Segmental Pattern Discovery in Music

Darrell Conklin
Department of Computing, City University London, EC1V 0HB, United Kingdom, conklin@city.ac.uk

Christina Anagnostopoulou
Faculty of Music, School of Philosophy, University of Athens, Panepistemiopolis, 15784 Athens, Greece,
chrisa@music.uoa.gr

In this paper we describe a new method for discovering recurrent patterns in a corpus of segmented melodies. Elements of patterns in this scheme do not represent individual notes but rather represent melodic segments that are sequences of notes. A new knowledge representation for segmental patterns is designed, and a pattern discovery algorithm based on suffix trees is used to discover segmental patterns in large corpora. The method is applied to a large collection of melodies, including Nova Scotia folk songs, Bach chorale melodies, and sections from the Essen folk song database. Patterns are ranked using a statistical significance method that integrates pattern self-overlap, length, and frequency in a corpus into a single measure. A musical interpretation of some of the statistically significant discovered patterns is presented.

Key words: data mining; pattern discovery; music analysis; knowledge representation
History: Accepted by Elaine Chew, Guest Editor for the Special Cluster on Computation in Music; received March 2004; revised August 2004; accepted November 2004.

1. Introduction

The discovery of recurrent and significant patterns in data is a fundamental form of data mining. In music, there are many motivations for the discovery of patterns. In the analysis of a single piece, repeated patterns may reveal the construction of the piece in terms of repetitive basic structures such as themes, motives, harmonies, or pitch class sets. The discovery of recurrent patterns that occur across many pieces in a specific musical style can yield patterns that may be used in stylistic analysis, synthesis, and classification of new pieces in the style. Patterns also contribute to the definition of large-scale musical form.

Modern computational methods for pattern discovery in music have been used to reveal recurrent patterns in a single or small number of pieces (Rolland and Ganascia 2000, Hsu et al. 2001, Cambouropoulos 1998, Cope 1991, Lartillot 2004) and large stylistic corpora (Conklin and Anagnostopoulou 2001). Their applicability and success depend on the way pieces of music are structured and on the formal language used to describe musical patterns.

Current methods for pattern discovery, while useful for finding melodic patterns, lack the ability to represent patterns whose elements refer not simply to notes but also to melodic segments that are sequences of notes. Such *segmental patterns* can be used to represent patterns of recurrent segment types discovered within a corpus. In this paper, the third of a series on pattern representation and discovery in

music, we demonstrate that a knowledge representation method used previously for melodic (Conklin and Anagnostopoulou 2001) and vertical (Conklin 2002) pattern discovery can naturally be used for segmental patterns.

One way to evaluate a proposed pattern representation is to ask whether pieces in a large and diverse melodic corpus have significantly recurrent patterns. The pattern discovery algorithm used here, which is based on a suffix tree algorithm, has been applied previously to the discovery of melodic patterns (Conklin and Anagnostopoulou 2001). Here it is applied to segmental patterns, using a generalized knowledge representation and an improved method for computing the statistical significance of patterns.

The method is applied to a large collection of melodic data sets, including Nova Scotia folk songs (Creighton 1966), Bach chorale melodies, and sections from the Essen folk song database (Schaffrath 1995). Many statistically significant patterns, spanning across musical styles, have been discovered, and a musical interpretation of a few of these patterns is discussed.

2. Methods

Music analysis is concerned with understanding pieces of music by identifying their constituent structures and how these are transformed as a piece unfolds in time. Analytic decisions are based on different properties of a piece, and quite often analysts

will concentrate on only a few properties at a time, such as rhythm, melodic contour, or cadential structure. An analysis will depend on the way in which a piece is segmented or structured and on the properties chosen to represent its constituent structures.

The approach for segmental pattern representation described in this paper is partially motivated by the technique of *semiotic analysis* (Nattiez 1975), where similar segments are grouped together into classes according to their shared properties (*paradigmatic analysis*) and a piece is viewed as a sequence of paradigmatic class labels (*syntagmatic analysis*). The topic of formal paradigmatic analysis of music has also been approached using automated clustering methods (Anagnostopoulou and Westermann 1997, Cambouropoulos 1998, Höthker et al. 2001, Grilo et al. 2001). A difference between the present approach and previous investigations into computational paradigmatic analysis is in the technique used to represent segment classes. In our approach, rather than adjusting the distance thresholds and other parameters of a clustering procedure, the analyst focuses on adjusting the level of abstraction used to describe segments so that those posited to be in the same class are equivalent in the representation.

To illustrate the basic idea of semiotic analysis, Figure 1 presents a fragment of a popular Scottish traditional melody. Although other segmentations of this melody are possible, for illustration purposes we have chosen to structure the melody as four two-bar segments, each with twelve beats. An analyst approaching this fragment using semiotic analysis may decide to group segments 1 and 3 together by virtue of their equivalent rhythmic values and melodic contour. Segments 2 and 4, on the other hand, are similar but not equivalent in either of these dimensions, and their grouping will depend on their measured similarity or on their equivalence with respect to more abstract features. Thus the overall analysis of this melodic fragment might be written as ABAC or ABAB, where A, B, and C are paradigmatic class labels.

In this research, the representation for segmental patterns is used as a basis for the discovery of recurrent sequences of segments in a corpus.

2.1. Music Objects and Structuring of Pieces

To encode pieces of music, we use an algebraic data type that permits a hierarchical structuring of a piece of music. A *music object* is a note (type **Note**), or (recursively) a sequence (type **Seq**), or simultaneity (type **Sim**) of music objects joined or layered together. Music objects have *basic attributes*: for example, all **Note** music objects have a *pitch*. When a music object is joined or layered as a component of another music object, it acquires an *onset time*, and becomes an *event*. All music objects have a *duration*: for sequences and simultaneities the duration is computed by inspecting the durations of component events. For this work we measure duration in terms of 96th note ticks: this permits a convenient integer encoding of various triplet durations (e.g., a note in a triplet of eighth notes has duration eight; a note in a triplet of sixteenth notes has duration four). In this paper, we consider only those regular sequences where all events within the sequence are of the same type. The type of a regular sequence of events of type X is denoted **Seq(X)**. For example, a sequence of notes has type **Seq(Note)**, and a segmented melody has type **Seq(Seq(Note))**. A **Seq** of n music objects, or more generally an ordered sequence of n items, can be written as $[e_1, \ldots, e_n]$ and will sometimes be abbreviated as $\overline{e_n}$.

2.2. Knowledge Representation

A piece of music is described using music objects such as notes, sequences, and simultaneities. The properties of these objects—properties at a higher level of abstraction than the basic musical surface—can be represented and inferred using the *viewpoints* knowledge representation method.

2.2.1. Viewpoints. A *viewpoint* is a partial function that computes attribute values or *viewpoint elements* for events in a sequence. In music these values represent properties of musical events. More precisely, for a viewpoint τ and sequence of events $\overline{e_n}$, if $\tau(\overline{e_n}) = v$, then the event e_n is inferred to have the attribute value v in the context of the sequence $\bar{e}_{n-1}$. Since viewpoints are functions of an event and its contextual sequence, they can compute attributes representing *relations* between an event and any number of events in the context. A viewpoint is a partial rather than a total function, meaning that it can be undefined for some events in sequence; in this case it returns the special value $\perp$. For example, a melodic interval viewpoint will be undefined for the first event in a sequence of notes.

The top part of Table 1 shows the formal schema for viewpoints. The *domain* of a viewpoint is always of the form **Seq(X)**, where X is a variable that can refer to any type of music object. A viewpoint τ computes attribute values in the set $[\tau]$ (called the *range* of the

Figure 1   A Fragment of the Scottish Traditional Melody *All The Blue Bonnets Are Over The Border* Grouped into Four Segments

Table 1   Abstract Syntax and Informal Semantics of the Viewpoints Knowledge Representation Scheme for Music

Viewpoint function type	Description
τ: **Seq**$(X) \rightarrow [\tau]$	General viewpoint schema
contour: **Seq(Note)** $\rightarrow \{-, =, +\}$	Melodic contour
pcint: **Seq(Note)** $\rightarrow \{0, \ldots, 11\}$	Pitch class interval
intref: **Seq(Note)** $\rightarrow \{0, \ldots, 11\}$	Pitch class interval from a reference pitch
beats: **Seq(Seq(Note))** $\rightarrow \mathbb{Z}^+$	Beats in segment
shape: **Seq(Seq(Note))** $\rightarrow \mathbb{H}$	Segment shape type
link(τ_1, τ_2): **Seq**$(X) \rightarrow [\tau_1] \times [\tau_2]$	Linking two viewpoints with domain **Seq**(X)
f: **Seq**$(X) \rightarrow \mathbb{B}$	f is a Boolean viewpoint, returning true or false
select(τ, f): **Seq**$(X) \rightarrow [\tau]$	The τ-viewpoint element of an event if f is true
ratio(τ): **Seq**$(X) \rightarrow \mathbb{Q}$	The ratio of two τ-viewpoint elements
new(τ): **Seq**$(X) \rightarrow \mathbb{B}$	True when the τ-viewpoint element changes
lift(τ): **Seq(Seq**(X)**)** $\rightarrow [\tau]^*$	The viewpoint sequence for a segment
pair(τ): **Seq(Seq**(X)**)** $\rightarrow \mathscr{P}([\tau])$	The set of all τ-relations between two segments
set(τ): **Seq(Seq**(X)**)** $\rightarrow \mathscr{P}([\tau])$	The set of τ-viewpoint elements occurring in a segment
g: **Seq**$(X) \rightarrow X$	g selects an event from a sequence
thread(τ, g): **Seq(Seq**(X)**)** $\rightarrow [\tau]$	Thread over g-selected events in segments

viewpoint) for events of type X. A viewpoint with domain **Seq(Note)** is called a *melodic viewpoint*, while a viewpoint with domain **Seq(Seq(Note))** is called a *segmental viewpoint*. Although not discussed further in this paper, a viewpoint with domain **Seq(Sim(Note))** is called a *vertical viewpoint* (Conklin 2002).

In addition to describing individual events, for knowledge representation and pattern discovery in music it is necessary to describe a complete piece of music by a *sequence* of event attributes. A τ-*viewpoint sequence* is simply the successive application of τ to each of the prefixes of a sequence of events $\overline{e_n}$:

$$[\tau(\bar{e}_1), \ldots, \tau(\overline{e_n})] \qquad (1)$$

with the constraint that no $\perp$ (undefined) elements are present in the viewpoint sequence.

2.2.2.   Primitive Viewpoints. Viewpoints can be further categorized as being either *primitive* or *composite*. Primitive viewpoints are computed directly from the musical surface and are described only in terms of basic attributes (e.g., pitch, duration, onset time) and mathematical functions. Some primitive melodic and segmental viewpoints are presented in the second part of Table 1 ($\mathbb{Z}^+$ denotes the positive integers, and $\mathbb{H}$ denotes the set of segment shape types):

contour the melodic contour between an event and its preceding event;

pcint pitch class interval: the melodic interval in a modulo 12 system;

intref interval from a reference pitch (the key of a piece), in a modulo 12 system;

beats the number of beats in a segment, between the downbeat of the segment until the end of the last bar of the segment;

shape the overall shape of a segment computed and classified into one of nine segment types using the method of Huron (1996).

2.2.3.   Composite Viewpoints and Constructors. In contrast to primitive viewpoints, composite viewpoints are created from other viewpoints using higher-order functions called *constructors*; these are functions that take viewpoints as arguments and return new viewpoints. There are two kinds of constructors: *type-preserving* constructors that create a viewpoint having the same domain as the argument(s) of the constructor and *type-changing* constructors that create a viewpoint applying to a different type of music object. For example, four type-preserving constructors found in the third part of Table 1 ($\mathbb{B}$ denotes the Booleans, and $\mathbb{Q}$ denotes the rational numbers) are:

link(τ_1, τ_2) computes a tuple comprising the separate application of viewpoints τ_1 and τ_2;

select(τ, f) selects the τ-viewpoint element if the Boolean viewpoint f returns true;

ratio(τ) computes the ratio of the τ-viewpoint elements for an event and its preceding event. The viewpoint τ must return integer values, and the ratio is expressed as a rational number in normal form;

new(τ) true if the τ-viewpoint element of an event has changed from that of its preceding event.

Four type-changing constructors that we have developed for constructing new segmental viewpoints from melodic viewpoints can be found in the bottom part of Table 1 ($\mathscr{P}$ denotes the power set function):

lift(τ) computes the τ-viewpoint sequence (1) for a segment. In this way the whole τ-viewpoint sequence is treated as an attribute value of a segment;

pair(τ) computes the set of all τ-relations between every event in one segment and every event in the preceding segment. It is a generalization of Lewin's (1987) *interval function*, which computes the frequency of occurrence of every pitch class interval between the notes in two segments;

set(τ) computes the set of τ-viewpoint elements occurring in a segment (in contrast to lift(τ), which computes a *sequence* of τ-viewpoint elements);

thread(τ, g) computes the τ-relation between events selected from segments using the function g. Selected events from successive segments together form a new event sequence, over which the viewpoint τ computes viewpoint elements.

Viewpoint constructors can be applied to both primitive and composite viewpoints. This means that from a given set of primitive viewpoints and a repertoire

Viewpoint	Viewpoint sequence
pitch	[72, 71, 69, 67, 67, 69, 71, 72, 74, 72, 71, 69, 71, 72, 69, 67]
duration	[24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 12, 12, 48, 24]
contour	[−, −, −, =, +, +, +, +, −, −, −, +, +, −, −]
pcint	[11, 10, 10, 0, 2, 2, 1, 2, 10, 11, 10, 2, 1, 9, 10]
intref	[0, 11, 9, 7, 7, 9, 11, 0, 2, 0, 11, 9, 11, 0, 9, 7]
select(contour, new(contour))	[−, =, +, −, +, −]
ratio(duration)	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1/2, 1, 4, 1/2]
new(contour)	[1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0]
beats	[4, 4, 8]
shape	[descending, ascending, descending]
lift(contour)	[[−, −, −], [+, +, +], [−, −, −, +, +, −, −]]
lift(select(contour, new(contour)))	[[−], [+], [−, +, −]]
pair(pcint)	[{0, 1, 2, 3, 4, 5, 7, 8, 9, 10, 11}, {0, 1, 2, 3, 4, 5, 7, 8, 9, 10, 11}]
set(intref)	[{0, 7, 9, 11}, {0, 7, 9, 11}, {0, 2, 7, 9, 11}]
thread(pcint, last)	[5, 7]
thread(contour, highest)	[=, +]
ratio(beats)	[1, 2]
link(beats, thread(pcint, last))	[<4, 5>, <8, 7>]

Figure 2   Examples of Viewpoint Sequences for the First Three Phrases of a Bach Chorale Melody (BWV 255)

of viewpoint constructors it is possible to construct a very large number of new viewpoints to describe music objects.

2.2.4.   A Music Example. As an illustration of the viewpoint representation for music, Figure 2 shows some primitive and composite melodic and segmental viewpoint sequences for a three-phrase melodic fragment. The top half of Figure 2 shows some melodic viewpoints, each viewing the fragment as a **Seq(Note)** object. The bottom half shows some segmental viewpoints, each viewing the fragment as a **Seq(Seq(Note))** object, segmented on phrase boundaries. The melodic and segmental viewpoints have been further subdivided into primitive and composite viewpoints. All notes in the melody have the basic attributes of pitch (represented here by MIDI note number) and duration (measured in 96th note ticks). The pcint viewpoint measures the melodic interval modulo 12, hence for the second note, the pitch class interval is $(71 − 72) \bmod 12 = 11$. The select(...) viewpoint shows that the contour of the melody changes direction six times. Note that for several segmental viewpoints there are only two elements in the viewpoint sequence. For example, the first thread(pcint, last) element in the sequence is five: the first segment has no previous event and, therefore, its viewpoint element is undefined and does not appear in the viewpoint sequence. A similar phenomenon

can be seen for the primitive melodic viewpoints pcint and contour and for all composite melodic viewpoints in Figure 2.

A few points can be made regarding the example segmental viewpoint sequences. The beats viewpoint shows the number of beats in each of the three segments, and the shape viewpoint their broad shape class. A more general description of beats is given by the ratio(beats) viewpoint sequence: two segments with the same number of beats (ratio: 1) followed by one twice as long (ratio: 2). A more specific description of melodic shape is given by the lift(select(...)) viewpoint which, for notes in a segment, selects the contour value only where the contour changes. The pair(pcint) elements contain no interval of six: there is no tritone (6 semitones) between any two notes in a segment or its preceding segment. The first and second segments have the same set(intref) element, and there is no third (interval: 4) or fourth (interval: 5) scale degree used in any of the three segments. The link(...) viewpoint is a simple example of linking together two viewpoints; note that because one of its component viewpoints thread(pcint, last) is undefined for the first segment, the composite linked viewpoint is also undefined for the first segment.

2.3.   Patterns and Pattern Statistics
A *pattern* is a viewpoint sequence fragment encountered in a corpus of pieces. Whereas music objects

and viewpoint sequences represent individual pieces, patterns represent abstractions, or concepts. A pattern *occurs* in a piece (the piece is an *instance* of the pattern) if it is contained in the viewpoint sequence for the piece. A *melodic pattern* is one based on a melodic viewpoint, whereas a *segmental pattern* is one based on a segmental viewpoint.

The *piece count* of a pattern is the number of pieces in which the pattern occurs. The *total count* of a pattern is its total number of occurrences in a corpus, including possibly overlapping repetitions within a single piece. To evaluate patterns, a statistical hypothesis-testing method is employed. The *p-value* of a pattern with total count k is the probability that it occurs k or more times in a corpus of the same size; lower *p*-values indicating more surprising patterns. To compute *p*-values, it is necessary to use a null model; this model is presumed to generate "random" music unlike that of the corpus. The null model we use is a unigram model with method C smoothing (Jurafsky and Martin 2000), constructed from counts of viewpoint elements in the corpus. Smoothing—the allocation of some probability space to elements that are not encountered in the corpus—is necessary because element counts for complex composite viewpoints may be sparse and unreliable. The *probability* of a pattern is its probability of occurrence in a viewpoint sequence, generated by the null model, of the same length as the pattern.

To compute *p*-values with the null model, it is important to consider the effect of self-overlap in patterns. Patterns with a relatively high self-overlap will tend to be generated in "clumps," and it will not be as surprising to see more occurrences than for a pattern with a similar probability but relatively low self-overlap. To quantify the amount of self-overlap, the *period* of a pattern P is defined to be the length of its shortest subpattern Q such that P is a prefix of Q^m for some integer $m > 0$ (Apostolico and Crochemore 2002). Patterns with a high self-overlap have a low period, and conversely those with no self-overlap have a period equal to their length.

By simulation on data generated by the null model, we have determined that the distribution of pattern counts can be fit by a normal distribution with a standard deviation that depends on the pattern length and period. For a pattern with probability p, the expected total count μ is np where n is the total length of all τ-viewpoint sequences from the corpus. The standard deviation σ of the total count is

$$\sqrt{\mu(1-p)l/q} \qquad (2)$$

where l is the length of the pattern and q is the pattern period: the fraction l/q is an adjustment factor that quantifies the degree of self-overlap of the pattern. As desired, the standard deviation (2) of the pattern total count increases with the amount of pattern self-overlap.

The *z-score* of a pattern is $(k-\mu)/\sigma$, and represents the difference of the total count k from the expected total count μ, expressed in units of standard deviation (2). The *z*-score can be used directly to rank patterns, or if desired it can be converted to a *p*-value using the standard normal distribution.

As an example of patterns and pattern statistics, Figure 3 illustrates the Austrian folk song *Muede kehrt ein Wanderer zurueck nach seiner Heimat seiner Liebe Glueck* (Essen D1016) and the Swiss folk song *Muede kehrt ein Wandersmann zurueck nach der Heimat sehnt sich seinen Blick* (Essen D1005). The similarity between the pieces is quite apparent, with very similar rhythmic patterns, melodic contour and implied harmonies, although there is not a basic melodic property that is conserved over the span of the two melodies. By mining a large collection of melodies (see §3), a pattern for the set(intref) viewpoint was discovered to be shared between these two pieces (Figure 3). The pattern has length $l = 4$ and period $q = 2$; when shifted to the right by two positions, it can potentially occur again. For this viewpoint and pattern, the total viewpoint sequence length is $n = 5,955$, the pattern probability is $p = 1.31 \times 10^{-6}$, and the pattern total count is $k = 3$. These values in the expressions above yield a *z*-score of 24, indicating a highly significant pattern.

2.4. Pattern Discovery Algorithm

The pattern discovery algorithm we employ reports, for a specified viewpoint τ and *p*-value threshold, all significant patterns discovered in a corpus. For a specified viewpoint τ, the τ-viewpoint sequence (1) for every piece in the corpus is computed, and these

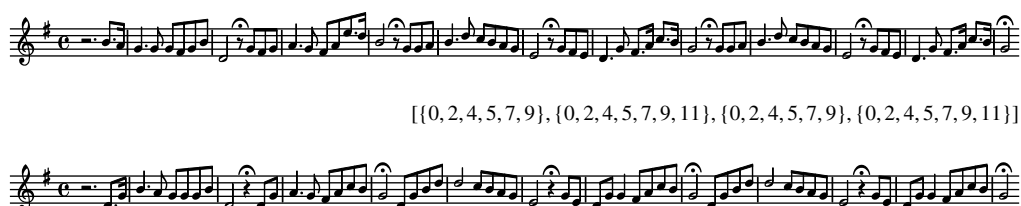$$[\{0,2,4,5,7,9\}, \{0,2,4,5,7,9,11\}, \{0,2,4,5,7,9\}, \{0,2,4,5,7,9,11\}]$$

Figure 3 **Two Pieces from the Essen Folk Song Database That Are Instances of a set(intref) Pattern**

sequences are used to build a *suffix tree*. The suffix tree data structure is useful for efficient discovery of common substrings in text processing and bioinformatics applications (Gusfield 1997) and is generally useful for many areas where data can be represented as sequences.

Following the suffix tree construction process, the tree is traversed to reveal all patterns with a total count of at least two. The resulting list is filtered for significant patterns, ordered by increasing p-value (or equivalently, by decreasing z-score), and reported to the analyst.

2.5. Piece Weighting

In any data analysis task it is important to avoid sample bias, or at the very least consider its effect on analysis results. In the data sets used in this study, there are some pieces (particularly in the German *kinder* songs) with extensive duplication of musical material. Including highly similar pieces in the analysis corpus will inflate some pattern counts and therefore overestimate their statistical significance. However, rather than discard outright such pieces from the corpus, a piece weighting scheme was devised whereby they can be retained.

To weight pieces, it is assumed that a long sequence of identical intervals and durations indicates not melodic similarity but rather identity (possibly transposed) of musical material. Therefore, all significant melodic patterns for the viewpoint link(pcint, duration), of length at least ten, were found in the data set using the pattern discovery algorithm. Then, for a cluster of c pieces covered by a pattern, every piece in the cluster is given a weight of $1/c$. If a piece is a member of more than one cluster, its weight is set to the minimum of its cluster weights.

Piece weights need only be computed once and are used during all subsequent pattern discovery runs to reduce the total count k of some patterns. For example, a pattern occurring once within each of three pieces, two with weight 0.5, will receive a reduced total count of $k = 1 + 0.5 + 0.5 = 2$ rather than $k = 3$. The weighting procedure will correctly reduce the significance of patterns arising mainly due to extensive duplication of musical material. This is because the total counts for these patterns will be reduced, and the difference from their expected total count will be smaller, leading to a lower z-score.

3. Results

To evaluate a general knowledge representation scheme for music, two quite different approaches can be taken. One approach could take a small number of works, posited to be related, and find a description that reveals the relationships among their constituent structures. The approach chosen for this work

Table 2 Melodic Data Sets Used, with (Unweighted) Total Piece, Segment, and Note Counts: Mean Number of Segments Per Piece and Notes Per Segment

Description	Pieces	Segments	Notes	Segs/Piece	Notes/Seg
Nova Scotia folk songs (CCARH)	152	885	8,551	5.8	9.7
Bach Chorale melodies (CCARH)	185	1,141	9,227	6.2	8.1
Alsatian folk songs (Essen)	91	558	4,496	6.1	8.1
Yugoslavian folk songs (Essen)	119	303	2,691	2.5	8.9
Swiss folk songs (Essen)	93	518	4,586	5.6	8.9
Austrian folk songs (Essen)	102	604	5,132	5.9	8.5
German folk songs (Essen) (kinder dataset)	213	1,201	8,393	5.6	7.0
Chinese folk songs (Essen) (shanxi dataset)	237	745	11,056	3.1	14.8
Total	1,192	5,955	54,132		

is a purely exploratory or data mining approach that does not at the outset assume musical significance of descriptions in the representation but rather attempts to reveal this by inspection of recurrent patterns discovered in a large corpus of music. In this section we present some preliminary results from applying the segmental pattern representation and discovery method to a large corpus.

A collection of melodic data sets (Table 2) in the *kern* format (Huron 1997) was used to develop and test the method for segmental pattern discovery. Most of these data sets are from the Essen folk song database (Schaffrath 1995); in addition, we used 185 Bach chorale melodies and a collection of 152 Nova Scotia folk songs (Creighton 1966), both available from the Center for Computer Assisted Research in the Humanities (CCARH: www.ccarh.org). Although the Essen and Bach data sets have phrase boundaries and fermatas annotated, the Nova Scotia folk song encodings lack these markings. For this study, phrase boundaries were manually added to the Nova Scotia folk songs by comparing the score data with the lyrics and entering a phrase boundary at the end of every line of lyrics. The corpus comprises 1,192 pieces and 5,955 segments; although the complete Essen database has many more pieces, we have decided to work initially with this more compact corpus.

The pooling of such varied data sets may seem not ideal for a pattern discovery task; however, all of the melodies belong to a much broader musical style of tunes to be sung by both experts and non-experts. The melodies are highly tonal, with clear phrase boundaries, few passing modulations, and most importantly, comfortable melodic lines with easily sung leaps, simple rhythmic structures, and confined register. Chinese folk songs are also tonal, since

the pentatonic scale used can be thought of as a further restriction of the Western diatonic scale. Our results suggest that the segmental pattern approach is general enough to reveal patterns across different sub-styles and may therefore hint toward a more unified music theory of sung melodies.

To discover segmental patterns, pieces were first converted from the *kern* representation into a **Seq(Seq(Note))** music object representation (§2.1), segmented on annotated phrase boundaries. In pieces containing repeats with two alternative endings, only the second ending was chosen. The piece weighting method (§2.5) was applied to the collection of pieces. The pattern discovery algorithm (§2.4) was used to discover recurrent segmental viewpoint patterns in these data sets. Patterns for various viewpoints were discovered in the large melodic data set and sorted by decreasing z-score, as described in §2.3. A p-value threshold of 0.01 was used; patterns with a p-value above this threshold are not reported.

A few general properties of the pattern discovery method were observed. First, some significant patterns have many occurrences in the corpus: these are patterns whose probabilities (§2.3) are quite high and must therefore occur many times in the corpus in order to achieve statistical significance. On the other hand, some patterns are significant despite their low total count in the corpus, because their probabilities are quite low. The statistical significance measure therefore appropriately balances the conflicting wishes for long patterns and frequent patterns. Second, it was observed that a set of discovered patterns often exhibited extensive subsumption; many significant discovered patterns were fully contained within other significant patterns in the set. However, experiments with filtering patterns by retaining only the most general significant patterns were not successful: the filtered output then tended to comprise mainly short patterns with many instances, yet with weak statistical significance hovering around the upper p-value threshold boundary. Therefore, the complete list of all significant patterns is reported, and an analyst can later decide on the appropriate level of generality for chains of subsuming patterns.

The thread(contour, highest) viewpoint results in Table 3 illustrate the two points above. This viewpoint measures the melodic contour between the highest events in adjacent segments. A total of 18 significant patterns were discovered for this viewpoint: the most significant was the pattern $[+, -]$, with a total count of $k = 663.0$. This pattern has a simple musical interpretation: it covers three successive segments, and the note with the highest pitch in the middle segment is approached (contour value $+$) and left (contour value $-$) by either a step or leap. As indicated in Table 3, the pattern $[-, +]$ is also significant in the corpus. The five thread(contour, highest)

Table 3   Some Results of Segmental Pattern Discovery

Viewpoint	Pattern	k	p	z-Score
thread(contour, highest)	$[+, -]$	663.0	0.1	9
	$[-, +]$	562.9	0.1	4
	$[+, -, +]$	234.7	0.04	5
	$[+, -, +, -]$	117.8	0.01	6
	$[-, +, -, +, -]$	46.3	0.005	3
thread(pcint, last)	$[5, 7]$	139.0	0.02	5
	$[7, 5]$	161.2	0.02	8
	$[5, 7, 5]$	54.8	0.003	8
set(intref)	$[\{0, 2, 4, 7, 9\}, \{0, 2, 4, 7, 9\}]$	133.0	0.005	14
pair(pcint)	$[\{0, 1, 2, 3, 4, 5, 7, 8, 9, 10\},$	66.9	0.003	14
	$\{0, 2, 3, 4, 5, 7, 8, 9, 10, 11\}]$			

Note. The viewpoint, pattern, weighted total count k, pattern probability p, and the pattern z-score are indicated.

patterns in Table 3 also illustrate a partial chain of three subsumed patterns, ordered from general to specific. Although the very general pattern $[+, -]$ has high probability, it has many instances to compensate for this and receives the highest z-score. As the subsumption chain is traversed from general to specific, the pattern probabilities become lower, yet the total count also falls. The z-score follows neither the pattern specificity nor the pattern probability. This illustrates the complex relationship between pattern probability, pattern total count, and pattern significance.

The thread(pcint, last) viewpoint is similar to thread(contour, highest) in that it applies a specified viewpoint to selected notes of adjacent segments. A total of 281 significant patterns were discovered for this viewpoint. The three patterns presented in Table 3 reflect in tonal terms the alternating melodic motions of a perfect fifth up (interval: 7) and perfect fourth down (interval: 5). This is an expected motion in melodic analysis, reflecting cadential structure, and is often an empirical rule for songwriting or for melody completion in the various music theory exams.

A number of very significant patterns were found using the set(intref) viewpoint, which describes the set of all intervals from the key of the piece. Table 3 shows one pattern discovered: each element of this pattern refers to the five notes of a major pentatonic scale. The pattern thus represents two successive phrases, both using only the notes from a pentatonic scale. With only a few exceptions, the instances of this pattern are from the Shanxi folktune section of the Essen database. This illustrates the ability of the pattern representation and discovery method to reveal style-specific patterns within a corpus.

The pair(pcint) viewpoint compares a segment with its predecessor, and collects all pitch class intervals between all notes of each segment. The pattern presented in Table 3 therefore covers three successive

segments; between the first two segments no interval of 6 (called a tritone) or 11 is permitted, and between the last two segments no interval of 1 or 6 may occur. Though we find highly statistically significant patterns for this pair(pcint) viewpoint, it is hard to assess the musical validity of the discovered patterns. This musical relationship between successive segments would seem to be difficult for a listener to detect, apart from perhaps noting the scale degrees omitted from the various segments.

4. Discussion

This paper has presented an approach for the representation and discovery of segmental patterns in music using viewpoints. Melodic segments are grouped together by virtue of their equivalent feature encodings from a particular viewpoint of representation.

To reveal recurrent sequences of equivalence classes of melodic segments, a pattern discovery method was applied to a large collection of segmented folk songs and Bach chorale melodies. For each pattern a *p*-value, which takes into account pattern self-overlap, is computed using a null model. Although having high statistical significance, the instances of many patterns found do not have an immediate or obvious musical similarity and require careful analysis. This mismatch between musical and statistical significance is to be expected from a pattern ranking approach that is purely statistical in nature.

Several avenues for future research into segmental patterns can be indicated. The topic of music classification using predictive style-specific patterns— those that occur predominately within pieces of a single style—has been explored (Sawada and Satoh 2000) but not within the context of segmental patterns. Predictive segmental patterns could be used, for example, for the classification of folk songs into stylistic categories or for generation of new melodies in a style. Predictive segmental patterns, discovered in an annotated corpus, could be used to guide the segmentation of new unsegmented melodies.

A statistical significance method for evaluating segmental patterns was used in this research to rank discovered patterns. A desirable property of this method is a new measure for melodic similarity: two pieces are considered similar if they contain a common segmental pattern with a low *p*-value. One can envision music information retrieval applications of segmental patterns, based on pattern discovery and statistical significance. For example, an analyst could express a query in terms of a segmented melody and the significance of discovered patterns could be used to rank the retrieved query results. In a related sense, this similarity measure between pieces may allow a large

database to be clustered into related groups based on shared segmental patterns.

Segmentation of melodies is a very subjective process, and listeners will group notes into phrases in different ways. This work has relied on pre-segmented melodies, annotated by a variety of different editors and schemes, and has not considered the issue of alternative or optimal segmentations. An objective computational approach to segmentation into phrases could be applied, but if the segmentation was itself based on repetition, this would introduce a problematic circularity between score segmentation and pattern discovery.

The method of mining a large music corpus presented here shares many similarities with the methods and objectives found in traditional music analysis. Music analysis is founded on musical properties, and different properties will give different analytical results. This is why there are always many potential answers to the same music analysis problem. The ambiguity in music analysis, where results depend on the chosen music properties, is also reflected in the work reported here, where the consideration of different segmental viewpoints would produce different significant patterns. There is no optimal viewpoint from which to look at music, and this repertoire of segment descriptors, with simple mechanisms to construct new ones, will be valuable for computational music analysis.

References

Anagnostopoulou, C., G. Westermann. 1997. Classification in music: A computational model for paradigmatic analysis. *Proc. Internat. Comput. Music Conf.* Thessaloniki, Greece, 125–128.

Apostolico, A., M. Crochemore. 2002. String pattern matching for a deluge survival kit. J. Abello, P. Pardalos, G. Mauricio, eds. *Handbook of Massive Data Sets.* Kluwer Academic Publishers, Boston, MA, 151–194.

Cambouropoulos, E. 1998. Towards a general computational theory of musical structure. Ph.D. thesis, Faculty of Music, University of Edinburgh, Edinburgh, Scotland.

Conklin, D. 2002. Representation and discovery of vertical patterns in music. C. Anagnostopoulou, M. Ferrand, A. Smaill, eds. *Music and Artificial Intelligence: Lecture Notes in Artificial Intelligence 2445.* Springer-Verlag, Berlin, Germany, 32–42.

Conklin, D., C. Anagnostopoulou. 2001. Representation and discovery of multiple viewpoint patterns. *Proc. Internat. Comput. Music Conf.* Havana, Cuba, 479–485.

Cope, D. 1991. *Computers and Musical Style.* A-R Editions, Madison, WI.

Creighton, H. 1966. *Songs and Ballads from Nova Scotia.* Dover Publications, Inc., New York.

Grilo, C., F. Penousal, A. Cardoso. 2001. Paradigmatic analysis using genetic programming. *Proc. AISB 2001 Sympos. Artificial Intelligence and Creativity in the Arts and Sciences*. York, UK, 51–57.

Gusfield, D. 1997. *Algorithms on Strings, Trees, and Sequences*. Cambridge University Press, Cambridge, UK.

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

Hsu, J.-L., C.-C. Liu, A. Chen. 2001. Discovering nontrivial repeating patterns in music data. *IEEE Trans. Multimedia* **3** 311–325.

Huron, D. 1996. The melodic arch in Western folksongs. *Comput. Musicology* **10** 3–23.

Huron, D. 1997. Humdrum and kern: Selective feature encoding. E. Selfridge-Field, ed. *Beyond MIDI: The Handbook of Musical Codes*, Chap. 26. The MIT Press, Cambridge, MA, 375–401.

Jurafsky, D., J. Martin. 2000. *Speech and Language Processing*. Prentice-Hall, Englewood Cliffs, NJ.

Lartillot, O. 2004. An adaptive and multi-parametric approach for motivic pattern discovery. *Proc. Sound and Music Comput. Conf.* Paris, France, 117–124.

Lewin, D. 1987. *Generalized Musical Intervals and Transformations*. Yale University Press, New Haven, CT.

Nattiez, J.-J. 1975. *Fondements d'une Sémiologie de la Musique*. Union Générale d'Editions, Paris, France.

Rolland, P.-Y., J.-G. Ganascia. 2000. Musical pattern extraction and similarity assessment. E. Miranda, ed. *Readings in Music and Artificial Intelligence*, Chap. 7. Harwood Academic Publishers, Amsterdam, The Netherlands, 115–144.

Sawada, T., K. Satoh. 2000. Composer classification based on patterns of short note sequences. *Proc. AAAI-2000 Workshop on AI and Music*, Austin, Texas, 24–27.

Schaffrath, H. 1995. The Essen folksong collection. D. Huron, ed. *Database Containing 6,255 Folksong Transcriptions in the Kern Format and a 34-Page Research Guide*. CCARH, Menlo Park, CA.