Melodic segmentation: evaluating the performance of algorithms and musical experts

Belinda Thom[*], Christian Spevak[†], Karin Höthker[†]
Institut für Logik, Komplexität und Deduktionssysteme, Universität Karlsruhe
bthom@cs.cmu.edu, {spevak, hoethker}@ira.uka.de

Abstract

We review several segmentation algorithms, qualitatively highlighting their strengths and weaknesses. We also provide a detailed quantitative evaluation of Temperley's Grouper and Cambouropoulos' Local Boundary Detection Model. In order to benchmark the algorithms' performance against musicians, we compiled a corpus of different melodic excerpts and collected individual segmentations from 19 musicians. We then evaluated each algorithm's flexibility and practicality by running several experiments using large, hand-segmented portions of the Essen folk song collection and the melodies from our corpus.

1 Introduction

Segmenting or parsing a melody means imposing a temporal structure on a sequence of discrete pitch symbols. This structure is defined by pairs of boundaries that break the sequence up into various subsequences. It may involve different levels of hierarchy (Lerdahl and Jackendoff 1983), overlapping boundaries (Crawford et al. 1998), and unclassified areas not belonging to any segment. From a computational point of view it makes sense to simplify the task, to focus on breaking a melody into a *segment stream*, which we define as a series of non-overlapping, contiguous fragments. In addition, rather than explicitly dealing with hierarchy, we consider segment streams on two different levels of granularity: the *phrase level* and the *subphrase level*. Phrases capture large scale structure of a piece; subphrases were intended only to be more fine-grained than phrases, providing additional structure at "the next level down."

An algorithm that automatically produces a "musically reasonable" segment stream provides an invaluable tool for melodic modeling – and this is what motivates our interest in the subject – because it allows a melody to be divided into "locally salient chunks." For example, Höthker is exploring what higher-level features are required to adequately represent structure in melodies within and across different musical styles. She needs a general-purpose segmentation algorithm in order to consider features that can extend beyond fixed-size contexts. Thom also intends to integrate such an algorithm into BoB (Thom 2001) – a real-time agent designed to trade customized solos with an improvising musician – so that she may investigate whether or not the agent's interaction with a user noticeably improves when the agent's melodic representation is no longer limited to a per-bar representation.

Melodic segmentation and representation are inherently intertwined. What appears salient in a melodic surface will depend on how the sequence is encoded – is the focus rhythmic? intervallic? etc. – and how the melody is locally divided when building higher-level features. Ultimately, the development of a melody's encoding, its segmentation, and its higher-level abstractions should be combined into a single algorithm because then the cooperation and conflict between these different components could be unified. However, in the meantime, since such an algorithm has yet to be implemented, we strive to understand these components in isolation. For fixed-size melody segments, a detailed investigation of clustering algorithms and melodic encodings is already available (Höthker et al. 2001). This paper is intended to be accompany the other, focusing on general-purpose segmentation algorithms.

We begin by reviewing some segmentation algorithms, *qualitatively* highlighting their strengths and weaknesses (Section 2). The remaining sections focus on *quantitative* evaluation, exploring the behavior of Cambouropoulos' *Local Boundary Detection Model (LBDM)* and Temperley's *Grouper* in a variety of musical settings. Note that our ultimate goal is not to determine which algorithm is "better" but to obtain a feel for how these algorithms might perform when used off-the-shelf in our applications. Our quantitative experiments are presented in Section 5.

One reason that quantitative evaluation is challenging is that segmentation is *ambiguous*: there might be several (or even many) musically plausible output segmentations for a given input melody. This situation arises because many different factors – local structure (e.g. gestalt principles), higher-level structure (e.g. recurring motives, harmony), style-dependent norms and the breaking of these norms – influence the perception

[*] supported by the German-American Fulbright Commission
[†] supported by the Klaus Tschira Foundation

of salient melodic chunks. If a meaningful quantitative assessment of an algorithm's performance is to be made, the melodies' segmentation ambiguity should be explicitly considered in the evaluation. As an important precursor to our quantitative investigation, therefore, we collected segmentation data from nineteen musicians for a small corpus of ten melodic excerpts in different musical styles (Section 3.2). To quantify how ambiguous each melodic segmentation task was, the solutions for each excerpt were then compared and analyzed in Experiment A. This quantification is important, offering a baseline on how the musicians compare with respect to one another.

LBDM and Grouper both have adjustable parameters, which allows a user application to control, for example, how much rests or large intervals influence its determination of segment boundaries. When using these algorithms, an obvious practical question is how to set their parameters. We explore this question along two dimensions: flexibility and stability. An algorithm's *flexibility* reflects how well it could perform in principal, which depends on how rich its underlying hypothesis space is and how compatible this is with the segmentation processes that operate in realistic musical situations. An algorithm's *stability*, on the other hand, considers how practical an algorithm might be, evaluating to what extent the performance obtained in one musical domain (e.g. a particular style) or circumstance (e.g. a particular tune) requires a customized set of parameters. Note that there is no guarantee that a more flexible algorithm will necessarily be more stable: musically plausible behavior in one situation will only perform well in another provided the algorithm can make the necessary distinctions to segment new melodies in spite of fixed parameters.

We gain insight into an algorithm's flexibility by observing how well it performs when its parameters are tuned to two different settings. In Experiment D large, relatively homogeneous, hand-segmented subsets of the Essen folk tune collection (Schaffrath 1993) are used. Tuning also takes place per excerpt in Experiment B, which allows ambiguity in specific musical circumstances to be considered. In Experiment C, stability is measured by observing how an algorithm behaves when its parameters were tuned to the Essen collection but its behavior is evaluated with respect to the hand-segmented corpus.

2 Segmentation Algorithms

that produce segment streams have been proposed: rule- vs. memory-based, lower- vs. higher-level structure, real-time vs. non-real-time, etc. We now review several algorithms, considering both their strengths and weaknesses. An algorithm's input typically includes a melodic line, encoded as a note list, where each note is defined by a discrete pitch, an onset time, and an offset time. Its output can be viewed as a *segmentation vector*, $\vec{s} = (s_1, ..., s_i, ..., s_n) \in \{0, 1\}^n$, where a boundary is inserted between notes $i - 1$ and i if and only if s_i equals one. An algorithm might also extract "higher-level" features from its note list. For instance, a list of inter-onset intervals *(IOIs)* measures the duration between successive pitches; a list of offset-to-onset intervals *(OOIs)* measures the rests between successive note pairs; and a list of absolute pitch intervals *(APIs)* measures the absolute difference (in semitones) between adjacent pitches, where each interval's direction is ignored.

2.1 Grouper

Grouper, a part of the Melisma Music Analyzer,[1] was designed to extract melodic phrase structure from a monophonic piece of music. A fundamental assumption in Melisma is that intelligent music analysis can be computationally simulated using a purely rule-based approach. Grouper is based on a set of phrase structure preference rules adapted from Lerdahl and Jackendoff (1983). A *Gap Rule* prefers to insert boundaries at large IOIs and OOIs, a *Phrase Length Rule* prefers phrases to have a predefined number of notes, and a *Metrical Parallelism Rule* prefers to begin successive groups at parallel points in the metric structure. The first rule depends on the note list and the second depends upon the structure of a given solution. The last rule, in contrast, requires an additional input: a *beat list* specifying the melody's metric structure. Because duration values are normalized, tempo information is lost. Grouper also ignores a melody's pitch content.

The algorithm begins by calculating a gap score for each pair of notes, summing their IOI and OOI values. Next, a logarithmically scaled penalty is assigned to all boundaries that deviate from an "ideal" length. Another penalty is assigned to successive phrases that start at different metrical positions.[2] The optimal segmentation is the solution with the lowest sum of penalties. The penalties are controlled by five user-adjustable parameters: a gap weight; an ideal phrase length; a length penalty, and two metric penalties. With 2^n possible segmentations, dynamic programming is used to efficiently calculate an optimal solution.

2.2 LBDM

LBDM is motivated by the gestalt principles of similarity and proximity. It assigns a boundary strength to each pair of notes in a melody, quantifying how discontinuous the note pair is. Peaks in the sequence of strengths suggest where boundaries should be inserted. Cambouropoulos (2001) points out that LBDM is not a complete model of grouping in itself because issues

[1] Ready-to-use software is available at `www.links.cs.cmu.edu/music-analysis`.

[2] Melisma provides a module for inferring a beat list from a note list. To ensure that this input is error free, we generate it directly using a song's meter.

like musical parallelism (Cambouropoulos 1998) are not explictly handled. Nonetheless, we consider the model as is because of its elegant simplicity, which should make it straightforward to apply in real-time MIDI environments. As a result of the model's parsimony, it was also fairly easy to implement.[3]

LBDM begins by transforming the note list into three interval profiles: APIs, IOIs, and OOIs. Per profile, two rules control how the discontinuity associated with each interval is calculated. A *Change Rule* assigns each interval a value that is proportional to the rate of change between successive *pairs* of intervals. A *Proximity Rule* further scales each value proportionally to the size of the corresponding interval. Before combining the profiles into an overall weighted sum, each is normalized over its length. Boundaries are then inserted between those notes whose overall profile values correspond to local maxima. Six user-adjustable parameters include: three profile weights; two offset values;[4] and a threshold parameter. The threshold controls how large a maximum must be before it is considered as a potential boundary and has great impact on LBDM's output. It was not specified how to determine the threshold; we chose to define it as a real-valued fraction of the profile's global maximum. Only the highest peak in any region that crossed this cutoff was assigned a boundary. Other schemes, such as returning a fixed number of peaks, are also worth considering.

2.3 Other Approaches

Bod argues for a memory-based approach to segmentation and implemented such a system using probabilistic grammar techniques (Bod 2001). His most sophisticated and successful grammar performs data oriented parsing (DOP), where probabilistic trees are learned from a large training set of Essen data. Musically, DOP is interesting because it imposes a higher-level structural definition upon the model that is to be learned, biasing it to prefer solutions that contain an "ideal" number of segments. Trees are learned directly from melodies encoded in the Essen Associative Code *(EsAC)*; pitch, in relation to the tonic, and note length features are used. To implement DOP would involve significantly more work than LBDM. Because we did not have access to a complete ready-to-use version, we do not consider this model in Section 5.

Some algorithms also explicitly consider harmonic structure. One example is the phrase boundary agent used in the interactive music system *Cypher* (Rowe 1993). This agent collects information about discontinuities from a complex set of features (including register, dynamics, duration, harmony and beat) in real-time and calculates a weighted sum for each event. When the sum exceeds a threshold, the respective event is considered the beginning of a new phrase. This threshold can be adjusted dynamically according to the musical context. Two other relevant algorithms, one a rule-based system and the other based on a multi-layer neural network, were developed for automatic musical punctuation of a score (Friberg et al. 1998). Both systems receive information about pitch, duration, and melodic tension (determined with respect to the underlying harmonies) and operate on a local context of up to five notes. A primary way in which our research differs from their approach is that we focus on ambiguity whereas they focused on handling a single musician's interpretation. We do not consider any of these models in Section 5 because, for our purposes, it made sense to decouple harmonic analysis from the segmentation process. One practical reason for this decision is that the Essen database does not provide harmonic information. Furthermore, in BoB, harmonic information may not always be available, and in Höthker's research, where a modular, exploratory approach is sought, one could imagine needing segmentation as a preprocessing step for harmonic analysis.

2.4 Discussion

LBDM is a local algorithm in the following sense: its two rules never consider more than four and two adjacent notes respectively. An attractive aspect of LBDM is that it handles raw MIDI data, which means that rhythmic transcription is not required. As a result, only online peak estimation would need to be added before the algorithm could be used in real time. Using raw MIDI data is also advantageous because it provides *performed* IOIs and OOIs, articulation data that we expect to be relevant in many situations. Another advantage is that the algorithm's boundary strength calculations can provide insight into why certain boundaries are chosen over others. Although it would be non-trivial to convert this data into a probabilistic measure (Spevak, Thom, and Höthker 2002), this information might prove valuable when evaluating LBDM's performance in musically ambiguous situations.

The Grouper algorithm explicitly combines a local view, i.e. the current proposed segment, with higher-level metric structure. Grouper is able to handle raw MIDI data using additional tools provided in the Melisma package. To use Grouper in real-time would require a paradigm shift however, since dynamic programming must scan an entire melody before returning an optimal solution. For some parameter settings and melodies, we have found the given solution hard to understand (Spevak, Thom, and Höthker 2002). One cause is that Grouper's combination of local and more long-term knowledge is fairly complex: dynamic pro-

[3] Cambouropoulos recommended that we implement LBDM ourselves. This software is available at `i11www.ira.uka.de/~musik/segmentation`.

[4] These shift the API and OOI profiles above zero and were suggested by in the N.B. comment in Figure 1 of Cambouropoulos (2001). IOIs do not need an offset because they are always larger than zero.

gramming allows a solution's fitness to efficiently depend upon its most recent previous boundary as well as local metric information and duration content. We would have found it useful if Grouper had also output a profile that indicated how relevant each segment was to its overall optimization.

DOP also explicitly combines local and higher-level structure by simultaneously considering an arbitrary-length sequence of most recent notes and a preferred total number of phrases. Given a fairly large training set (around 4000 tunes), this algorithm can learn to segment similar unseen melodies (Bod 2001). Learning is an asset – it certainly makes the algorithm more flexible than a hard-wired algorithm – but reliance on a large hand-segmented training set is costly. In order for such a scheme to be of practical use, one would need to demonstrate that a model trained in one setting (e.g. Essen folk tunes) might also generalize well in other settings (e.g. music analysis, performance settings). Another drawback is DOP's reliance on EsAC, a score-based encoding that precludes the use of raw MIDI data. Raw MIDI could be transcribed, but this would ignore potentially useful articulation data. We believe DOP's greatest asset is its probabilistic basis, which provides a mechanism for explicitly handling ambiguity. For example, with this model, one should be able to estimate how likely or *fit* a solution is. This fitness could then be used to penalize a solution based on the discrepancy between the algorithm's assessment of its fitness and an external measure that quantifies its musical plausibility.

In contrast to Grouper, LBDM considers pitch content but ignores metric content. To the extent that metric structure is "more relevant" than pitch content, one could thus argue that it is unfair to directly compare the two (as we do in Section 5). At the same time, our present need is to use an algorithm off-the-shelf, and this led us to resist modifying either algorithms' underlying representation. This decision had two important consequences. First, by not extending LBDM to include a metric profile into its boundary strength calculation, the results in Section 5 provide a valuable suggestion: in many practical situations, perhaps pitch content is not as useful as metric information. Second, the unmodified algorithms have a comparable number of user-adjustable parameters (Grouper: 5, LBDM: 6) which meant that a comparable amount of time could be spent in tuning their parameters.

It is worth noting that much inconvenience can be hidden in a user-adjustable parameter, and it is easy to gloss over important free parameters that need to be specified when an algorithm is extended. For example, with one additional parameter (a weight for integrating a profile that indicates where similar motives occur), one could integrate motivic parallelism into LBDM (Cambouropoulos 1998). To implement this change, however, would require a motive clustering algorithm which would introduce its own set of adjustable parameters. A similar situation arises if learning is used. For example, as opposed to the brute-force search we use to tune LBDM's and Grouper's parameters to a specific dataset, one could imagine using an explicit learning procedure like gradient ascent. We would like to see Grouper and LBDM extended in this way, but to do so would require the specification of additional meta-parameters for controlling the learning process (when to stop learning, how much change to incorporate per learning iteration, etc.).

When learning is not incorporated, all degrees of freedom are provided by the model's user-adjustable parameters. Learning introduces more freedom, but it still relies on a (hard-wired) set of meta-parameters, a fixed input representation, and so on. Certainly, some aspects of the segmentation process are hard-wired (e.g. primitive, gestalt-based), while others rely on cultural or stylistic norms (e.g. familiarity of certain a genre, motive, or musical pattern). Given this realization, perhaps the most promising future direction for algorithmic segmentation is to combine the musically more informative features used by the more hard-wired algorithms (e.g. LBDM, Grouper, Cypher) with the more flexible, probabilistic DOP approach. The motivation for this suggestion is that with better features, less data may be needed to configure a model's free parameters.

3 Data

3.1 The Essen folk song collection

The *Essen* folk song collection is a large corpus of mostly European folk songs initiated by Schaffrath (1993) and maintained by Dahlig.[5] The Essen data is encoded in EsAC format, recording (among other things) its meter and key, and a sequence of discrete pitches and durations. Phrase boundary locations in a song are also included. These are typically modeled after a song's text phrases. Essen data has already been used to assess the performance of DOP and Grouper (Bod 2001; Temperley 2001), although not in a uniform setting.

We use the Essen segmentation as a reference to benchmark the algorithms' performance because it provides an unambiguous, single-layer phrase segmentation. A better understanding of how it was obtained is instructive. Dahlig (2002) outlined the Essen segmentation process as follows:

> "When we encode a tune without knowing its text, we do it just intuitively, using musical experience and sense of the structure. The idea is to make the phrase not too short (then it is a motive rather than a phrase) and not too long (then it becomes a musical period with cadence

[5]Approximately 6000 songs are publicly available at `www.esac-data.org`.

Subset	Songs	Notes per song	Notes per phrase
Kinder	208	39.1 ± 20.9	7.0 ± 1.3
Fink	554	58.7 ± 24.4	8.3 ± 1.6
Ballad	869	37.9 ± 14.2	9.2 ± 1.7
Kaszuby	981	45.6 ± 29.4	11.5 ± 2.9
All	2612	45.3 ± 22.6	9.7 ± 2.1

Table 1: Properties of our Essen data subsets: average ± standard deviation.

> etc.). But, of course, there are no rules, the division is and will be subjective."

Nothing about this definition precludes the musical appropriateness of more ambiguous segmentations. Their primary focus, rather, was to capture a phrase-based level of granularity.

Essen melodies are segregated into subsets with different characteristics, such as Polish folk songs or German children's songs, which allows one to investigate algorithmic performance as a function of context. We used the subsets outlined in Table 1 as a baseline on which to test an algorithm's performance. These subsets are interesting because across subsets, melodic content is quite different. Most of the subsets are also fairly large, and exhibit a characteristic balance between notes per song and notes per phrase. For a given subset, the number of songs reported in this table (column two) is smaller than the number available in the original collection, because only those songs with simple meters were included in the subsets.[6] The last data set was constructed by merging the other four together.

3.2 Human data collection

Ten musical excerpts in very different styles were chosen as the basis of the *musician corpus* listed in Table 2. Nineteen musicians with various levels of expertise – including professionals, musicologists, music teachers, and amateurs – were asked to identify "salient melodic chunks" for each excerpt, drawing upon their musical intuition to identify boundaries at the *phrase level* and the *subphrase* level. Subphrases were only required to be more fine-grained than phrases, providing additional structure at "the next level down." Instructions were kept deliberately vague. For instance, the term "motive" was not used because we wanted the musicians to draw upon their musical intuition rather than perform explicit melodic analyses. Subjects were instructed to identify each segment by placing a mark above its first note. In this way, each previous segment was defined as ending just before the next mark, which ensured that a musician's solution was always a segment stream. For each excerpt, subjects were given a minimal score of the monophonic melody including its meter and a "dead-pan" MIDI file.

Some excerpts were selected because of their ambiguous musical structure (e.g. excerpt 5 and 6), others because of their noticeable lack of structure (i.e. excerpt 7). Essen tunes were included in order to explore how unambiguous a "simple" folk tune might (or might not) be. Finally, excerpts 8 through 10 (an improvisation over a Bluegrass standard, a Pop standard, and a Bebop jazz solo) were chosen for practical reasons – when building interactive musical agents, one could imagine people wanting to be able to interact in these kinds of settings.

[6] Meters 4/4, 3/4, 2/4, 3/8, 6/8 were chosen to enable a straightforward automatic generation of the corresponding beat lists.

	Source	N_b	N_n
1	Essen song *E0356*	17	52
2	Essen song *E0547*	14	39
3	Essen song *Q0034*	12	57
4	Essen song *F0927*	8	45
5	WTC I, Fugue XV (J. S. Bach)	13	90
6	String Quartet op. 76/4, 1st mvt. (J. Haydn)	21	87
7	*Parsival, Karfreitagszauber* (R. Wagner)	11	30
8	*Grey Eagle* (Bluegrass Standard)	8	121
9	*Saving all my love for you* (Goffin & Masser)	11	52
10	*KoKo* (C. Parker)	32	190

Table 2: Musician Corpus. N_b and N_n are the number of bars and the number of notes the excerpt contains.

4 Methodology

4.1 Determining algorithm parameters

In our experiments, the algorithms are "tuned" to a particular data set by varying their adjustable parameters. Both the Essen subsets and the musician corpus are used. When we compare an algorithm's output to a segmentation from the Essen database or the musician corpus, the latter is referred to as the *reference* segmentation. The parameter set that provides the "best" overall performance on the data is chosen. On the Essen subsets, this parameter tuning should not lead to overfitting because the algorithms have only a few degrees of freedom in relation to the size of the data.

4.2 Assessing a segmentation's quality

Searching for an optimal parameter requires a measure for assessing the quality of a segmentation. The overriding problem is the small proportion of segment boundaries in the data. For instance, with respect to Essen, an algorithm would do quite well if it always predicted not to segment. In Figure 1, the first eight bars of Excerpt 2 are displayed, below which two segmentations are given.

If we assume that each position in the sequence is independent, the difference between any two binary sequences can be completely described by four numbers (Manning and Schütze 2001): The number of true positives (TP) corresponds to the number of locations where both sequences have a value of 1; true negatives (TN) are similarly defined for 0. In contrast, false positives (FP) and false negatives (FN) quantify the number of locations in which the prediction algorithm failed.

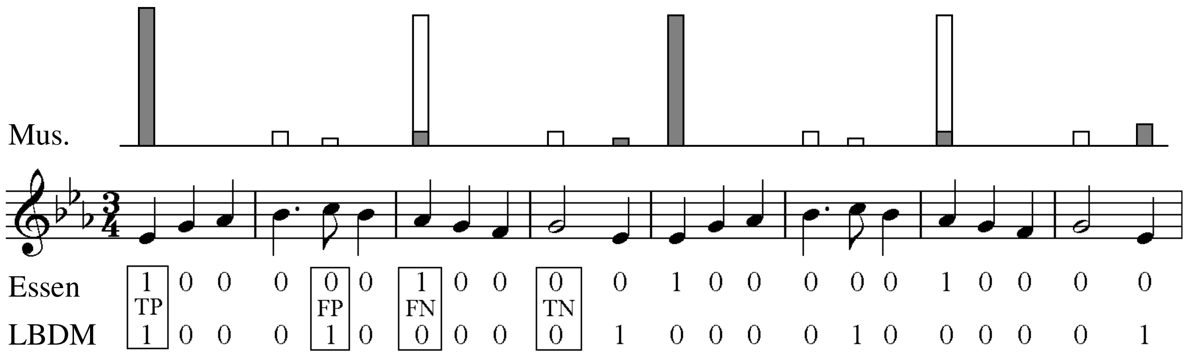

Figure 1: A Segmentation Example: Part of Excerpt 2. The Essen and LBDM segmentation vectors are shown below the score. Above the score, histograms illustrate the distribution of the musicians phrase (gray bars) and subphrase (gray and white bars) boundaries.

For clarity, in Figure 1 we highlight an example of each type of outcome.

We use the F score to assess the (dis)agreement between two segmentations because it considers the three numbers most relevant to our task:

$$\mathrm{F} = \frac{1}{1 + \frac{\mathrm{FN}+\mathrm{FP}}{2\mathrm{TP}}} \in [0, 1]$$

When no matches occur, F is 0; and 1 results when both sequences are identical. For TP=0, F is set to 0, because $\lim_{\mathrm{TP}\to 0} \mathrm{F} = 0$. F score combines both the algorithm's *precision* (the proportion of the selected boundaries that are correct) and *recall* (the proportion of the correct boundaries that were identified). The main reason for using F score is that it ignores true negatives, which would dominate the evaluation measure.

One obvious problem with this measure is that mismatches are treated in the same way regardless of their position. For example, two adjacent '1's would have the same contribution as two '1's in distant locations. One way to address this problem would be to build a probabilistic model that explicitly considered both segment positions and boundaries. We address this issue in Spevak, Thom, and Höthker (2002). In this paper, we assess the performance of the segmentation algorithms on the musician corpus as follows: for each melody, we calculate F between the algorithm's segmentation and every musician's segmentation. The average of these scores is reported.

4.3 The experiments

Four experiments were run to investigate the questions posed in Section 1. The remainder of this section describes the experiments.[7]

Experiment A: The purpose of Experiment A was to investigate the (dis)agreement between the musicians' segmentations of the ten melody excerpts in order to find out how ambiguous the segmentation task was for musicians and how much can be expected from a segmentation algorithm. The left section Table 3 shows the average F scorebetween the musicians' solutions for each excerpt.

Experiment B: The purpose here was to find out how well the segmentation algorithms could adapt to segmenting melodies of different styles. We ran Grouper and LBDM on the melodic excerpts listed in Table 2. For each melody, we searched for the parameter that maximized the average F scorebetween the computed segmentation and the musician segmentations of the same melody. The results are reported in the middle section of Table 3.

Experiment C: The purpose of Experiment C was to test how well the best parameters computed on the Essen database performed on the musician corpus in order to investigate the practicability of a fixed parameter set in different contexts. We ran Grouper and LBDM on the musician set using the best parameters from Experiment D, computed on the subset 'All' (see Table 2). The resulting F scores, averaged over all musicians, are given the right section of Table 3.

Experiment D: The purpose here was to test the performance of the segmentation algorithms on large data sets. We ran Grouper and LBDM on the five subsets of the Essen database described in Table 1. For each melody, the computed segmentation was compared to the phrase segmentation provided in the Essen collection using the F score. For each data set, the average F scoreand its standard deviation were recorded . This procedure was repeated for a variety of different parameter sets. Table 4 shows the F scores obtained for the best parameters computed on a data set. The tables will be evaluated in Section 5.

5 Discussion

How ambiguous is the segmentation task? Experiment A shows that the melody excerpts in the musician corpus are ambiguous to a varying extent. Ta-

[7] For more details concerning musician data, see `i11www.ira.uka.de/~musik/segmentation`.

	Exp A: Ambiguity		Exp B: Optimal Parameters				Exp C: Fixed Parameters			
	Musicians		Grouper		LBDM		Grouper		LBDM	
Exc	Phrase	Subphrase	Phrase	Subphrase	Phrase	Subphrase	Phrase	Subphrase	Phrase	Subphrase
1	.70 ± .27	.80 ± .16	.78 ± .25	.88 ± .15	.35 ± .15	.40 ± .14	.27 ± .17	.56 ± .14	.17 ± .22	.29 ± .16
2	.66 ± .30	.80 ± .31	.56 ± .23	.90 ± .24	.15 ± .17	.42 ± .11	.09 ± .25	.04 ± .14	.08 ± .17	.07 ± .21
3	.76 ± .16	.79 ± .24	.82 ± .14	.89 ± .19	.81 ± .15	.84 ± .23	.64 ± .09	.49 ± .14	.81 ± .15	.40 ± .11
4	.57 ± .36	.75 ± .32	.72 ± .31	.77 ± .22	.48 ± .16	.77 ± .09	.72 ± .31	.65 ± .20	.02 ± .09	.01 ± .05
5	.42 ± .27	.50 ± .23	.37 ± .16	.43 ± .09	.45 ± .17	.39 ± .15	.28 ± .19	.34 ± .10	.20 ± .20	.11 ± .09
6	.61 ± .22	.56 ± .29	.48 ± .11	.68 ± .32	.73 ± .24	.60 ± .32	.42 ± .11	.52 ± .21	.46 ± .18	.14 ± .04
7	.14 ± .25	.35 ± .26	.28 ± .23	.54 ± .29	.28 ± .32	.44 ± .27	.16 ± .23	.34 ± .23	.30 ± .26	.45 ± .25
8	.41 ± .25	.36 ± .19	.43 ± .21	.44 ± .21	.39 ± .14	.43 ± .17	.23 ± .13	.28 ± .15	.42 ± .15	.41 ± .18
9	.82 ± .14	.76 ± .16	.89 ± .12	.84 ± .13	.82 ± .13	.70 ± .17	.89 ± .12	.59 ± .10	.65 ± .13	.36 ± .06
10	.82 ± .12	.58 ± .17	.71 ± .09	.64 ± .13	.89 ± .10	.70 ± .22	.67 ± .08	.62 ± .12	.81 ± .12	.52 ± .19

Table 3: Experiments A, B and C: average F score ± standard deviation.

Data Set	Grouper	LBDM
Kinder	.64 ± .32	.51 ± .30
Kaszuby	.70 ± .35	.49 ± .34
Ballad	.60 ± .36	.51 ± .31
Fink	..65 ± .28	.56 ± .25
All	.62 ± .34	.50 ± .31

Table 4: Experiment D: average F score ± standard deviation.

ble 3 (left) presents the average inter-expert F score for each excerpt. Results range considerably, from 0.14 for the phrase segmentation of the Wagner melody (Excerpt 7) to 0.82 for *Saving all my love* (Excerpt 9) and *KoKo* (Excerpt 10). Wagner's melodic surface is very smooth, which makes it difficult to segment. The contents of the other two are much less ambiguous. *Saving all my love* has a repetitive structure and rests at verse boundaries. In *KoKo*, the lines are broken up by rests which coincide with phrase boundaries in most musician segmentations. $F = 0.58$ for subphrases indicates that finding structure within *KoKo*'s improvised lines was more difficult.

The results obtained in Experiments B and C have to be interpreted with respect to the agreement between the musicians computed in Experiment A, i.e. if the musicians disagree on the segmentation of a particular excerpt (low F score in Experiment A), the algorithm cannot possibly agree with all musicians at the same time, causing the lower F score in Experiments B and C.

How flexible is an algorithm? Experiment B provides insight into how flexible each algorithm is (Table 3, middle): Grouper tends to compute segmentations that agree better with the musicians' segmentations than those computed by LBDM (Excerpts 1 and 2), although LBDM outperforms Grouper in some cases (Excerpt 6, phrases). In Excerpt 2, LBDM received a low F score ($F = 0.15$) although the musicians mostly agreed on the locations of the phrase boundaries (Figure 1). In this example, the gestalt principle of proximity conflicts with the higher level melodic arch structure . The song starts with two identical arch-shaped phrases, each beginning and ending with the same pitch. There are, therefore, no contour-based cues in the phrase boundaries of this score. Another difficulty is that the longest durations occur in the middle of a phrase, a situation that will confuse any algorithm that relies only on the local melodic surface. For this folk tune, LBDM might improve in a performance setting, where articulation could provide some discriminatory power.

Grouper performs somewhat better ($F = 0.56$) because the melody exhibits a regular phrase structure, which is captured by the Metrical Parallelism Rule, and because pitch information is ignored. At first sight, it seems puzzling that, in some cases in Table 3, the sum of the average F score and its standard deviation is larger than 1. This effect might be due to a skewed[8] F score and to the small sample of musician segmentations in our experiment. In Table 4, which is based on larger data sets, this effect is less noticeable.

How stable is an algorithm? Experiments C and D provide insight into how practical each algorithm might be in realistic situations where the parameters cannot be adjusted to each individual example. This situation occurs, for example, in a real-time performance setting or when not enough appropriate hand-segmented data is available for parameter tuning. In Experiment C, performance was measured for individual excerpts in the corpus, but the parameters were tuned on the large "All" Essen subset (Table 3, right). As expected, the performance of the algorithms decreased in comparison with Experiment B. Interestingly, this decrease is particularly noticeable for subphrases (0.26 and 0.30 on average for Grouper and LBDM, respectively). For phrases, it is less significant (0.17 and 0.15). An explanation might be that the phrase granularity in the Essen database is more similar to the phrases in the excerpts than to the subphrases.

In Experiment D, Grouper performs better than LBDM, which seems to indicate that metrical parallelism has significant influence on the Essen folk tunes' phrase structure. However, the standard deviation is high for both algorithms and all data sets. Even for optimal parameters, the minimum F scores is 0 in most of the cases, which means that there was at least one melody in the set whose boundaries were not repro-

[8] F score might not be normally distributed.

duced at all. Therefore, the average F score does not provide a reliable prediction about the performance of an algorithm on the individual melodies, thereby questioning its stability.

Parameters. In our experiments, the best parameters found by searching confirm those recommended by Cambouropoulos. The inter-onset stream appears to be the most relevant parameter. Since our experiments are based on score-data (instead of performance data), the rest stream carries less information. In the latter case, the additional articulation information captured by the rest stream might improve the algorithm's performance. In Grouper, the length related parameters play an important role: in the best parameter settings, the length weight are always non-zero. In Experiment D, the optimal length parameter is correlated with the number of notes per phrase (Table 1).

6 Conclusion

In this paper, we reviewed several segmentation stream algorithms, and qualitatively highlighted their strengths and weaknesses. Algorithms tend to differ across their temporal processing (local vs. global), the required inputs (score vs. MIDI, real-time vs. batch, harmonic or metric information, training data), their flexibility (hard-wired vs. learned), and their ease-of-use (availability, setup time). For offline applications, Temperley's Grouper provides a well-balanced choice. Although an LBDM-style approach would be ideal in online settings, as is, it performs less well than Grouper. Performance might significantly improve if metric information were included. For such an extension to be of practical use, a ready-to-use implementation would be required.

In the remaining sections, we quantified how ambiguous specific segmentation tasks were and used this as a baseline from which to compare Grouper's and LBDM's performance. A notable aspect of our evaluation is that two radically different types of hand--segmented data were used: The first type of dataset was made up of a large, relatively homogeneous subset of folk tunes. The second, obtained from a small set of musicians, was composed of relatively distinct segmentations for the same melody. The first type has many stylistically similar songs, but individually, each song contains its own distinctive melody and "correct" segmentation. Thus, while an algorithm's user-adjustable parameters can provide the flexibility needed to significantly improve its performance on average, the variance across songs is impractically large. The second type, on the other hand, demonstrate that (even for "simple" folk songs) ambiguity is substantial, which suggests that measurements made using the first type of data should be interpreted with caution.

Acknowledgments

We are grateful to all those who volunteered to segment the melodic excerpts for us. We also would like to thank R. Bod, E. Cambouropoulos and E. Dahlig for their input.

References

Bod, R. (2001). Memory-based models of melodic analysis: challenging the gestalt principles. *Journal of New Music Research 30*(3).

Cambouropoulos, E. (1998). Musical parallelism and melodic segmentation. In *Proceedings of the XII Colloquium on Musical Informatics*, Gorizia, Italy, pp. 111–114.

Cambouropoulos, E. (2001). The *Local Boundary Detection Model (LBDM)* and its application in the study of expressive timing. In *Proceedings of the International Computer Music Conference (ICMC01)*, Havana, Cuba.

Crawford, T., C. S. Iliopoulos, and R. Raman (1998). String-matching techniques for musical similarity and melodic recognition. In W. B. Hewlett and E. Selfridge-Field (Eds.), *Melodic Similarity: Concepts, Procedures, and Applications*, Chapter 3, pp. 73–100. Cambridge, Mass.: MIT Press.

Dahlig, E. (2002). Personal communication.

Friberg, A., R. Bresin, L. Frydén, and J. Sundberg (1998). Musical punctuation on the microlevel: automatic identification and performance of small melodic units. *Journal of New Music Research 27*(3), 271–292.

Höthker, K., D. Hörnel, and C. Anagnostopoulou (2001). Investigating the influence of representations and algorithms in music classification. *Computers and the Humanities 35*, 65–79.

Lerdahl, F. and R. Jackendoff (1983). *A Generative Theory of Tonal Music*. Cambridge, Mass.: MIT Press.

Manning, C. D. and H. Schütze (2001). *Foundations of Statistical Natural Language Processing*. MIT Press.

Rowe, R. (1993). *Interactive Music Systems: Machine Listening and Composing*. Cambridge, Mass.: MIT Press.

Schaffrath, H. (1993). Repräsentation einstimmiger Melodien: computerunterstützte Analyse und Musikdatenbanken. In *Neue Musiktechnologie*, pp. 277–300. Mainz: Schott.

Spevak, C., B. Thom, and K. Höthker (2002). Evaluating melodic segmentation. In *Proceedings of the 2nd Conference on Music and Artificial Intelligence, ICMAI'02*, Edinburgh, Scotland.

Temperley, D. (2001). *The Cognition of Basic Musical Structures*. Cambridge, Mass.: MIT Press.

Thom, B. (2001). *BoB: An Improvisational Music Companion*. PhD thesis, Carnegie Mellon University, Pittsburgh, Pennsylvania.