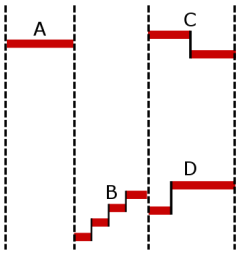

Figure 11: Fragments A and B are in different contigs due to the overlap of previous notes. Both CW-Prioritized and the original implementation of Ishigaki *et al.* link the fragments $(A + B + D)$ and (C), whereas CW links the fragments $(A+C)$ and $(B + D)$.

5.2 Partitioned notes and link weights

With CW algorithm, when a note is cut between two contigs and the voices assigned to those two fragments are different, the predicted voices contain more notes than in the input data. This case was not detailed in the study [2]. To avoid split notes in the output of the algorithm, we choose to allow voice crossing exactly at these points (Figure 10).

Our results for CW-Prioritized differ from the ones obtained in [9]: Their AVC was better compared to CW. In our implementation, the NPR ratio is lower for CW-Prioritized compared to CW. In our implementation (as in the

original algorithm of CW), there is a $-K$ weight to the link between two parts of the same note. In the Ishigaki *et al.* implementation, this weight is -1, and thus the algorithm keeps partitioned notes in the output (see Figure 10). Despite this difference, our CW-Prioritized implementation gives good results by considering TR both on the fugues and on the pop corpus. even if it merges incorrectly some contigs (see Figure 11).

5.3 A challenging exposition passage in a fugue

Figure 12 shows the results of the algorithms on a extract of Bach's Fugue #16 – book I. This passage is quite challenging for voice separation: all the four voices enter in succession, and there is a sixth interval in the head of the subject that often put voices very close. In the last measure of the fragment, there is even a crossing of voices when the soprano is playing this sixth interval.

The algorithms behave differently on this passage, but none of them perfectly analyze it. Only CW-Prioritized predicts correctly the first three measures, especially the start of the alto voice at the first two beats of measure 12. CW selects a bad link at the third beat of measure 14, resulting in a bad prediction in measures 12/13/14 (but a high TR ratio overall). Except on the places where almost all the algorithms fail, Streamer has a correct result. Stream Segment creates many more streams, and, as expected, assigns notes that overlap in the same stream, as on the first beat of measure 12.

Finally, none of the algorithms successfully handles the voice crossing, measure 15. CW-Contigs made here its only clustering error (otherwise it has an excellent TR-prec), linking the D of the soprano with the following G of the alto. As expected, this error is found again in CW and CW-Prioritized, and Streamer also splits apart the notes with the highest pitch from the notes with the lowest pitch. At this point, Stream Segment creates streams containing both voices. Handling correctly this passage would require to have a knowledge of the patterns (including here the head of the subject with the sixth leap) and to favor to keep these patterns in a same voice, allowing voice crossing.

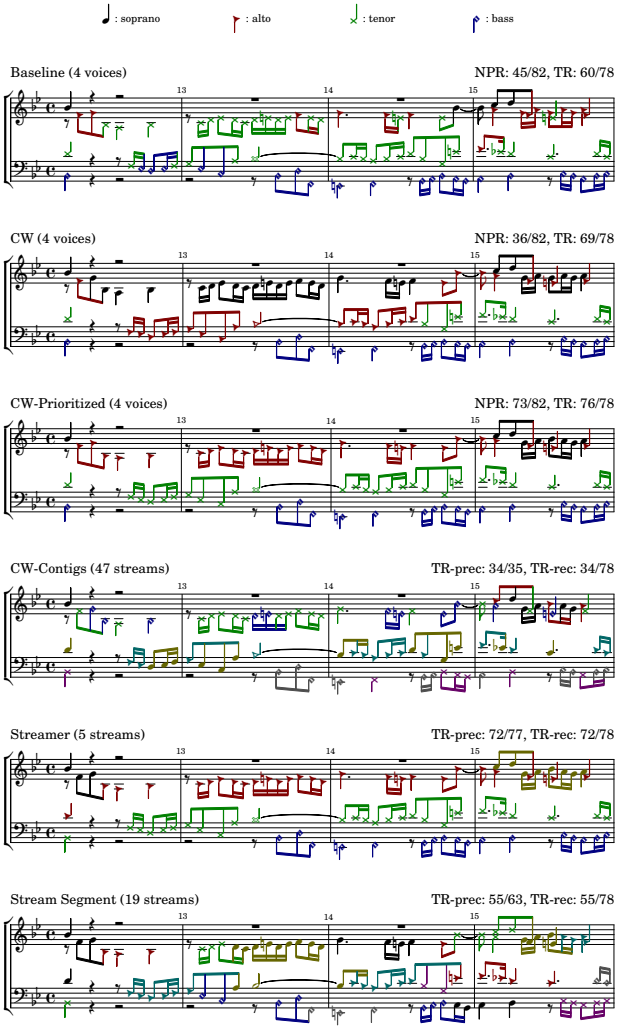

Figure 12: Output of the five algorithms on the measures 12 to 15 of Bach's Fugue #16 – book I (in G minor, BWV 861). After the initial chord with almost all the voices, the voices enter in succession (alto and tenor: m12, bass: m13, soprano: m15).

6. CONCLUSIONS

Both voice and stream segmentation methods cluster notes from polyphonic scores into relevant units. One difficulty when benchmarking such algorithms is to define a ground truth. Beside the usual fugues corpus, we proposed some ideas to establish a pop corpus with polyphonic data suitable for evaluation.

Even stream segmentation algorithms give good results in separating voices, as seen by the TR ratios and the S_u score. The Streamer algorithm is very close to a full voice separation, predicting monophonic streams. The Stream Segment algorithm further enables to output some polyphonic streams that may be relevant for the analysis of the score.

Focusing on voice separation problem, the contig approach, as initially proposed by [2], seems to be an excellent approach – very few transition errors are made inside contigs, as shown by the raw results of the CW-Contigs algorithm. The challenge is thus to do the right connections between the contigs. The ideas proposed by [9] are interesting. In our experiments, we saw a small improvement in our CW-Prioritized implementation compared to CW, but details on how partitioned notes are processed should be handled carefully. Further research should be done to improve again the contig connection.

7. REFERENCES

[1] Samer Abdallah, Katy Noland, Mark Sandler, Michael A Casey, Christophe Rhodes, et al. Theory and evaluation of a bayesian music structure extractor. In *International Conference on Music Information Retrieval (ISMIR 2005)*, pages 420–425, 2005.

[2] Elaine Chew and Xiaodan Wu. Separating voices in polyphonic music: A contig mapping approach. In *International Symposium on Computer Music Modeling and Retrieval (CMMR 2005)*, pages 1–20. Springer, 2005.

[3] Michael Scott Cuthbert and Christopher Ariza. music21: A toolkit for computer-aided musicology and symbolic music data. In *International Society for Music Information Retrieval Conference (ISMIR 2010)*, 2010.

[4] Justin de Nooijer, Frans Wiering, Anja Volk, and Hermi JM Tabachneck-Schijf. An experimental comparison of human and automatic music segmentation. In *International Computer Music Conference (ICMC 2008)*, pages 399–407, 2008.

[5] Reinier de Valk, Tillman Weyde, and Emmanouil Benetos. A machine learning approach to voice separation in lute tablature. In *International Society for Music Information Retrieval Conference (ISMIR 2013)*, pages 555–560, 2013.

[6] Diana Deutsch. Grouping mechanisms in music. *The psychology of music*, 2:299–348, 1999.

[7] David Huron. Tone and voice: A derivation of the rules of voice-leading from perceptual principles. *Music Perception*, 19(1):1–64, 2001.

[8] David Huron. Music information processing using the Humdrum toolkit: Concepts, examples, and lessons. *Computer Music Journal*, 26(2):11–26, 2002.

[9] Asako Ishigaki, Masaki Matsubara, and Hiroaki Saito. Prioritized contig combining to segragate voices in polyphonic music. In *Sound and Music Computing Conference (SMC 2011)*, volume 119, 2011.

[10] Anna Jordanous. Voice separation in polyphonic music: A data-driven approach. In *International Computer Music Conference (ICMC 2008)*, 2008.

[11] Ioannis Karydis, Alexandros Nanopoulos, Apostolos Papadopoulos, Emilios Cambouropoulos, and Yannis Manolopoulos. Horizontal and vertical integration/segregation in auditory streaming: a voice separation algorithm for symbolic musical data. In *Sound and Music Computing Conference (SMC 2007)*, 2007.

[12] Jürgen Kilian and Holger H Hoos. Voice separation-a local optimization approach. In *International Conference on Music Information Retrieval (ISMIR 2002)*, 2002.

[13] Phillip B Kirlin and Paul E Utgoff. VOISE: learning to segregate voices in explicit and implicit polyphony. In *International Conference on Music Information Retrieval (ISMIR 2005)*, pages 552–557, 2005.

[14] Hanna M Lukashevich. Towards quantitative measures of evaluating song segmentation. In *International Conference on Music Information Retrieval (ISMIR 2008)*, pages 375–380, 2008.

[15] Søren Tjagvad Madsen and Gerhard Widmer. Separating voices in midi. In *International Conference on Music Information Retrieval (ISMIR 2006)*, pages 57–60, 2006.

[16] Dimitris Rafailidis, Alexandros Nanopoulos, Emilios Cambouropoulos, and Yannis Manolopoulos. Detection of stream segments in symbolic musical data. In *International Conference on Music Information Retrieval (ISMIR 2008)*, 2008.

[17] David Temperley. *The Cognition of Basic Musical Structures*. Number 0-262-20134-8. Cambridge, MA: MIT Press, 2001.