

Lecture Notes in Artificial Intelligence 2445

Subseries of Lecture Notes in Computer Science

Edited by J. G. Carbonell and J. Siekmann

Lecture Notes in Computer Science

Edited by G. Goos, J. Hartmanis, and J. van Leeuwen

Springer

Berlin

Heidelberg

New York

Barcelona

Hong Kong

London

Milan

Paris

Tokyo

Christina Anagnostopoulou Miguel Ferrand
Alan Smaill (Eds.)

Music and Artificial Intelligence

Second International Conference, ICMAI 2002
Edinburgh, Scotland, UK, September 12-14, 2002
Proceedings



Springer

Series Editors

Jaime G. Carbonell, Carnegie Mellon University, Pittsburgh, PA, USA
Jörg Siekmann, University of Saarland, Saarbrücken, Germany

Volume Editors

Christina Anagnostopoulou
Miguel Ferrand
University of Edinburgh, Faculty of Music
12 Nicolson Square, Edinburgh EH8 9DF, UK
E-mail: chrisa@dai.ed.ac.uk, mferrand@music.ed.ac.uk

Alan Smaill
University of Edinburgh, Division of Informatics
80 South Bridge, Edinburgh EH1 1HN, UK
E-mail: a.smaill@ed.ac.uk

Cataloging-in-Publication Data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Music and artificial intelligence : second international conference ;
proceedings / ICMAI 2002, Edinburgh, Scotland, UK, September 12 - 14, 2002.
Christina Anagnostopoulou ... (ed.). - Berlin ; Heidelberg ; New York ;
Barcelona ; Hong Kong ; London ; Milan ; Paris ; Tokyo : Springer, 2002
(Lecture notes in computer science ; Vol. 2445 : Lecture notes in
artificial intelligence)
ISBN 3-540-44145-X

CR Subject Classification (1998): I.2, J.5, I.5

ISSN 0302-9743

ISBN 3-540-44145-X Springer-Verlag Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer-Verlag. Violations are liable for prosecution under the German Copyright Law.

Springer-Verlag Berlin Heidelberg New York,
a member of BertelsmannSpringer Science+Business Media GmbH

<http://www.springer.de>

© Springer-Verlag Berlin Heidelberg 2002
Printed in Germany

Typesetting: Camera-ready by author, data conversion by PTP-Berlin, Stefan Sossna e.K.
Printed on acid-free paper SPIN: 10871055 06/3142 5 4 3 2 1 0

Preface

This volume contains the full research papers of the *2nd International Conference on Music and Artificial Intelligence* (ICMAI02), held in St. Cecilia's Hall, Edinburgh, UK, 12–14 September 2002. The conference was jointly organized by the Faculty of Music and the Division of Informatics, University of Edinburgh.

Each of the papers in this volume was refereed by at least three reviewers from the Program Committee. Additional papers were accepted as Work-in-Progress reports, published separately by the Division of Informatics of the University of Edinburgh. The conference program also included round-table discussions, electronic music concerts, early keyboard instrument demonstrations, and various social events.

The focus of the conference was on the interplay between musical theory and practice on the one hand and techniques and methods from Artificial Intelligence, including computational cognitive modeling of musical processes, on the other. We are especially interested in issues of musical representation and analysis, associated algorithms, and the evaluation of musical artefacts.

Music presents many challenges for AI and Informatics in general. While the analogies with natural language and with vision are productive, music requires novel solutions to its own representational and algorithmic problems. For the musician, work in this area contributes towards musicological study, composition, performance, interactive systems, and sound synthesis.

Three distinguished scholars in the area, Mira Balaban, Jean-Claude Risset, and Antonio Camurri, agreed to give the invited keynotes. These talks covered different features of musical experience and its relation to Artificial Intelligence, namely musical concepts, composition, and performance. More specifically, Jean-Claude Risset's topic was *Musical Composition and Artificial Intelligence: Some Precedents and Prospects*, Mira Balaban's was *Structure and Interpretation of Music Concepts – Music from a Computational Perspective*, and Antonio Camurri's was *Computational Models of Expressive Gesture*.

The papers in the conference dealt with a wide range of aspects of current research, including structural, harmonic, and reductional music analysis, pattern representation and discovery in both monophonic and polyphonic music, music perception of melody and rhythm, the relation between music and natural language, similarity and categorization, music and intonation, musical expression and performance, sound processing, sound classifications, commercial applications, and music on the web.

Acknowledgements. The editors are grateful for all the help of those who made the publication of this volume possible, and especially Darrell Conklin,

VI Organization

Peter Nelson, Emiliós Cambouropoulos, Alison Pease, and Mark Steedman; also, Jordan Fleming, Ewen Maclean, and Robert Dow; and the Faculty of Music and the Division of Informatics, University of Edinburgh for financial support.

July 2002

Christina Anagnostopoulou
Miguel Ferrand
Alan Smaill

Organizing Committee

Alan Smaill (Chair)
Christina Anagnostopoulou (Co-chair)
Miguel Ferrand
Peter Nelson
Alison Pease
Mark Steedman

Program Committee

Jens Arnspar (Ålborg)	Gillian Hayes (Edinburgh)
Mira Balaban (Ben-Gurion)	Henkjan Honing (Nijmegen)
Marta Olivetti Belardinelli (Rome)	Jukka Louhivuori (Jyväskylä)
Rens Bod (Amsterdam)	Tod Machover (MIT-Media Lab)
Carola Boehm (Glasgow)	Guerino Mazzola (Zurich and Vienna)
Amilcar Cardoso (Coimbra)	Stephen McAdams (IRCAM)
Emiliós Cambouropoulos (Thessaloniki)	Eduardo Miranda (SONY CSL-Paris)
Elaine Chew (USC)	Peter Nelson (Edinburgh)
Darrell Conklin (ZymoGenetics)	François Pachet (SONY CSL-Paris)
Ian Cross (Cambridge)	Stephen Travis Pope (Santa Barbara)
Roger Dannenberg (CMU)	Alan Smaill (Edinburgh)
Peter Desain (Nijmegen)	Mark Steedman (Edinburgh)
Anastasia Georgaki (Athens)	David Temperley (Rochester)

Invited Speakers

Mira Balaban (Ben-Gurion University, Israel)
Antonio Camurri (University of Genoa, Italy)
Jean-Claude Risset (Laboratoire de Mécanique et d'Acoustique, France)

Table of Contents

Invited Talks

Structure and Interpretation of Music Concepts: Music from a Computational Perspective	1
<i>Mira Balaban</i>	
Expressive Gesture	4
<i>Antonio Camurri</i>	

Regular Contributions

A General Parsing Model for Music and Language	5
<i>Rens Bod</i>	
The Spiral Array: An Algorithm for Determining Key Boundaries	18
<i>Elaine Chew</i>	
Representation and Discovery of Vertical Patterns in Music	32
<i>Darrell Conklin</i>	
Discovering Musical Structure in Audio Recordings	43
<i>Roger B. Dannenberg, Ning Hu</i>	
Real Time Tracking and Visualisation of Musical Expression	58
<i>Simon Dixon, Werner Goebel, Gerhard Widmer</i>	
Automatic Classification of Drum Sounds: A Comparison of Feature Selection Methods and Classification Techniques	69
<i>Perfecto Herrera, Alexandre Yeterian, Fabien Gouyon</i>	
Some Formal Problems with Schenkerian Representations of Tonal Structure	81
<i>Tim Horton</i>	
Respiration Reflecting Musical Expression: Analysis of Respiration during Musical Performance by Inductive Logic Programming	94
<i>Soh Igarashi, Tomonobu Ozaki, Koichi Furukawa</i>	
Mimetic Development of Intonation	107
<i>Eduardo Reck Miranda</i>	
Interacting with a Musical Learning System: The Continuator	119
<i>François Pachet</i>	

VIII Table of Contents

Recognition of Isolated Musical Patterns Using Hidden Markov Models . . .	133
<i>Aggelos Pikrakis, Sergios Theodoridis, Dimitris Kamarotos</i>	
A Model for the Perception of Tonal Melodies	144
<i>Dirk-Jan Povel</i>	
Control Language for Harmonisation Process	155
<i>Somnuk Phon-Amnuaisuk</i>	
Evaluating Melodic Segmentation	168
<i>Christian Spevak, Belinda Thom, Karin Höthker</i>	
Combining Grammar-Based and Memory-Based Models of Perception of Time Signature and Phase	183
<i>Neta Spiro</i>	
A Bayesian Approach to Key-Finding	195
<i>David Temperley</i>	
Author Index	207

Structure and Interpretation of Music Concepts: Music from a Computational Perspective

Mira Balaban

Department of Information Systems Engineering
Ben-Gurion University of the Negev, Israel. mira@cs.bgu.ac.il

Research in the field of Computer-Music is directed towards construction of computer systems for composition, performance of musical tasks, music training, signal processing and extension of traditional music sounds, notation study, music analysis, storage and communication of musical data, and music information retrieval and classification. Computer applications in music lead to the development of computational models. Examples are the works of Winograd, Ebcioglu, Laske, Cope, Bamberger, Berger & Gang and Orlarey. Berger & Gang use artificial neural networks to investigate music perception, Orlarey et al. introduce the notion of Concrete Abstraction (borrowed from Lambda Calculus) as a basis for their purely functional composition environment, while Winograd uses a Systemic Grammar approach borrowed from Linguistics.

In spite of rich activity in musicology and in computer music, there is little research (if at all) on the development of a computational basis for computer music (e.g., Schenker, Narmour, Lerdhal and Jackendoff). Most computer music systems indeed have huge musical knowledge and intuitions embedded in them, but cannot point on underlying explicit computational basis. Clearly they assume some level of *primitives*, and use different *abstraction levels* and kinds, but these are not anchored in computational models that account for their basics, organization, usage and semantics. In a sense, this situation reminds one of the early days of Natural Language Processing, when a large amount of linguistic knowledge was embedded in huge semantic networks.

In this talk we propose to initiate a study of computational frameworks for music knowledge. The idea is to make the music knowledge that underlies computer-music applications (either intentionally or not) explicit. As a starting point, we suggest to study *music schemas*. Here is how Musicologist Dahlia Cohen (dept. of Musicology, Hebrew University, Israel) describes the role that music schemas play in music understanding:

Listening to music (perception and reaction) and its memorization, are cognitive activities that depend on *schemas* that are created, unconsciously, at an early age. The schemas represent relationship types between musical events that are dictated by principles of organization, in various abstraction levels. The schemas trigger expectations that may or may not be realized. Music is distinguished from other arts in having the systematic organization as the sole means for triggering various types of reactions (whereas other arts employ non-artistic, non-organizational means for reaction triggering).

Music schemas are classified into *learned* and *natural* ones. The learned, like scales, chords and rhythmic patterns, are man-made, culture dependent, and have hardly any parallel outside the music domain (but are not arbitrary). The natural schemas can be recognized in various human activities and in natural phenomena, outside the music domain - like asymptotic behavior of the various parameters, operation kinds, etc. The study of natural schemas is relatively new.

Music knowledge is based on the combination of the two kinds of schemas. We propose a *constructive-computational* study into the principles of the learned and natural schemas. By constructive we mean an implemented computational study. There is a need for an investigation of different levels of primitives and parameters, and different abstraction manners. *Procedural abstraction* should be investigated in the light of natural *Operation Schemas*, while *Data Abstraction* should be studied in the light of the theory of *Abstract Data Types* and other specification approaches. Once a computational basis is laid for music schemas, there is ground for studying various computational models as developed in natural language processing, artificial intelligence, information retrieval and programming. Such models include grammar models such as *Functional Unification Grammars*, *Feature* and *Attribute logics* for language, *Description logics*, *temporal ontologies* such as *time intervals* and *point algebras*, *temporal constraints* approaches, etc. Our expectation is that the investigation of computational models from a musical perspective will lead to the emergence of computational music frameworks.

In order to accomplish this plan, we adopt the computational modeling approach that Abelson and Sussman introduce in their milestone book [1], *Structure and Interpretation of Computer Programs* (SICP). During the last 4 years we were involved in teaching a course along these lines in the Musicology Department in Bar-Ilan University in Israel, and in the Computer Science Department in Carnegie-Mellon University in Pittsburgh. Next year, we continue this initiative in a course given in the Computer Science Department in the Hebrew University in Israel. In this course we study and implement the principles of the learned and natural schemas, focussing on the Western Music, but within a wide perspective of music in different cultures. The computational tools emphasize the *process-concept* distinction in music, and its analogy to the *procedure-data* distinction in computation. The representation and computation models that are considered include grammatical models, constraint approaches, and delayed evaluation for real time processing. The overall approach is in the flavor of Abelson and Sussman's SICP book, and the computational tools are built in the Scheme language.

In the talk we will describe our approach, results and experience in teaching this course. The teaching and experimentation with a group of interested students seems to play an essential role in the understanding of the concepts, methods, necessities, etc. In particular, it is clear that the impact of the teaching framework is twofold:

1. The constructive approach has an unexpected influence on the musical understanding and thinking on the students' part.
2. The discussion and feedback within a group of students with analytical thinking and some musical background is essential for the development of the computational foundations.

This notion was further extended by Balaban, Barzilay and Elhadad, and implemented in the powerful editing prototype BOOMS.

This is joint work with Yosi Saporta, Musicology department, Bar-Ilan University, Ramat-Gan, Israel, and Dahlia Cohen, Musicology Department, Hebrew University, Jerusalem, Israel.

References

1. H. Abelson and G. J. Sussman. *Structure and interpretation of computer programs*. MIT Press, Cambridge, Mass., second edition, 1996.

Expressive Gesture

Antonio Camurri

DIST – University of Genova, Italy music@dist.unige.it

Abstract. Gesture is the carrier of a set of temporal/spatial features responsible of conveying expressiveness. The concept of expressive gesture includes music, human movement, visual (e.g., computer animated) gesture components. Modelling and communication of expressive and emotional content in non-verbal interaction by multi-sensory interfaces is receiving a raising interest from research and industry communities. Music performance and full-body movements (e.g dance) are first class conveyors of expressiveness through gesture. Automatic analysis and synthesis of expressive gesture can open novel scenarios in the field of interactive multimedia, especially in artistic contexts, and represent at the same time an opportunity and a challenge for interactive music systems designers. The seminar will include aspects related to experiments, modelling, control and application of expressiveness in interactive music and new media performance. Sample audiovisual and real-time demonstrations will be presented to clarify main concepts. This research work is partially funded by the EU IST projects MEGA (Multisensory Expressive Gesture Applications, www.megaproject.org) and CARE HERA (Creating Aesthetical Resonant Environments for Therapy and Rehabilitation).

References

1. A. Camurri and A. Coglio. An architecture for emotional agents. *IEEE Multimedia*, 5(4):24–33, 1998.
2. A. Camurri and P. Ferrentino. Interactive environments for music and multimedia. *ACM Multimedia Systems*, 7:32–47, 1999. Special issue on Audio and Multimedia.
3. A. Camurri, S. Hashimoto, M. Ricchetti, R. Trocca, K. Suzuki, and G. Volpe. EyesWeb — toward gesture and affect recognition in dance/music interactive systems. *Computer Music Journal*, 24:57–69, 2000.
4. A. Camurri, G. De Poli, M. Leman, and G. Volpe. A multi-layered conceptual framework for expressive gesture applications. In *Proc. Intl EU-TMR MOSART Workshop, Univ Pompeu Fabra, Barcelona*, 2001.
5. A. Camurri and R. Trocca. Movement and gesture in intelligent interactive music systems. In M. Battier and M. Wanderley, editors, *Trends in Gestural Control of Music*. IRCAM, Paris, France, 2000.
6. A. Camurri, R. Trocca, and G. Volpe. Interactive systems design: A KANSEI-based approach. In *Proc. Intl. Conf. NIME-2002*, Dublin, 2002. MediaLab Europe.

A General Parsing Model for Music and Language

Rens Bod

Institute for Logic, Language and Computation
University of Amsterdam, &
School of Computing, University of Leeds
`rens@illc.uva.nl`

Abstract. Is there a general mechanism that governs the perception of phrase structure in music and language? While it is usually assumed that humans have separate faculties for music and language, this work focuses on the commonalities rather than on the differences between these modalities, aiming at finding a deeper "faculty". We present a series of data-oriented parsing (DOP) models which aim at balancing the simplest structure with the most likely structure of an input. Experiments with the Essen Folksong Collection and the Penn Treebank show that exactly the same model with the same parameter setting achieves maximum parse accuracy for both music and language. This suggests an interesting parallel between musical and linguistic processing. We show that our results outperform both the melodic component of Temperley (2001) and the musical parser of Bod (2001b).

1 Introduction

It is widely accepted that the human cognitive system tends to organize perceptual information into complex, hierarchical descriptions that can be represented by tree structures. Tree structures have been used to describe linguistic perception (e.g. Wundt 1901; Chomsky 1965), musical perception (e.g. Longuet-Higgins 1976; Lerdahl & Jackendoff 1983) and visual perception (e.g. Palmer 1977; Marr 1982). Yet, there seems to be little or no work that emphasizes the commonalities between these different forms of perception and that searches for a general, underlying mechanism which governs all perceptual organization. This paper studies exactly that question: acknowledging the differences between the perceptual modalities, is there a general model which can predict the perceived tree structure for sensory input? While we will argue for a general model of structural organization, we will carry out experiments only with music (using the Essen Folksong Collection -- Schaffrath 1995) and language (using the Penn Treebank -- Marcus et al. 1993), since no benchmark of visual tree structures is currently available, to the best of our knowledge.

Figure 1 gives three simple examples of linguistic, musical and visual information with their corresponding tree structures printed below.

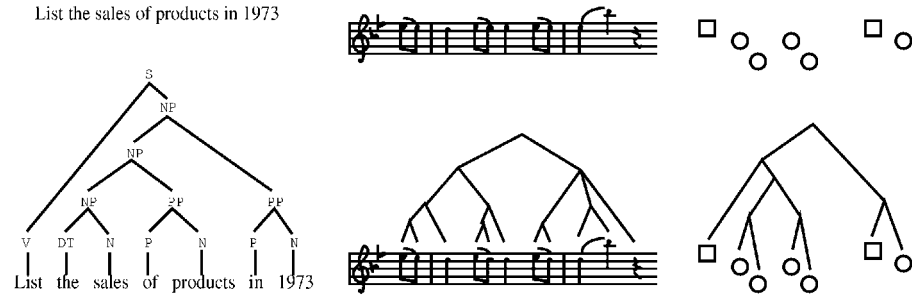


Fig. 1. Examples of linguistic, musical and visual input with their tree structures.

Note that the linguistic tree structure is labeled with syntactic categories, whereas the musical and visual tree structures are unlabeled. This is because in language there are syntactic constraints on how words can be combined into larger constituents, while in music (and to a lesser extent in vision) there are no such restrictions: in principle any note may be combined with any other note.

Apart from these differences, there is also a fundamental commonality: the perceptual input undergoes a process of hierarchical structuring which is not found in the input itself. The main problem is thus: how can we derive the perceived tree structure for a given input? That this problem is not trivial may be illustrated by the fact that the inputs above can also be assigned the following, alternative structures:

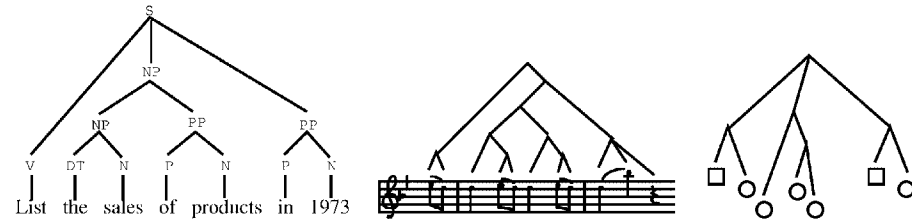


Fig. 2. Alternative tree structures for the inputs in figure 1.

These alternative structures are possible in that they *can* be perceived. The linguistic tree structure in figure 1 corresponds to a meaning which is different from the tree in figure 2. The two musical tree structures correspond to different melodic groupings. And the two visual structures correspond to different visual Gestalts. But while the alternative structures are all possible, they are not plausible: they do not correspond to the structures that are actually perceived by the perceptual system.

The phenomenon that the same input may be assigned different structural organizations is known as the *ambiguity problem*. This problem is one of the hardest problems in cognitive science. Even in language, where a phrase-structure grammar specifies which words can be combined into constituents, the ambiguity problem is notoriously hard. Charniak (1997: 37) shows that many sentences from the Wall Street Journal have more than one million different parse trees. Humans will fail to

notice this dazzling degree of ambiguity; not more than a few analyses will spontaneously come to mind. The ambiguity problem for musical and visual input is even harder, since there are no constraints on how the perceptual primitives may be combined into constituents. Talking about rhythm perception in music, Longuet-Higgins and Lee (1987) note that "Any given sequence of note values is in principle infinitely ambiguous, but this ambiguity is seldom apparent to the listener."

2 Two Principles: Likelihood and Simplicity

How can we predict from the set of all possible tree structures the tree that is actually perceived by the human cognitive system? In the field of visual perception, two competing principles have traditionally been proposed to govern perceptual organization. The first, initiated by Helmholtz (1910), advocates the *likelihood principle*: perceptual input will be organized into the most probable organization. The second, initiated by Wertheimer (1923) and developed by other Gestalt psychologists, advocates the *simplicity principle*: the perceptual system is viewed as finding the simplest perceptual organization consistent with the input (see Chater 1999 for an overview). These two principles have also been used in musical and linguistic perception. In the following, we briefly discuss these principles for each modality, after which we will argue for a new integration of the two principles.

The likelihood principle has been particularly influential in the field of natural language processing (see Manning and Schütze 1999 for a review). In this field, the most appropriate tree structure of a sentence is assumed to be its most likely structure. The likelihood of a tree is usually computed from the probabilities of its parts (e.g. phrase-structure rules) taken from a large manually analyzed language corpus (i.e. a *treebank*). State-of-the-art probabilistic parsers such as Collins (2000), Charniak (2000) and Bod (2001a) obtain around 90% precision and recall on the Wall Street Journal (WSJ -- see Marcus et al. 1993). The likelihood principle has also been applied to musical perception, e.g. in Raphael (1999) and Bod (2001b/c). As in probabilistic natural language processing, the most probable musical tree structure can be computed from the probabilities of rules or fragments taken from a large annotated musical corpus. A musical benchmark which has been used to test various models is the Essen Folksong Collection (Schaffrath 1995). Also in vision science, there is a huge interest in probabilistic models (e.g. Kersten 1999). Mumford (1999) has even seen fit to declare the Dawning of Stochasticity. Unfortunately, no visual treebanks are currently available.

The simplicity principle has a long tradition in the field of visual perception psychology (e.g. Leeuwenberg 1971; Simon 1972; Buffart et al. 1983). In this field, a visual pattern is formalized as a constituent structure by means of a "visual coding language" based on primitive elements such as line segments and angles. Perception is described as the process of selecting the simplest structure corresponding to the "shortest encoding" of a visual pattern. The notion of simplicity has also been applied to music perception. Collard et al. (1981) use the coding language of Leeuwenberg (1971) to predict the metrical structure for four preludes from Bach's *Well-Tempered Clavier*. More well-known in music perception is the theory proposed by Lerdahl and Jackendoff (1983). Their theory contains two kinds of rules: "well-formedness rules" and "preference rules". The role of well-formedness rules is to define the kinds of formal objects (grouping structures) the theory employs. What grouping structures a

listener actually hears, is then described by the preference rules (see also Temperley 2001), which describe Gestalt-preferences of the kind identified by Wertheimer (1923), and which can therefore also be seen as an embodiment of the simplicity principle. Notions of simplicity also exist in language processing. For example, Frazier (1978) can be viewed as arguing that the parser prefers the simplest structure containing minimal attachments. Bod (2000) defines the simplest tree structure of a sentence as the structure generated by the smallest number of subtrees from a given treebank.

3 Combining the Two Principles

The key idea of the current paper is that both principles play a role in perceptual organization: the simplicity principle as a general cognitive preference for economy, and the likelihood principle as a probabilistic bias due to previous perceptual experiences. To combine these principles into one model that selects the perceived tree structure, we need to start out with a model that characterizes the set of *pos-sible* structures of an input. In the current work, we have chosen for a model which has been quite successful in natural language processing and computational music-ology, and which is known as Data-Oriented Parsing or DOP (cf. Bod 1998). DOP learns a grammar by extracting subtrees from a given treebank and combines these subtrees to analyze new input. We have chosen for DOP because (1) it uses subtrees of arbitrary size, capturing non-local dependencies, and (2) it has obtained competitive results on various benchmarks (Bod 2001a/b, Collins & Duffy 2002).

We will illustrate DOP with a simple example. Suppose we are given the following linguistic treebank of only two trees for resp. *She wanted the dress on the rack* and *she saw the dog with the telescope* (we will come back to musical treebanks in the next section):

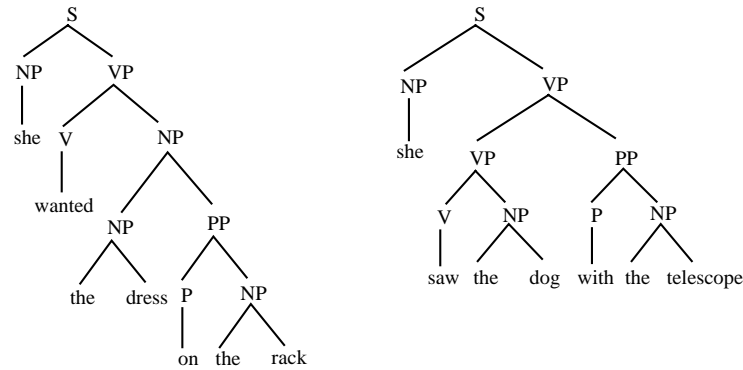


Fig. 3. An example treebank

The DOP model parses a new sentence, e.g. *She saw the dress with the telescope*, by combining subtrees from this treebank by means of a simple *substitution operation* (indicated as \circ), as in figure 4. Thus the substitution operation combines two subtrees by substituting the second subtree on the leftmost nonlexical leaf node of the first

subtree (the result of which may be combined with a third subtree, etc.). A combination of subtrees that results in a tree structure for the whole sentence is called a *derivation*. Since there are many different subtrees, of various sizes, there are typically also many different derivations that produce, however, the *same* tree, as in figure 5.

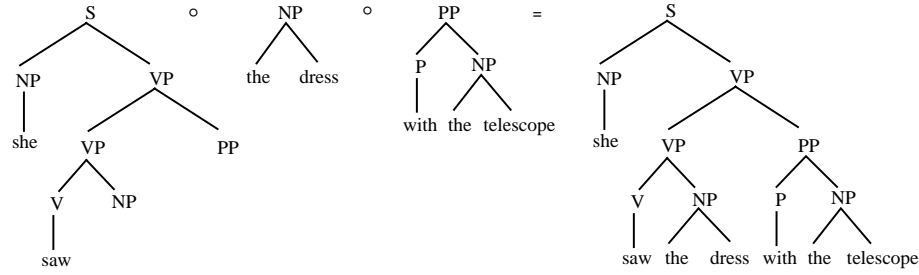


Fig. 4. Parsing a sentence by combining subtrees from figure 3.

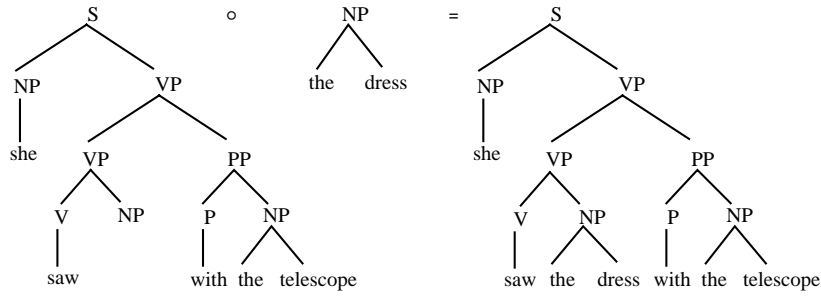


Fig. 5. A different derivation which produces the same parse tree.

The more interesting case occurs when there are different derivations that produce *different* trees. This happens when a sentence is ambiguous; for example, DOP also produces the following alternative parse tree for *She saw the dress with the telescope*: In Bod (1993), the likelihood principle is used to predict the perceived tree structure. This model, which we will refer to as *Likelihood-DOP*, compositionally computes the most probable tree of an input from the occurrence-frequencies of the subtrees. The probability of a subtree t is computed as the number of occurrences of t , divided by the total number of occurrences of treebank-subtrees that have the same root label as t (see Bod et al. 2002b, for some other probability models for subtrees). The probability of a derivation is computed as the product of the probabilities of the subtrees involved in it. Finally, the probability of a parse tree is equal to the probability that any of its derivations occurs, which is the sum of the probabilities of all distinct derivations that produce that tree. In Bod (2001a) and Goodman (2002), efficient algorithms are given that compute for an input string the most probable parse tree.

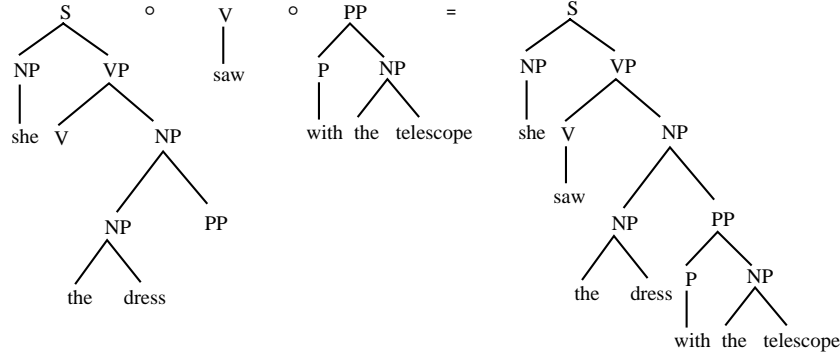


Fig. 6. A different derivation which produces a different parse tree.

Likelihood-DOP does not do justice to the preference humans display for the simplest structure. In Bod (2000), the simplest tree structure of an input is defined as the tree that can be constructed by the smallest number subtrees from the treebank. We will refer to this model as *Simplicity-DOP*. Instead of producing the most probable parse tree for an input, *Simplicity-DOP* thus produces the parse tree generated by the shortest derivation consisting of the fewest treebank-subtrees, *independent* of the probabilities of these subtrees. For example, given the treebank in Figure 3, the simplest parse tree for *She saw the dress with the telescope* is given in Figure 5, since that parse tree can be generated by a derivation of only two treebank-subtrees, while the parse tree in Figure 6 (and any other parse tree) needs at least three treebank-subtrees to be generated. In case the shortest derivation is not unique, the model backs off to a frequency ordering of the subtrees. That is, all subtrees of each root label are assigned a rank according to their frequency in the treebank: the most frequent subtree (or subtrees) of each root label gets rank 1, the second most frequent subtree gets rank 2, etc. Next, the rank of each shortest derivation is computed as the sum of the ranks of the subtrees involved. The shortest derivation with the smallest sum, or highest rank, is taken as the derivation producing the simplest tree structure. In Bod (2000) it is shown how this simplest structure can be efficiently computed by means of a standard best-first parsing algorithm. While *Simplicity-DOP* obtains very similar parse accuracy as *Likelihood-DOP* on the WSJ (Bod 2000), the set of correctly predicted parse trees of *Simplicity-DOP* does not match with the set of correctly predicted parse trees of *Likelihood-DOP*. This suggests that a combined model, which does justice to both simplicity and likelihood, may boost the accuracy considerably. We will refer to such a model as *Combined-DOP*.

The underlying idea of *Combined-DOP* is that the human perceptual system searches for the simplest tree structure but in doing so it is biased by the likelihood of the tree structure. That is, instead of selecting the simplest tree *per se*, *Combined-DOP* selects the simplest tree from among the n most probable trees, where n is our free parameter. There are of course other ways to combine simplicity and likelihood within the DOP framework. A straightforward alternative would be to select the most probable tree from among the n simplest trees, suggesting that the perceptual system is searching for the most probable structure only from among the simplest ones. We will refer to the first combination of simplicity and likelihood (which selects the simplest among the n most probable trees) as *Combined-DOP1*, and to the second

combination (which selects the most probable among the n simplest trees) as Combined-DOP2. Note that for $n=1$, Combined-DOP1 is equal to Likelihood-DOP, since there is only one most probable tree to select from, and Combined-DOP2 is equal to Simplicity-DOP, since there is only one simplest tree to select from. Moreover, if n gets large, Combined-DOP1 converges to Simplicity-DOP while Combined-DOP2 converges to Likelihood-DOP. By varying the parameter n , we will be able to compare Likelihood-DOP, Simplicity-DOP and several instantiations of Combined-DOP1&2.

4 The Test Domains

As our linguistic test domain we will use the Wall Street Journal (WSJ) portion in the Penn Treebank (Marcus et al. 1993). This portion contains approx. 50,000 sentences that have been manually enriched with the perceived linguistic tree structures. The WSJ has been extensively used to test and compare natural language parsers (cf. Manning & Schütze 1999).

As our musical test domain we will use the European folksongs in the Essen Folksong Collection (Schaffrath 1995), which correspond to approx. 6,200 folksongs that have been manually enriched with their perceived musical grouping structures. The Essen Folksong Collection has been previously used by Bod (2001b) and Temperley (2001) to test their melodic parsers. The Essen folksongs are encoded by the Essen Associative Code (ESAC). The pitch encodings in ESAC resemble "solfege": scale degree numbers are used to replace the movable syllables "do", "re", "mi", etc. Thus 1 corresponds to "do", 2 corresponds to "re", etc. Chromatic alterations are represented by adding either a "#" or a "b" after the number. The plus ("+") and minus ("-") signs are added before the number if a note falls resp. above or below the principle octave (thus -1, 1 and +1 refer al to "do", but on different octaves). Duration is represented by adding a period or an underscore after the number. A period (".") increases duration by 50% and an underscore ("_") increases duration by 100%; more than one underscore may be added after each number. If a number has no duration indicator, its duration corresponds to the smallest value. A pause is represented by 0, possibly followed by duration indicators. ESAC uses hard returns to indicate a phrase boundary. We automatically converted ESAC's phrase boundary indications into bracket representations, where "(" indicates the start of a phrase and ")" the end of a phrase. For example, The following figure gives an an encoding of an actual folksong from the Essen Folksong Collection ("Schlaf Kindlein feste") converted to our bracket representation:

(3_221_-5)(-533221_-5)(13335432)(13335432_)(3_221_-5_)

Fig. 7. Bracket representation for folksong K0029, "Schlaf Kindlein feste".

To use these structures by our DOP models, we automatically added three basic labels to the annotations: the label "S" to each whole folksong, the label "P" to each phrase, and the label "N" to each note. Thus the annotation in figure 7 becomes (we only represent the first 5 notes):

$$S(P(N(3_) N(2) N(2) N(1_) N(-5)) P(...) P(...) \dots)$$

Fig. 8. Labeled bracket representation for (first five notes of) the structure in figure 7.

The use of the label "N" distinguishes our work from the annotations used in Bod (2001b/c) where only labels for song and phrases were used ("S" and "P"). The addition of "N" for notes enhances the productivity and robustness of the model, although it also leads to a much larger number of subtrees.

5 Experimental Evaluation and Comparison

To evaluate our DOP models, we used the blind testing paradigm which dictates that a treebank be randomly divided into a training set and a test set, where the strings from the test set are parsed by means of the subtrees from the training set. We applied the PARSEVAL metrics of precision and recall to compare a proposed parse tree P with the corresponding correct test set parse tree T (Black et al. 1991):

$$\text{Precision} = \frac{\# \text{ correct constituents in } P}{\# \text{ constituents in } P} \quad \text{Recall} = \frac{\# \text{ correct constituents in } P}{\# \text{ constituents in } T}$$

A constituent in P is "correct" if there exists a constituent in T of the same label that spans the same elements (i.e. words or notes). To balance precision and recall into a single measure, we will employ the widely-used F-score: $F\text{-score} = 2 * \text{Precision} * \text{Recall} / (\text{Precision} + \text{Recall})$.

For our experiments, we divided both treebanks into 10 random training/test set splits; 10% of the WSJ was used as test material each time, while for the Essen Folksong Collection test sets of 1,000 folksongs were used (otherwise the musical test sets would become rather small). For words in the test material that were unknown in the training set, we estimated their categories as in Bod (2001a); there were no unknown notes. We also used the same parsing algorithm as in Bod (2001a) and restricted the maximum depth of the subtrees to 14, using random samples of 400,000 subtrees for each depth > 1 . Table 1 shows the mean F-scores obtained by both versions of Combined-DOP for language and music and for various values of n (recall that for $n=1$, Combined-DOP1 is equal to Likelihood-DOP while Combined-DOP2 is equal to Simplicity-DOP).

The table shows that there is an increase in accuracy for both versions of Combined-DOP if the value of n increases from 1 to 11. But while the accuracy of Combined-DOP1 decreases after $n=13$ and converges to Simplicity-DOP (i.e. Combined-DOP2 at $n=1$), the accuracy of Combined-DOP2 continues to increase and converges to Likelihood-DOP (i.e. Combined-DOP1 at $n=1$). The highest accuracy is obtained by Combined-DOP1 at $11 \leq n \leq 13$, for both language and music. Thus Combined-DOP1 outperforms both Likelihood-DOP and Simplicity-DOP, and the selection of the simplest structure out of the top most probable ones turns out to be a more promising model than the selection of the most probable structure out of the top simplest ones. According to paired t -testing, the accuracy improvement of Combined-DOP1 at $n=11$ over Combined-DOP1 at $n=1$ (when it is equal Likelihood-DOP) is statistically significant for both language ($p < .0001$) and music ($p < .006$).

Table 1. F-scores obtained by the two versions of Combined-DOP for language and music.

n	Combined-DOP1 (Simplest among n most probable)		Combined-DOP2 (Most probable among n simplest)	
	Language	Music	Language	Music
1	87.9%	86.0%	85.6%	84.3%
5	89.3%	86.8%	86.1%	85.5%
10	90.2%	87.2%	87.0%	85.7%
11	90.2%	87.3%	87.0%	85.7%
12	90.2%	87.3%	87.0%	85.7%
13	90.2%	87.3%	87.0%	85.7%
14	90.2%	87.2%	87.0%	85.7%
15	90.2%	87.2%	87.0%	85.7%
20	90.0%	86.9%	87.1%	85.7%
50	88.7%	85.6%	87.4%	86.0%
100	86.8%	84.3%	87.9%	86.0%
1,000	85.6%	84.3%	87.9%	86.0%

It is surprising that Combined-DOP1 reaches highest accuracy at such a small value for n . But it is even more surprising that the same model (with the same parameter setting) obtains maximum accuracy for both language and music. This model embodies the idea that the human perceptual system strives for the simplest structure but in doing so it only searches among a few most probable structures.

It is interesting to compare our musical results with Bod (2001b/c), who tested three probabilistic parsing models of increasing complexity on the same training/test set splits from the Essen Folksong Collection. The best results were obtained by a hybrid DOP-Markov parser: 80.7% F-score. This is significantly worse than our best result of 87.3% obtained by Combined-DOP1. This difference may be explained by the fact that the hybrid DOP-Markov parser in Bod (2001b/c) only takes into account context from higher nodes in the tree and not from any sister nodes, while the DOP models presented in the current paper take any subtree into account of arbitrary width (and depth), thereby covering a larger amount of musical context. Moreover, as mentioned in section 4, the models in Bod (2001b/c) do not use a label "N" for notes (which would result in less productivity if these models were not extended with the Markov grammar approach). In Bod (2002) a different combination of likelihood and simplicity was investigated which searches for the shortest derivation minimizing the "weights" of the subtrees. Although that combination also outperformed Simplicity-DOP and Likelihood-DOP, it obtained an F-score of 86.9% which is lower than our result of 87.3% ($p < .05$).

It would also be interesting to compare our musical results to the melodic parser of Temperley (2001), who uses a system of preference rules similar to Lerdahl and Jackendoff (1983), and which is also evaluated on the Essen folksongs. But while we have tested on several test sets of 1,000 randomly selected folksongs, Temperley used only one test set of 65 folksongs that was moreover cleaned up by eliminating folksongs with irregular meter (Temperley 2001: 74). A comparison between our and Temperley's parser is thus virtually meaningless; yet, it is noteworthy that Temperley's parser correctly identified 75.5% of the phrase boundaries. Although this is lower than the 87.3% obtained by Combined-DOP1, Temperley's parser is not

"trained" on previously analyzed examples like our model (though we should mention that Temperley's results were obtained by tuning the optimal phrase length of his parser on the average phrase length of the Essen Folksong Collection).

Parsing models trained on treebanks are still rather uncommon in musical analysis, though they are widely used in language parsing. Most musical parsing models, including Temperley's, are based on the Gestalt principles identified by Wertheimer (1923), which tend to assign phrase boundaries on larger intervals. However, in Bod (2001b) we have shown that the Gestalt principles often predict the incorrect phrase boundaries for the Essen folksongs. More than 32% of the Essen folksongs contain a phrase boundary that falls just *before* or *after* a large pitch or time interval (which we have called *jump-phrases*, e.g. in figure 7) rather than *on* such intervals -- as would be predicted by the Gestalt principles. We have also shown that for 98% of these jump-phrases, higher-level phenomena such as melodic parallelism, meter or harmony are not of any help: they simply reinforce the incorrect predictions made by the Gestalt principles (see Bod 2001b/c for details and examples). Our DOP models, on the other hand, can easily deal with such jump-phrases since they take into account any sequence of notes that has been observed with a certain structure rather than trying to capture these by a few rules.

6 Discussion and Conclusion

We have seen that our combination of simplicity and likelihood is quite rewarding for linguistic and musical structuring, suggesting an interesting parallel between the two modalities. Yet, we might raise the question whether a model which massively memorizes and re-uses previously perceived structures has any cognitive plausibility. Although this question is only important if we want to claim cognitive relevance for our model, there actually turns out to be some evidence that people store various kinds of previously heard fragments, both in music (Saffran et al. 2000) and language (Jurafsky 2002). But do people store fragments of *arbitrary* size, as proposed by DOP? In his overview article, Jurafsky (2002) reports on a large body of psycholinguistic evidence showing that people not only store lexical items and bigrams, but also frequent phrases and even whole sentences. For the case of sentences, people not only store idiomatic sentences, but also "regular" high-frequency sentences. Thus, at least for language it seems that humans store fragments of arbitrary size provided that these fragments have a certain minimal frequency. And this suggests that humans need not always parse new input by the rules of a grammar, but that they can productively re-use previously analyzed fragments. Yet, there is no evidence that people store *all* fragments they hear, as suggested by DOP. Only high-frequency fragments seem to be memorized. However, if the human perceptual faculty needs to *learn* which fragments will be stored, it will initially need to keep track of all fragments (with the possibility of forgetting them) otherwise frequencies can never accumulate. This results in a model which continuously and incrementally updates its fragment memory given new input, which is in correspondence with the DOP approach.

There have been other proposals for integrating the principles of simplicity and likelihood in perceptual organization (see Chater 1999 for a review). Chater notes that in the context of Information Theory (Shannon 1948), the principles of simplicity and likelihood are identical. In this context, the simplicity principle is interpreted as

minimizing the expected length to encode a message i , which is $-\log_2 p_i$ bits, and which leads to the same result as maximizing the probability of i . If we used this information-theoretical definition of simplest structure in Simplicity-DOP, it would return the same structure as Likelihood-DOP, and no improved results would be obtained by a combination of the two. On the other hand, by defining the simplest structure as the one generated by the smallest number of subtrees, independent of their probabilities, we created a notion of simplicity which is provably different from the notion of most likely structure, and which, if combined with Likelihood-DOP, obtained improved results.

Another integration of the two principles may be provided by the notion of Minimum Description Length or MDL (cf. Rissanen 1989). The MDL principle can be viewed as preferring the statistical model that allows for the shortest encoding of the training data. The relevant encoding consists of two parts: the first part encodes the model of the data, and the second part encodes the data in terms of the model (in bit length). MDL has been used in natural language processing for estimating the parameters of a stochastic grammar (e.g. Osborne 1999). We will leave it as an open research question as to whether MDL can be successfully used for estimating the parameters of DOP's subtrees. However, since MDL is known to give asymptotically the same results as maximum likelihood estimation (MLE) (Rissanen 1989), its application to DOP may lead to an unproductive model. This is because the maximum likelihood estimator will assign the training set trees their empirical frequencies, and assign 0 weight to all other trees. This would result in a model which can only parse the training data and no other input.

We have seen that DOP presents an entirely supervised approach to the problem of musical structuring. It is very likely that there are grouping properties for which no prior expectations are necessary. DOP does not contribute to the discovery of such properties, but it does neither neglect them, as long as they are implicit in the treebank annotations. For example, in Bod (2001c) we showed that Wertheimer's Gestalt preferences (which are not based on prior experience) are reflected in about 68% of the folksongs in the Essen Folksong Collection, where phrase boundaries fall on large intervals. DOP automatically takes these preferences into account by subtrees that contain such phrase boundaries. But DOP also takes into account phrases where boundaries do *not* fall on large intervals (e.g. the jump-phrases mentioned in section 5). By using all subtrees, it is hoped that DOP takes into account any preference humans may have used in analyzing the data, whether based on prior experience or not.

References

- Black, E. et al. (1991). A Procedure for Quantitatively Comparing the Syntactic Coverage of English, *Proceedings DARPA Speech and Natural Language Workshop*, Morgan Kaufmann.
- Bod, R. (1993). Using an Annotated Language Corpus as a Virtual Stochastic Grammar, *Proceedings AAAI-93*, Menlo Park, Ca.
- Bod, R. (1998). *Beyond Grammar: An Experience-Based Theory of Language*, Stanford: CSLI Publications (Lecture notes number 88), distributed by Cambridge University Press.
- Bod, R. (2000). Parsing with the Shortest Derivation. *Proceedings COLING-2000*, Germany.
- Bod, R. (2001a). What is the Minimal Set of Fragments that Achieves Maximal Parse Accuracy? *Proceedings ACL 2001*, Toulouse, France.

- Bod, R. (2001b). A Memory-Based Model for Music Analysis. *Proceedings International Computer Music Conference (ICMC'2001)*, Havana, Cuba.
- Bod, R. (2001c). Memory-Based Models of Melodic Analysis: Challenging the Gestalt Principles. *Journal of New Music Research*, 31(1), in press. (available at <http://turing.wins.uva.nl/~rens/jnmr01.pdf>)
- Bod, R. (2002). Combining Simplicity and Likelihood in Language and Music. *Proceedings CogSci'2002*. Fairfax, Virginia.
- Bod, R., J. Hay and S. Jannedy (eds.) (2002a). *Probabilistic Linguistics*. Cambridge, The MIT Press. (in press)
- Bod, R., R. Scha and K. Sima'an (eds.) (2002b). *Data-Oriented Parsing*. Stanford, CSLI Publications. (in press)
- Buffart, H., E. Leeuwenberg and F. Restle (1983). Analysis of Ambiguity in Visual Pattern Completion. *Journal of Experimental Psychology* 9, 980-1000.
- Charniak, E. (1997). Statistical Techniques for Natural Language Parsing, *AI Magazine*, Winter 1997, 32-43.
- Charniak, E. (2000). A Maximum-Entropy-Inspired Parser. *Proceedings ANLP-NAACL'2000*.
- Chater, N. (1999). The Search for Simplicity: A Fundamental Cognitive Principle? *The Quarterly Journal of Experimental Psychology*, 52A(2), 273-302.
- Chomsky, N. (1965). *Aspects of the Theory of Syntax*, Cambridge, The MIT Press.
- Collard, R., P. Vos and E. Leeuwenberg, (1981). What Melody Tells about Metre in Music. *Zeitschrift für Psychologie*. 189, 25-33.
- Collins, M. (2000). Discriminative Reranking for Natural Language Parsing, *Proceedings ICML-2000*, Stanford, Ca.
- Collins, M. and N. Duffy (2002). New Ranking Algorithms for Parsing and Tagging: Kernels over Discrete Structures, and the Voted Perceptron. *Proceedings ACL'2002*, Philadelphia.
- Frazier, L. (1978). *On Comprehending Sentences: Syntactic Parsing Strategies*. PhD. Thesis, University of Connecticut.
- Goodman, J. (2002). Efficient Parsing of DOP with PCFG-Reductions. In R. Bod et al. 2002b.
- von Helmholtz, H. (1910). *Treatise on Physiological Optics* (Vol. 3), Dover, New York.
- Jurafsky, D. (2002). Probabilistic Modeling in Psycholinguistics: Comprehension and Production. In R. Bod et al. 2002a.
- Kersten, D. (1999). High-level vision as statistical inference. In S. Gazzaniga (ed.), *The New Cognitive Neurosciences*, Cambridge, The MIT Press.
- Leeuwenberg, E. (1971). A Perceptual Coding Language for Perceptual and Auditory Patterns. *American Journal of Psychology*. 84, 307-349.
- Lerdahl, F. and R. Jackendoff (1983). *A Generative Theory of Tonal Music*. The MIT Press.
- Longuet-Higgins, H. (1976). Perception of Melodies. *Nature* 263, October 21, 646-653.
- Longuet-Higgins, H. and C. Lee, (1987). The Rhythmic Interpretation of Monophonic Music. In: *Mental Processes: Studies in Cognitive Science*, Cambridge, The MIT Press.
- Manning, C. and H. Schütze (1999). *Foundations of Statistical Natural Language Processing*. Cambridge, The MIT Press.
- Marcus, M., B. Santorini and M. Marcinkiewicz (1993). Building a Large Annotated Corpus of English: the Penn Treebank, *Computational Linguistics* 19(2).
- Marr, D. (1982). *Vision*. San Francisco, Freeman.
- Mumford, D. (1999). The dawning of the age of stochasticity. Based on a lecture at the Accademia Nazionale dei Lincei. (available at <http://www.dam.brown.edu/people/mumford/Papers/Dawning.ps>)
- Osborne, M. (1999). Minimal description length-based induction of definite clause grammars for noun phrase identification. *Proceedings EACL Workshop on Computational Natural Language Learning*. Bergen, Norway.
- Palmer, S. (1977). Hierarchical Structure in Perceptual Representation. *Cognitive Psychology*, 9, 441-474.

- Raphael, C. (1999). Automatic Segmentation of Acoustic Musical Signals Using Hidden Markov Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21, 360-370.
- Rissanen, J. (1989). *Stochastic Complexity in Statistical Inquiry*. Series in Computer Science - Volume 15. World Scientific, 1989.
- Saffran, J., M. Loman and R. Robertson (2000). Infant Memory for Musical Experiences. *Cognition* 77, B16-23.
- Schaffrath, H. (1995). The Essen Folksong Collection in the Humdrum Kern Format. D. Huron (ed.). Menlo Park, CA: Center for Computer Assisted Research in the Humanities.
- Shannon, C. (1948). A Mathematical Theory of Communication. *Bell System Technical Journal*. 27, 379-423, 623-656.
- Simon, H. (1972). Complexity and the Representation of Patterned Sequences of Symbols. *Psychological Review*. 79, 369-382.
- Temperley, D. (2001). *The Cognition of Basic Musical Structures*. Cambridge, The MIT Press.
- Wertheimer, M. (1923). Untersuchungen zur Lehre von der Gestalt. *Psychologische Forschung* 4, 301-350.
- Wundt, W. (1901). *Sprachgeschichte und Sprachpsychologie*. Engelmann, Leipzig.

The Spiral Array: An Algorithm for Determining Key Boundaries

Elaine Chew

University of Southern California, Integrated Media Systems Center,
Daniel J. Epstein Department of Industrial and Systems Engineering,
3715 McClintock Avenue GER240 MC:0193, Los Angeles CA 90089, USA,
echew@usc.edu; <http://www-rcf.usc.edu/~echew>.

Abstract. Computer models for determining key boundaries are important tools for computer analysis of music, computational modeling of music cognition, content-based categorization and retrieval of music information and automatic generating of expressive performance. This paper proposes a Boundary Search Algorithm (BSA) for determining points of modulation in a piece of music using a geometric model for tonality called the Spiral Array. For a given number of key changes, the computational complexity of the algorithm is polynomial in the number of pitch events. We present and discuss computational results for two selections from J.S. Bach's "A Little Notebook for Anna Magdalena". Comparisons between the choices of an expert listener and the algorithm indicates that in human cognition, a dynamic interplay exists between memory and present knowledge, thus maximizing the opportunity for the information to coalesce into meaningful patterns.

1 Introduction

This paper proposes a method for segmenting a piece of music into its respective key areas using the Spiral Array model. The *key* of a musical piece imparts information about the principal pitch set of the composition. More importantly, it identifies the most stable pitch, the *tonic*, and its hierarchical relations to all other pitches in that pitch set. The *tonic* is sometimes referred to as the *tonal center*, or simply as "doh". Once the key is established, ensuing notes are heard in reference to the tonic. This reference point allows the listener to navigate through the soundscape generated by the piece. The computational modeling of the human ability to determine the key of a melody is a well-studied problem and numerous methods have been proposed for mimicking this cognitive ability. See, for example, [5], [13], [16] and [21]; and, [1], [18] and [22] in [10].

Modulation or the change of key is a common compositional device. When the key changes, each pitch assumes a different role in relation to the new tonal center. This shift in paradigm transforms the tonal context in which the notes are heard. One way in which the human mind organizes music information into coherent chunks is by way of its key contexts. Conversely, key changes often

occur at structural boundaries. The skill with which a composer moves persuasively or abruptly from one context to another constitutes an important feature of her style. A performer may choose to highlight unusual modulations through dynamic or rate changes. These are but a few examples to illustrate the importance of developing robust and computationally viable methods for determining key boundaries. Computer modeling of the cognitive task of determining key boundaries is potentially an exponentially hard problem, and very little work has been done in this area.

By using the Spiral Array and by incorporating music knowledge in modeling the problem, this paper proposes a computationally efficient, knowledge-based way to determine points of modulation in a piece of music. The Spiral Array is a spatial arrangement of pitch classes, chords and keys proposed by Chew [6] as a geometric model for *tonality*, the system of musical principles underlying tonal music. According to Bamberger [2], “*Tonality and its internal logic frame the coherence among pitch relations in the music with which [we] are most familiar.*” The perception of key results from this system of hierarchical relations amongst pitches.

The Spiral Array is derived from the Harmonic Network (described in various contexts in [7], [8], [14], [15] and [16]), an arrangement of pitch classes on a lattice. Chord pitches and pitches in a key map to compact sets on this grid. The Spiral Array preserves the clustering of tonally important pitch sets. In addition, its 3D configuration allows higher level musical entities to be defined and embedded in the interior of the structure. A generating principle in the Spiral Array is that higher level musical entities can be represented unambiguously as convex combinations of their lower level components. Spatial coordinates representing each key were generated from weighted averages of its I, V and IV chord representations, which were in turn generated from each chord’s root, fifth and third pitch representations.

In the Spiral Array, musical information is condensed and summarized by 3D coordinates that represent the center of effect (c.e.) of the pitches. This idea was extended in the CEG algorithm [5] to generate c.e.’s that mapped the establishing of key over time to spatial trajectories. The distance from the moving c.e. to the key representations serves as a likelihood indicator that the piece is in that key. The distance-minimizing key is elected as the most likely solution. It was shown that the CEG algorithm offers a competitive and viable means to identify the key of a melody.

In this paper, we show that determining the points of modulation can be re-cast as a problem of finding the distance-minimizing boundaries for the different key contexts using the Spiral Array framework. For this reason, we call the algorithm the Boundary Search Algorithm (BSA). A straightforward implementation of the algorithm yields an $O(n^m)$ performance time, where n is the number of event onsets and $m(\geq 2)$ is the number of boundaries. With more information, a listener can assess the key identity with more confidence, and correspondingly, the algorithm works best when n is large and m is small.

Given a piece of music, exhaustively enumerating all segmentation possibilities based on key boundaries is a formidable and exponential task. We show that by integrating musical knowledge, one can drastically reduce the search space and propose a tractable polynomial-time solution to the problem of determining key boundaries for the two Bach examples addressed in this paper. Realistically speaking, the number of key changes, m , is typically a small number in relation to the number of event onsets, n . The two Bach examples each contain one point of departure and one return to a primary key area, that is to say, $m = 2$. The problem is further constrained by the fact that the first and third (last) key areas must be identical.

Apart from computational tractability, the benefits of a knowledge-based approach to determining key boundaries also include more accurate models for computer analysis and tracking of music, a better understanding of the perception of musical structure, more realistic computer-generated expressive performances and more robust algorithms for content-based music information retrieval.

Comparisons between a human expert's perception of key boundaries and the algorithm's choices indicate that human processing of music information, a dynamic interplay exists between past (memory) and present knowledge, thus maximizing the opportunity for the information to coalesce into meaningful patterns. Also, the human's perception of key boundaries is influenced by phrase structure.

2 A Mathematical Model for Tonality

Any computer model for processing music information must begin with a representation that converts content information to numerical data for analysis. The representation we use is the Spiral Array model first introduced in [6]. The Spiral Array grew out of a 3D configuration of the Harmonic Network to eliminate periodicity and redundancy (see Figure 1). The use of this geometric model to determine the key of a musical passage with a likelihood score based on spatial proximity is outlined in [5]. We briefly describe how key representations are generated in this structure.

2.1 Placement of Pitch Classes, Chords, and Keys

In the Spiral Array, pitch classes are represented by spatial coordinates along a spiral. Each pitch class is indexed by its number of perfect fifths from an arbitrarily chosen reference pitch, C (set at position $[0,1,0]$). An increment in the index results in a quarter turn along the spiral. Four quarter turns places pitch classes related by a major third interval in vertical alignment with each other. Strict enharmonic equivalence would require that the spiral be turned into a toroid. The spiral form is assumed so as to preserve the symmetry in the distances among pitch entities.

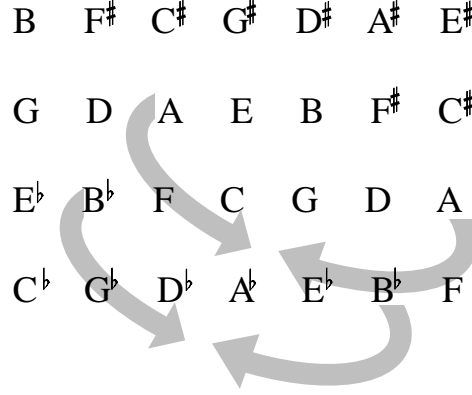


Fig. 1. Rolling up the Harmonic Network to eliminate redundancy.

Definition 1. The radius of the cylinder, r , and the height gain per quarter turn, h , uniquely define the position of a pitch representation, which can be described as:

$$\mathbf{P}(k) \stackrel{\text{def}}{=} \begin{bmatrix} x_k \\ y_k \\ z_k \end{bmatrix} = \begin{bmatrix} r \sin \frac{k\pi}{2} \\ r \cos \frac{k\pi}{2} \\ kh \end{bmatrix}.$$

Chord representations are defined uniquely as convex combinations of their component pitches. Triads map to non-overlapping triangles on the Spiral Array. Hence, each triangle can be uniquely represented by the convex combination of its component pitches. The weights are constrained to be monotonically decreasing from the root, to the fifth, to the third to mirror the relative important of each pitch to the chord.

Definition 2. The representation for a major triad is generated by the convex combination of its root, fifth and third pitch positions:

$$\mathbf{C}_M(k) \stackrel{\text{def}}{=} w_1 \cdot \mathbf{P}(k) + w_2 \cdot \mathbf{P}(k+1) + w_3 \cdot \mathbf{P}(k+4),$$

where $w_1 \geq w_2 \geq w_3 > 0$ and $\sum_{i=1}^3 w_i = 1$.

The minor triad is generated by a similar combination,

$$\mathbf{C}_m(k) \stackrel{\text{def}}{=} u_1 \cdot \mathbf{P}(k) + u_2 \cdot \mathbf{P}(k+1) + u_3 \cdot \mathbf{P}(k-3),$$

where $u_1 \geq u_2 \geq u_3 > 0$ and $\sum_{i=1}^3 u_i = 1$.

Major key representations are generated as convex combinations of their I, V and IV chords. Figure 2 shows an example of a major key representation. The weights are constrained to be monotonically decreasing from I to V to IV.

Definition 3. *A major key is represented by a convex combination of its tonic, dominant and subdominant chords. The weights are restricted to be monotonic and non-increasing.*

$$\mathbf{T}_M(k) \stackrel{\text{def}}{=} \omega_1 \cdot \mathbf{C}_M(k) + \omega_2 \cdot \mathbf{C}_M(k+1) + \omega_3 \cdot \mathbf{C}_M(k-1),$$

where $\omega_1 \geq \omega_2 \geq \omega_3 > 0$ and $\sum_{i=1}^3 \omega_i = 1$.

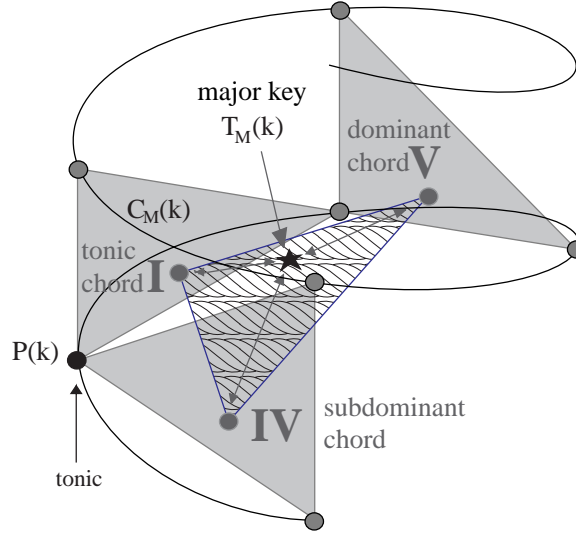


Fig. 2. Geometric representation of a major key, a composite of its tonic (I), dominant (V) and subdominant (IV) chords.

The minor key representation is a combination of the i, (V/v) and (iv/IV) chords. The additional parameters α and β model the relative importance and usage of V versus v and iv versus IV chords respectively in the minor key.

Definition 4. *The minor key representation is generated by a convex combination of its tonic, dominant (major or minor) and subdominant (major or minor) chords as follows:*

$$\begin{aligned}
\mathbf{T}_m(k) &\stackrel{\text{def}}{=} v_1 \cdot \mathbf{C}_m(k) \\
&\quad + v_2 \cdot [\alpha \cdot \mathbf{C}_M(k+1) + (1-\alpha) \cdot \mathbf{C}_m(k+1)] \\
&\quad + v_3 \cdot [\beta \cdot \mathbf{C}_m(k-1) + (1-\beta) \cdot \mathbf{C}_M(k-1)], \\
\text{where } &v_1 \geq v_2 \geq v_3 > 0 \text{ and } v_1 + v_2 + v_3 = 1, \\
&\text{and } 0 \leq \alpha \leq 1, 0 \leq \beta \leq 1.
\end{aligned}$$

See Figure 3 for the spatial representation of a minor key.

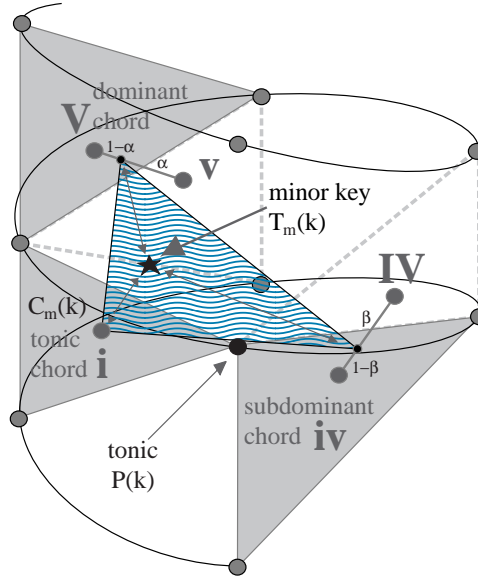


Fig. 3. Geometric representation of a minor key, a composite of its tonic (i), dominants (V/v) and subdominant (iv/IV) chords.

For the two examples in this paper, the key representations in the Spiral Array were generated by setting the weights (w, u, ω, v) to be the same and equal to $[0.516, 0.315, 0.168]$, $h = \sqrt{2/15}$ ($r=1$). α is set to 1 and β to 0. These weights satisfy perceived interval relations, such as, pitches related by a distance of a perfect fifth being closer than those a major third apart, and so on. In addition, these assignments also satisfy two other conditions: two pitches an interval of a half-step apart generates a center that is closest to the key of the upper pitch; and, the coordinates of a pitch class is closest to the key representation of the same name.

The Spiral Array turns the original 2D lattice of pitch classes into a 3D configuration that eliminates periodicity. By moving to a higher dimensional

space, the Spiral Array allows the definition of higher-level musical entities as coordinates in the same space as the pitch classes, blending both discrete and continuous space.

3 An Algorithm for Finding Key-Change Boundaries

The idea behind the Boundary Search Algorithm (BSA) is a simple one. The BSA collapses each set of pitch and duration information down to a spatial point, a center of effect (c.e.). This point is compared to the key representations in the Spiral Array Model. The better the pitch collection adheres to the pitch classes of the key in the proportions determined by their tonal relations, the closer the c.e. will be to the key representation.

When two or more key areas exist in a piece, the key areas are delineated by modulation boundaries. The problem of finding the point(s) of modulation can then be posed as a problem of determining the placement of boundaries so as to minimize the distances between the respective c.e.'s and their closest keys. This section presents a formal description of the BSA algorithm. Section 3.1 describes how music information is condensed to a spatial coordinate for comparison to a key object in the Spiral Array. Section 3.2 gives the problem statement the solution to which will determine the optimal choice of modulation boundaries.

3.1 The Center of Effect and Distance to Key

A musical passage consists of a collection of pitch events, each comprising pitch and duration information. If \mathbf{p}_i represents the position of the i -th pitch class in the Spiral Array, and d_i represents its duration, the collection of N notes can be written as $\{(\mathbf{p}_i, d_i) : i = 1 \dots N\}$. The center of effect of this pitch collection is defined as:

$$\mathbf{c} \stackrel{\text{def}}{=} \sum_{i=1}^N \frac{d_i}{D} \cdot \mathbf{p}_i \quad \text{where } D = \sum_{i=1}^N d_i$$

If \mathbf{T} is a finite collection of keys represented in the Spiral Array, both major and minor, that is to say:

$$\mathbf{T} = \{\mathbf{T}_m(k) \forall k\} \cup \{\mathbf{T}_M(k) \forall k\},$$

then, the most likely key of the passage is given by:

$$\arg \min_{T \in \mathbf{T}} \|\mathbf{c} - T\|,$$

which can be readily implemented using a nearest neighbor search. The likelihood that the pitch collection is in any given key is indicated by its proximity to that key. The minimizing distance, also a likelihood indicator, is given by:

$$d^{\min} = \min_{T \in \mathbf{T}} \|\mathbf{c} - T\|.$$

3.2 Problem Statement and Algorithm

Suppose that m boundaries have been chosen as given by (B_1, \dots, B_m) . Let time 0 be B_0 , and the end of the passage be $B_{m+1} = L$. The note material between any two boundaries (B_i, B_{i+1}) generates a c.e. given by $\mathbf{c}_{(B_i, B_{i+1})}$. The most likely key has a distance $d_{(B_i, B_{i+1})}^{\min}$ from this c.e.

Our hypothesis is that the best boundary candidates will segment the piece in such a way as to minimize the distances $d_{(B_i, B_{i+1})}^{\min}, i = 0, \dots, m$. Hence, The goal is to find the boundaries (B_1, \dots, B_m) that minimize the sum of these distances. Let n_i be the number of pitch events from the beginning up to the boundary B_i , with the appropriate boundary value of $n_0 = 0$. The optimal boundaries, (B_1^*, \dots, B_m^*) , are then given by the solution to the following system:

$$\begin{aligned} \min \quad & \sum_{i=0}^m d_{(B_i, B_{i+1})}^{\min} \\ \text{s.t.} \quad & d_{(B_i, B_{i+1})}^{\min} = \min_{T \in \mathbf{T}} \|\mathbf{c}_{(B_i, B_{i+1})} - T\|, \quad i = 0, \dots, m \\ & \mathbf{c}_{(B_i, B_{i+1})} = \sum_{j=n_i+1}^{n_{i+1}} \frac{d_j}{D_i} \cdot \mathbf{p}_j \text{ where } D_i = \sum_{k=n_i+1}^{n_{i+1}} d_k, \quad i = 0, \dots, m \\ & \text{and } B_i < B_{i+1}, \quad i = 0, \dots, m \end{aligned}$$

A naive and viable approach to the problem is to enumerate all possible sets of boundaries and evaluates each objective function value numerically in order to find the objective-minimizing boundaries. Assuming that the set of keys that are of interest \mathbf{T} is finite, and that any piece of music is of finite length with n event onsets, the computational complexity of this approach is $O(n^m)$.

The practical application of this method typically yields far better performance than that implied by the $O(n^m)$ computational time. The number of key changes and associated boundaries is typically small in relation to the number of event onsets ($m \ll n$). In smaller-scale compositions such as the two Bach examples from “A Little Notebook for Anna Magdalena” discussed in the next section, the pieces can typically be segmented into three key areas.

Using music knowledge and a little common sense, we can add a few constraints to further reduce the search space. Adjacent key areas should be distinct from each other, that is to say, $T_i \neq T_{i+1} (i = 1 \dots m)$. If the passage to be analyzed is a complete piece, the first and last key areas should be constrained to be the same, with the corresponding constraint being $T_1 = T_{m+1}$. For example, in pieces with three key areas, the first and last segments are in the same key as that of the composition, while the middle portion is in a different key.

4 Two Examples

We now proceed to apply the solution framework to two examples from J. S. Bach’s “A Little Notebook for Anna Magdalena”. The two examples chosen

each contain one departure and one return to the primary key area. The task at hand is to find the two boundaries, B_1 and B_2 separating the primary and the secondary key areas.

In these two problem instances, the three segments are required to satisfy the constraint that parts 1 and 3 must be in the same key ($T_1 = T_3$), and that this key must be distinct from that of part 2 ($T_1 \neq T_2$). These additional constraints greatly reduce the solution space. The reduced and modified problem instance is as follows:

$$\begin{aligned}
& \min d_{(B_0, B_1)}^{\min} + d_{(B_1, B_2)}^{\min} + d_{(B_2, B_3)}^{\min} \\
& \text{s.t. } d_{(B_i, B_{i+1})}^{\min} = \|\mathbf{c}_{(B_i, B_{i+1})} - T_i\|, \quad i = 0, 1, 2 \\
& \quad T_i = \arg \min_{T \in \mathbf{T}} \|\mathbf{c}_{(B_i, B_{i+1})} - T\|, \quad i = 0, 1, 2 \\
& \quad \mathbf{c}_{(B_i, B_{i+1})} = \sum_{j=n_i+1}^{n_{i+1}} \frac{d_j}{D_i} \cdot \mathbf{p}_j \text{ where } D_i = \sum_{k=n_i+1}^{n_{i+1}} d_k, \quad i = 0, 1, 2 \\
& \quad T_1 = T_3 \neq T_2 \\
& \text{and } 0 = B_0 < B_1 < B_2 < B_3 = L
\end{aligned}$$

In both instances, the BSA method identified the correct keys, $T_1 (= T_3)$ and $T_2 (\neq T_1)$, and determined the boundaries B_1 and B_2 . These boundaries are compared to a human expert's choices. The BSA's boundaries (A1, A2) and the expert's choices (EC1, EC2) are close but not identical in both cases. Sections 4.1 and 4.2 compare and discuss the rationale behind, and the relative merits of, these solutions.

4.1 Example 1: *Minuet in G* by Bach

The first example is Bach's *Minuet in G* from "A Little Notebook for Anna Magdalena". The Minuet is in the key of G, with a middle section in D major. Figure 4 shows a segment of the piece that contains the transitions away from and back to the intended key G. On the figure, are markings denoting a human expert's choice of boundaries (indicated by EC1 and EC2) between the keys of G and D. Also marked, are the BSA's choices for these same boundaries (indicated by A1 and A2).

The difference between A1 and EC1 is one between a synthetic and analytic choice. The human expert chose EC1, a boundary that occurs before the D major key is established conclusively in bar 20. In light of the D major key context established in bar 20, bar 19 is heard retroactively as being in the key of D major rather than G. In D major, the right hand melody in bar 19 traverses the scale degrees $\{\hat{4}, \hat{2}, \hat{3}, \hat{4}, \hat{1}\}$. It is not an uncommon practice for a human listener to retroactively frame the past in the light of present knowledge. In addition, EC1 also prioritizes phrase symmetry by delineating a 2-bar + 2-bar phrase structure.

Fig. 4. Modulation boundaries. [EC = human expert; A = BSA algorithm]

The Boundary Search Algorithm described in Section 3 was not designed to have backward and forward analysis capabilities. It selects the boundaries that produce pitch event groupings in such a way that each grouping maps to point clusters on the Spiral Array that aggregate to points closest to the respective key representations. Based on this selection criteria, the BSA chooses the beginning of bar 20 to mark the beginning of the D major section. This choice, in fact, agrees with the textbook definition of modulation.

The textbook boundary for the next key change falls squarely between EC2 and A2. The human expert's decision was influenced by the four-bar phrasing established from the beginning of the piece. Giving preference to boundaries that lock into this four-bar period, the human listener waited for the latest phrase to end (at the end of the bar) before placing EC2 at the barline. With this choice of boundary, the D and C \sharp on beats 2 and 3 in bar 24 are heard as an ending of the middle section. A performer, if attempting to highlight this fact, would play it quite different than if these two notes were the beginning of the next section.

Now consider A2. The BSA chose to group the notes in beats two and three of bar 24 with the final G major section, a choice that is closer in spirit to the textbook solution. In this case, the D on beat two of bar 24 is heard as the fifth degree $\hat{5}$ in G major, and not the first degree $\hat{1}$ in D major. Strictly speaking, the C \sharp is the first note that belongs to G major and not D. Thus, the textbook boundary would start the G major area on beat three of bar 24.

Both the algorithm's and the expert's choices for modulation boundaries are valid for different reasons. The human expert's choices involved retroactive decisions that framed past information in the light of present knowledge. The human's choices were also influenced by phrase structure, preferring to rein-

force the four-bar phrase boundaries. The BSA considered only the pitch events, taking the pitches literally and choosing the boundaries accordingly.

4.2 Example 2: *Marche in D* by Bach

The second example is Bach’s *Marche in D*, also from “A Little Notebook for Anna Magdalena”. This piece was chosen because it is a little more complex than the previous Minuet example. In the transition from A major back to D major, the piece hints at a number of different keys before returning unequivocally to D major. The A major middle section shifts to D major in the second beat of bar 12, portending the return to D major. The D acts as a V to the G major tonality at the end of bar 13; a G# in the bass at the end of bar 14 acts as a leading note to A major; there is a momentary hint of B major between bars 15 and 16 that acts as the V of E minor; E minor acts as a V to A, which works as a V to D major which returns in bar 18. Clearly, this is a less than perfect example with which to test an algorithm that looks for boundaries between only three parts. The salient portion of the piece is shown in Figure 5, with the same labeling convention as before.

The figure shows a musical score for a portion of Bach's *Marche in D*. The score is written in treble and bass staves with a key signature of two sharps (F# and C#). The measures shown are 4, 8, 12, 16, and 20. The score includes modulation boundaries marked by EC (human expert) and A (BSA algorithm). The boundaries are labeled as follows: EC1 at measure 12, A1 at measure 14, EC2 at measure 16, and A2 at measure 18. The score also includes a 4-measure rest at measure 4 and an 8-measure rest at measure 8.

Fig. 5. Modulation boundaries. [EC = human expert; A = BSA algorithm]

The first set of boundaries, EC1 and A1, were almost identical. The BSA breaks up the phrase structure in bar 6 to include the last beat in the new A major section, while the human expert waited for the completion of the phrase to place the boundary EC1. The slight discrepancy resulted because the computer algorithm was not designed to recognize the composer's phrasings. Otherwise, the two choices are not very different. However, it is important to note that $G\sharp$ occurs on the first beat of bar 6, an indication of the new key area, A major; so, a textbook boundary should begin the A major section at the $G\sharp$.

For the second set of boundaries, marked EC2 and A2 respectively in Figure 5, the choices differed quite a bit. The human expert took into account the sequential pattern starting from the middle of bar 13 and ending at the climax in bar 16, and chose to think of the final D major section as beginning in beat one of bar 18. The BSA, on the other hand, did not account for figural groupings and sequential patterns in the notes. It chose, instead, to begin the last D major section as soon as the note material of the third part agreed with the pitch collection for D major.

5 Conclusions

The Boundary Search Algorithm is designed to segment a stream of music data into contiguous sets by key. The algorithm gave accurate assessments of the goodness-of-fit between each collection of pitch events and the key candidate to suggest optimal placements of key boundaries. For the purposes of this application, the c.e. defined in Section 3.1 summarizes only pitch and duration information. The method does not account for pitch order within each data set, nor does it incorporate beat information. With only pitch and duration information, the BSA was demonstrated to perform well in the problem of determining modulation boundaries. Given only pitch and duration information, a human would probably perform in a comparable manner. Rough temporal information is embodied in the sequence of the key segments.

The BSA does not explicitly use chord functions to determine the key areas. Chord membership and functional relations to each key area are incorporated into the Spiral Array model by design. With some additional work, a functional analysis of a musical passage can be extracted from mappings of the data to the Spiral Array model. The problem of harmonic analysis can perhaps be addressed in a future paper.

In addition, the implementation of the BSA as described in this paper is designed to analyze a piece of music in its entirety. In order for the algorithm to mimic human musical perception as information is revealed over time, the ideas outlined in this paper can be adapted to detect key boundaries in real-time. For example, in a real-time system, analyses can be performed at each time increment and the number of boundaries, m , allowed to increase by at most one.

Finally, the human mind in apprehending music information performs more than sequential data processing. Pitch and duration data is but a fraction of

the total information. The mind also organizes music data into figural groupings and phrase structures, and it can sometimes revise prior assessments based on new information. The analysis of the difference in the human and algorithmic solutions indicates that a dynamic interplay exists between memory and present knowledge so that the chances of finding meaningful patterns in the time series data is maximized.

Acknowledgements. I thank Jeanne Bamberger for her guidance and cogent advice that made this research possible; and, Martin Brody for his insightful comments on interpreting the results in an early version of this document.

References

1. Wolfgang Auhagen and Piet G. Vos: Experimental Methods in Tonality Induction Research: A Review. *Music Perception* **17:4** (2000) 417–436
2. Jeanne S. Bamberger: *Developing Musical Intuition*. Oxford University Press, NY (2000)
3. Jeanne S. Bamberger: *The Mind Behind the Musical Ear*. Harvard University Press, MA (1991)
4. Brown: Tonal implications of the diatonic set. In *Theory Only* **5** (1981) 3–21
5. Chew, E.: Modeling Tonality: Applications to Music Cognition. *Proceedings of the 23rd Annual Meeting of the Cognitive Science Society* (2001)
6. Chew, E.: *Towards a Mathematical Model of Tonality*. PhD Dissertation, MIT (2000)
7. Richard Cohn: Introduction to Neo-Riemannian Theory: A Survey and a Historical Perspective. *Journal of Music Theory* **42:2** (1998) 167–180
8. Richard Cohn: Neo-Riemannian Operations, Parsimonious Trichords, and their Tonnetz Representations. *Journal of Music Theory* **41:1** (1997) 1–66
9. Peter Desain and Henkjan Honing and Huub Vanthienen and Luke Windsor: Computational Modeling of Music Cognition: Problem or Solution? . *Music Perception* **16:1** (1998) 151–166
10. Robert Gjerdingen (ed.): *Music Perception* **17:4** (2000) Note: Special issue on Tonality Induction
11. Oswald Jonas: *Introduction to the theory of Heinrich Shenker: the nature of the musical work of art*. Longman, NY. (1982)
12. Krumhansl, C. L.: Perceived Triad Distance: Evidence Supporting the Psychological Reality of Neo-Riemannian Transformations. *Journal of Music Theory* **42:2** (1998) 254–281
13. Krumhansl, Carol L.: *Cognitive Foundations of Musical Pitch*. Oxford University Press, NY (1990)
14. David Lewin: *Generalized Musical Intervals and Transformations*. Yale University Press, CT (1987)
15. H. C. Longuet-Higgins: *Mental Processes*. MIT Press, MA (1987)
16. H. C. Longuet-Higgins and M. J. Steedman: On Interpreting Bach. B. Meltzer and D. Michie (eds.) *Machine Intelligence* **6** Edinburgh U. Press (1971) 221
17. Robert Rowe: Key Induction in the Context of Interactive Performance. *Music Perception* **17:4** (2000) 511–530
18. Shmulevich, I., Yli-Harja, O.: Localized Key-Finding: Algorithms and Applications. *Music Perception* **17:4** (2000) 531–544

19. Steedman, Mark: The well-tempered computer. *Phil. Trans. R. Soc. Lond.* **349** (1994) 115–131
20. Temperley, David: What’s Key for Key? The Krumhansl-Schmuckler Key-Finding Algorithm Reconsidered. *Music Perception* **17**:1 (1999) 65–100
21. Temperley, David: The Perception of Harmony and Tonality: An Algorithmic Approach. PhD dissertation, Columbia University (1996)
22. Piet G. Vos and Erwin W. Van Geenen: A Parallel-Processing Key-Finding Model. *Music Perception* **14**:2 (1996) 185–223

Representation and Discovery of Vertical Patterns in Music

Darrell Conklin

Computational Biology
ZymoGenetics, Inc.
Seattle, USA
conklin@zgi.com

Abstract. The automated discovery of recurrent patterns in music is a fundamental task in computational music analysis. This paper describes a new method for discovering patterns in the vertical and horizontal dimensions of polyphonic music. A formal representation of music objects is used to structure the musical surface, and several ideas for viewing pieces as successions of vertical structures are examined. A knowledge representation method is used to view pieces as sequences of relationships between music objects, and a pattern discovery algorithm is applied using this view of the Bach chorale harmonizations to find significant recurrent patterns. The method finds a small set of vertical patterns that occur in a large number of pieces in the corpus. Most of these patterns represent specific voice leading formulae within cadences.

1 Introduction

The discovery of repeated structure in data is a fundamental form of inductive inference. In music, the discovery of repeated patterns within a single piece is a precursor to revealing its construction in terms of basic structures such as themes, motifs, and paradigmatic types [1]. The discovery of patterns that occur across many pieces in a style can yield signature motifs that can be used in the synthesis [2] and classification [3] of new pieces in the style. Regardless of the analytical task, computational methods for automated pattern discovery in music can be guided by two basic principles. First, to detect common structure, they should not be affected by typical musical transformations such as transposition and simple melodic elaboration. Second, to be of practical use, they should attempt to limit their output to patterns that have musical or statistical significance [4,5].

The computational analysis of polyphonic music presents unique challenges not encountered in the study of isolated melody. Individual voices, while moving separately through the horizontal or melodic dimension, create vertical sonorities by temporal overlap with notes in other voices. While each individual voice follows general principles of voice leading, sequences of vertical structures outline a coherent harmonic motion. To adequately model polyphonic music, a knowledge representation method must allow for efficient retrieval, manipulation, and reasoning about concurrent and successive music objects.

Most approaches to automated pattern discovery in music have restricted their attention to an isolated melodic line of a single piece [6,7,8,9]. This paper will report on an abstract representation method for music in which both melodic and harmonic structures of polyphonic music can be described. A new representation method, extending recent work on *viewpoints* in music [4], is used for the description of relationships between successive vertical structures. A pattern discovery algorithm is used to find significant recurrent vertical structures in multiple pieces. Initial results of this technique with the Bach chorales are presented. The representation and discovery method produces several significant vertical patterns that occur in a large number of pieces in the corpus.

The paper concludes with a discussion of the strengths and limitations of the approach, and ideas for future research.

2 Methods

2.1 Music Objects

In our ontology of music we consider three types of musical objects: notes, simultaneities, and sequences. All music objects in a piece acquire an *onset time* and *duration*; these temporal attributes will be assumed present throughout the paper and will not be explicitly notated. The *Note* is the basic primitive music object. The *Sim* and *Seq* object types are polymorphic and can contain any other type of music object as components, including not only *Note* objects, but (recursively) other *Sim* and *Seq* objects.

The primitive type *Note* is defined as a tuple of pitch class and octave, or alternatively, as an integer Midi number. Music objects are then defined by the recursive algebraic data type M :

$$\begin{aligned} M &::= \textit{Note} \mid \textit{Seq}(M) \mid \textit{Sim}(M) \\ \textit{join} &: \textit{Seq}(X) \times X \rightarrow \textit{Seq}(X) \\ \textit{layer} &: \textit{Sim}(X) \times X \rightarrow \textit{Sim}(X) \end{aligned} \tag{1}$$

Informally, objects of type $\textit{Seq}(X)$ are constructed by joining together music objects of type X , which do not overlap in time. Objects of type $\textit{Sim}(X)$ are constructed by layering objects of type X , all of which have the same onset time.

Throughout this paper, square brackets $[]$ will be used to denote a sequence. A sequence $[o_1, \dots, o_n]$ of joined music objects will be abbreviated as $\overline{o_n}$. Angle brackets $\langle \rangle$ will be used to denote a simultaneity.

This music ontology provides a natural way to describe polyphonic music in a hierarchical fashion. Naturally, similar music ontologies have appeared elsewhere, for example in the algebraic data type of Hudak et al. [10] and in the music structures approach of Balaban [11]. The OpenMusic environment also uses a similar ontology with simple elements, sequences, and superpositions as objects [12]. It appears that the various instances of this ontology differ mainly in the temporal overlapping restrictions imposed on objects, whether recursive embedding of objects is permitted, and whether rests are primitive objects.

2.2 Structuring and Partitioning of Pieces

A polyphonic piece of music is encoded at the basic level by a $Sim(Seq(Note))$ object; that is, by layered voices similar to a Midi multiple track encoding of a piece. For example, the Bach chorale fragment in Figure 1 would be encoded as four simultaneous sequences (octave encoding not shown):

$$\langle [A, D, A, B, A, G, F\sharp, F\sharp], \\ [D, D, D, D, D, C\sharp, D, D], \\ [F\sharp, F\sharp, G, A, G, F\sharp, E, D, E, D, D], \\ [D, B, F\sharp, G, A, A, D, D] \rangle$$

where $\langle \rangle$ denote a Sim object and $[]$ denote a Seq object.

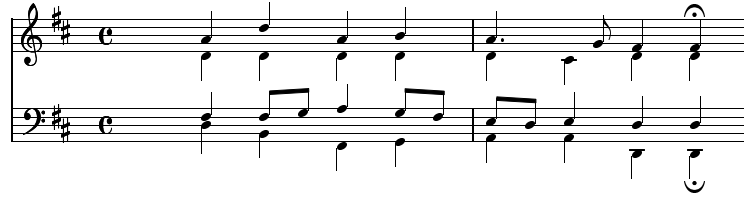


Fig. 1. From Bach, chorale BWV 262.

The pattern discovery algorithm used in this paper rapidly finds recurrent patterns in abstract sequences derived from the basic encodings of multiple pieces. Therefore, as a prelude to finding vertical patterns, it is necessary to restructure or partition the basic encoding of a piece from a simultaneity of Seq objects to a sequence of Sim objects. The music object data type is well-suited to this task because it allows the musical surface to be structured in many alternative ways.

There are two basic ways to restructure a piece into a sequence of Sim objects. The *natural* partitioning simply creates a new object whenever all voices have the same onset time. For example, the second bar of Figure 1 contains three natural $Sim(Seq(Note))$ objects, in the sequence:

$$[\langle [A, G], [D, C\sharp], [E, D, E], [A, A] \rangle, \\ \langle [F\sharp], [D], [D], [D] \rangle, \\ \langle [F\sharp], [D], [D], [D] \rangle]$$

Though sufficient for simple species counterpoint, this example illustrates that the natural partitioning technique will under-partition more complex polyphony because common onset times might not be equivalent to the harmonic rhythm of the piece. A standard solution to this problem is to perform a *full expansion* of the piece, wherein pitches are duplicated at every unique onset time. For

example, Figure 2b illustrates the expansion of the fragment in Figure 2a. This expansion would be represented by $Sim(Note)$ objects, in the sequence:

$$[\langle B, D, B, G \rangle, \\ \langle B, E, C, G \rangle, \dots, \\ \langle A, F\sharp, C, A \rangle]$$

A piece can now effectively be treated as a sequence of regular simultaneities. However, full expansion might over-partition a piece because simultaneities can be created even when there is no change in harmony. Pardo and Birmingham [13] consider as a solution a subsequent re-grouping of expanded partitions to correspond to predicted harmonic units. In the present method, the expanded piece can either be modeled directly or sampled at regular metric intervals using threaded viewpoints. This technique will be described below.



Fig. 2. From Bach, chorale BWV 260. a) original, b) expanded.

Full expansion of complex polyphony to reveal vertical structures for music generation is performed by Lartillot et al. [14] and Alamkan et al. [15]. Lemström and Tarhio [16] use full expansion in the context of polyphonic music information retrieval. Metric sampling of vertical structures for music generation is also performed by Ponsford and Wiggins [17]. The Humdrum toolkit [18] has a “ditto” function that performs full expansion.

2.3 Typed Viewpoints

A *viewpoint* is a mathematical function that computes features of music objects in a sequence, including features that denote relationships between objects within the sequence. Earlier work on viewpoints [19,4] was restricted to sequences of notes. With music objects defined as above, it is possible to extend the capability of viewpoints to model sequences of any type of music object.

A *viewpoint of type X* maps objects of type $Seq(X)$ to *viewpoint elements*. For example, the twelve-tone melodic interval viewpoint is of type *Note*, and has integers as viewpoint elements. The pitch class interval viewpoint (interval modulo 12) is of type *Note*, and has integers between 0 and 11 as viewpoint elements. Features of melodic segments [8,20] might be described by viewpoints of type $Seq(Note)$ with lists as viewpoint elements. Later in the paper we show how to construct viewpoints of type $Sim(X)$ from base viewpoints of type *X*.

The semantics of a viewpoint is that a viewpoint element models the attribute of an object or the relation between an object and the sequence of objects that precede it in a piece. More precisely, for a viewpoint τ , if

$$\tau(\text{join}(c, o)) = v \quad (2)$$

we infer that the music object o has the attribute value v in the context of the sequence c . This construction highlights three points; first, that a *relation* between music objects can be encoded as an *attribute* of the latest object. For example, the melodic interval between o and the last object in c can be encoded as an attribute of o . A single viewpoint encodes only one such relationship, because τ is a function. Second, the complete context preceding an object, and not only the immediately preceding object, is used to compute its attribute. This is to allow viewpoints to model relations between any number of preceding objects. Third, a viewpoint cannot look ahead in the sequence; the viewpoint element of an object depends on preceding objects in the sequence. This is because viewpoints were initially designed for left-to-right music prediction and generation [19].

2.4 Viewpoint Sequences

Given a sequence $\overline{o_n}$ of music objects of some type X , and a viewpoint τ of the same type X , the *viewpoint sequence* for $\overline{o_n}$ is a list of defined viewpoint elements which is the application of τ to each initial segment of $\overline{o_n}$:

$$[\tau(\overline{o_1}), \tau(\overline{o_2}), \dots, \tau(\overline{o_n})] \quad (3)$$

For example, for the melodic interval viewpoint, the viewpoint sequence for the soprano line $[A, D, A, B, A, G, F\sharp, F\sharp]$ of Figure 1 is

$$[5, -5, 2, -2, -1, 0] \quad (4)$$

Viewpoints can be seen as a method for handling sparse musical data. A viewpoint sequence is more abstract than the musical surface, and therefore significant recurrent patterns are more likely to be revealed within viewpoint sequences than within encodings of musical surfaces. In this sense, viewpoints are analogous to *class-based models* [21] of natural language, which view sentences as sequences of abstract word features. In a similar way, a viewpoint encodes notes by abstract equivalence classes rather than by pitch and duration.

2.5 Viewpoint Patterns

A *viewpoint pattern* is a viewpoint sequence fragment encountered in a corpus of pieces. Whereas music objects and viewpoint sequences represent individual, concrete entities, viewpoint patterns represent abstractions, or concepts. A pattern *occurs* in a piece if it is contained in the viewpoint sequence for that piece. The *coverage* of a set of patterns is the number of pieces containing one or more

patterns in the set. The *p-value* of a pattern that occurs n times in a corpus is the probability that the pattern can be found n or more times in a random corpus of the same size (see [4] and [5] for details on p-value computation). A *significant* pattern has a low p-value (say, < 0.01).

A viewpoint pattern P *subsumes* another viewpoint pattern P' , written $P \succeq P'$, if P is contained in P' . For example, for the melodic interval viewpoint,

$$[2, -2] \succeq [5, -5, 2, -2, -1, 0] \quad (5)$$

If $P \succeq P'$, then the coverage of P is always greater than the coverage of P' . Given a set of significant patterns, a *shortest significant pattern* is subsumed by no other pattern in the set, and a *longest significant pattern* subsumes no other pattern in the set.

Shortest and longest significant patterns are useful in different analytical situations. Shortest significant patterns are interesting because they are the most general significant patterns found in a corpus (by definition, any shorter pattern would not be significant) and can be found in many pieces, including those not in the analysis corpus. Longest significant patterns can highlight extensive similarity within a single piece or a few pieces. In the study of the Bach chorales reported below, shortest significant patterns are found within a large corpus.

The pattern discovery algorithm used below in Results employs a suffix tree data structure [22] to rapidly enumerate recurrent patterns in a viewpoint sequence representation of the corpus [4]. Pattern p-values are computed for all recurrent patterns and the longest and shortest significant patterns are then revealed by performing pattern subsumption tests.

2.6 Constructing Vertical Viewpoints

A powerful feature of viewpoints as a representation formalism for music is that complex viewpoints can be constructed from simpler ones using viewpoint constructors. For example, two viewpoints might be linked together to produce a new one that computes pairs of viewpoint elements; or a viewpoint might be threaded through a piece at defined metric locations [4]. In this section a new constructor that creates vertical viewpoints is informally described.

There are many possible viewpoints for *Sim(Note)* objects, many drawing from atonal music theory research. A basic requirement for the analysis of multiple pieces is that a viewpoint τ is transposition invariant, meaning that $\tau(X) = \tau(T(X))$ for any transposition T of the notes embedded in a sequence X of music objects. Though the pitch class set is not invariant, various types of interval functions [23] are invariant. The harmonic function of a chord is invariant and abstract in that it ignores voicing. Figured bass notation is invariant provided that the bass note is expressed as a relation to a tonic rather than as an absolute pitch class. Various types of interval vectors are invariant; for example, intervals between a bass note and all other voices, or between all pairs of objects in a simultaneity [18]. Finally, we mention as candidates formal groups of transformations of triads including Parallel (major to minor mode) and Relative (major to relative minor) [24].

With the exception of figured bass notation, the viewpoints presented above do not retain information about voice leading. Encoding voice leading information is possible with viewpoints. A viewpoint of type X can be “lifted” to form a new (vertical) viewpoint of type $\text{Sim}(X)$. This new viewpoint represents relations between all objects in a simultaneity and preceding objects in the same voice. Figure 3 illustrates an example of this technique for the melodic interval viewpoint.

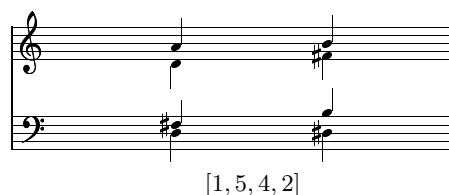


Fig. 3. The melodic interval viewpoint, of type $Note$, is lifted to a new viewpoint of type $\text{Sim}(Note)$. The new viewpoint returns a tuple of intervals between all voices.

As another example, the pitch class interval viewpoint (interval modulo 12), threaded on quarter note beats, and lifted, when applied to the expanded fragment of Figure 2b yields the viewpoint sequence

$$[[11, 3, 4, 10], [10, 2, 1, 3], [8, 0, 0, 9]] \quad (6)$$

representing the vertical pitch class interval between the $\text{Sim}(Note)$ objects at beats 2, 3, and 4 of the fragment. Viewpoints can be threaded using any other time span; for the Bach chorales a quarter note time span is meaningful because it corresponds to harmonic rhythm.

Though the examples above used melodic and pitch class interval viewpoints, the vertical viewpoint construction technique can be applied to any other base viewpoint, for example, scale degree of a note, diatonic intervals, and melodic contour. Furthermore, the technique is not restricted to $Note$ viewpoints, and can be applied to any type of music object and appropriately typed base viewpoint.

3 Results

The typed viewpoint scheme described above in Methods was applied to 185 Bach four-part chorales. Pieces were expanded, and viewed using the pitch class interval viewpoint, threaded at every quarter note beat, and lifted. There are approximately 9000 vertical structures in this data set. The pattern discovery algorithm discussed in Methods was used to find all shortest significant patterns.

A total of 32 shortest significant patterns was found (see Table 1). The coverage of this set was 142 pieces; therefore, the majority of pieces in the corpus

contain at least one occurrence of one of the 32 patterns. Table 1 also presents a harmonic analysis of instances of the pattern. For these significant patterns two lifted voice leading intervals suffice to fully specify the chord progression.

Table 1. The top 10 shortest significant vertical pitch class interval patterns found in the Bach chorales. The harmonic function of instances of the pattern is indicated. Brackets enclose a chord that contains an accented non-harmonic tone.

	Pattern	p-value	coverage	function
1	[[2,10,11,0],[5,9,8,10]]	≈ 0	36	$\text{ii}_5^6 - V - I$
2	[[2,5,11,0],[5,2,8,10]]	5e-14	19	$\text{ii}_5^6 - V - I$
3	[[2,11,11,0],[5,8,8,10]]	2e-11	19	$\text{ii}_5^6 - V - i$
4	[[2,11,10,0],[5,8,9,10]]	2e-10	16	$\text{ii}_5^6 - V - I$
5	[[4,0,0,11],[0,0,11,0]]	2e-06	15	$i^6 - V_4^5 - V_3^5$
6	[[0,0,11,0],[5,9,8,10]]	9e-06	20	$V_4^5 - V_3^5 - I$
7	[[2,6,11,0],[5,1,8,10]]	4e-05	7	$\text{ii}_5^6 - V - i$
8	[[3,0,0,0],[4,0,0,11]]	2e-05	12	$i - i^6 - V_4^5$
9	[[5,0,2,10],[2,11,10,0]]	2e-05	10	$I - \text{ii}_5^6 - V$
10	[[9,11,0,2],[0,10,0,1]]	2e-05	6	$i^6 - (V^6) - i$



Fig. 4. A selection of instances of the $\text{ii}_5^6 - V - I$ cadential formula represented by pattern 1 from Table 1. Clockwise from upper left: from Bach, chorales BWV 329, 361, 356, 379, 380, 355. The complete bars containing the pattern are notated.

Pattern 1 of Table 1 is the most statistically significant shortest pattern found. Figure 4 presents a sample of various instances of this pattern, which outlines a standard $ii_5^6 - V - I$ cadential formula. It is apparent that a wide diversity of melodic diminution is applied to this pattern, and that the method has succeeded in identifying a deeper structure than is evident in the musical surface alone. Passing tones, escape tones, anticipations and consonant skips are encountered. Instances of this pattern have a two beat repetition of the soprano $\hat{2}$ followed by $\hat{1}$. The leading tone in the alto voice falls by skip to the $\hat{5}$ in the final I. The motion of the bass from $V - I$ is either ascending or descending; the pitch class interval viewpoint is invariant to interval direction. The pattern occurs in both 3/4 and 4/4 meters. It is evident that expansion of the pieces was necessary in order to reveal this pattern. Furthermore, the pitch class interval viewpoint was necessary, and the pattern could not be revealed in its generality by searching for transposed pitch repetition.

A full analysis of the discovered vertical patterns will appear elsewhere; here we point out that several patterns are variants of others, for example, patterns 1 and 4 simply swap the alto and tenor voices. Instances of patterns 1, 2 and 4 are alternative voicings of the $ii_5^6 - V - I$ formula. Patterns 3 and 7 are minor mode versions of this cadential formula. If desired, viewpoints could be easily designed that are invariant to these effects of voicing and mode. Other patterns, for example, 5 and 8, overlap and are joined into a longer pattern simply by instructing the algorithm to seek longest rather than shortest significant patterns (results of longest significant pattern discovery not shown).

4 Discussion

A knowledge representation method for music should be able to express common abstract musical patterns. This paper has used a music representation method that allows a piece to be alternatively structured to reveal both horizontal and vertical patterns. The two dimensions of polyphonic music were coupled by constructing a lifted interval viewpoint that takes voice leading into account. Though several interesting patterns were found, a more complete study of different vertical viewpoints should be done. It is expected that viewpoints achieving an appropriate level of abstraction will give rise to extensive significant patterns.

A shortcoming of this technique for describing relationships between vertical objects is that a polyphonic piece must initially be structured as a set of monophonic voices. This precludes its use on free-form Midi data, where a single track can contain concurrent events, and where there may be no natural way to divide the track into voices. Nevertheless, this is a limitation only of lifted vertical viewpoints, and other vertical viewpoints suggested in this paper could be applied to free-form Midi data.

Related work on polyphonic music has produced other interesting ideas for representing vertical structures. Meredith et al. [25] describe a representation in which notes are encoded as multidimensional vectors, one dimension for each feature of a note, and repeated patterns within a piece are geometric translations

of vectors. Farbood and Schoner [26] use harmonic intervals as one component of a model for predicting first species counterpoint to a cantus firmus line. As discussed above in Methods, several groups have considered techniques for partitioning pieces into vertical objects.

A weakness of the expansion and sampling technique for partitioning polyphonic music into vertical structures is that it will unavoidably sample accented non-harmonic tones. For example, whenever a suspension occurs, after expansion the duplicated suspended note will become a non-harmonic tone. The vertical structures containing such non-harmonic tones might in fact be recurrent, but a deeper structure could probably be discovered if the non-harmonic tones were resolved prior to pattern discovery [17,27]. An interesting approach to this problem would be to first perform a limited Schenkerian-style reduction of the musical surface, followed by expansion and pattern discovery. To implement this idea effectively using current pattern discovery methods, the reduction process should produce only one deterministic reduction of a piece.

References

1. N. Cook. *A Guide to Musical Analysis*. Oxford University Press, 1987.
2. D. Cope. *Computers and Musical Style*. A-R Editions, 1991.
3. M. Westhead and A. Smaill. Automatic characterisation of musical style. In *Music Education: An AI Approach*, pages 157–170. Springer-Verlag, 1993.
4. D. Conklin and C. Anagnostopoulou. Representation and discovery of multiple viewpoint patterns. In *Proceedings of the International Computer Music Conference*, pages 479–485, Havana, Cuba, 2001. International Computer Music Association.
5. D. Huron. What is a musical feature? Forte’s analysis of Brahms’s Opus 51, No. 1, revisited. *Music Theory Online*, 7(4), July 2001.
6. P-Y. Rolland and J-G. Ganascia. Musical pattern extraction and similarity assessment. In E. Miranda, editor, *Readings in Music and Artificial Intelligence*, chapter 7, pages 115–144. Harwood Academic Publishers, 2000.
7. J-L. Hsu, C-C. Liu, and A. Chen. Discovering nontrivial repeating patterns in music data. *IEEE Transactions on Multimedia*, 3(3):311–325, September 2001.
8. E. Cambouropoulos. *Towards a General Computational Theory of Musical Structure*. PhD thesis, University of Edinburgh, 1998.
9. A. Smaill, G. Wiggins, and M. Harris. Hierarchical music representation for composition and analysis. *Computers and the Humanities*, 93:7–17, 1993.
10. P. Hudak, T. Makuzevich, S. Gadde, and B. Whong. Haskore music notation — an algebra of music. *Journal of Functional Programming*, 6(3):465–483, May 1996.
11. M. Balaban. The music structures approach in knowledge representation for music processing. *Computer Music Journal*, 20(2):96–111, 1996.
12. C. Agon, G. Assayag, O. Delerue, and C. Rueda. Objects, time and constraints in OpenMusic. In *Proceedings of the International Computer Music Conference*, Ann Arbor, Michigan, 1998. International Computer Music Association.
13. B. Pardo and W. Birmingham. Automated partitioning of tonal music. In *Conference of the Florida Artificial Intelligence Research Society*, May 2000.
14. O. Lartillot, S. Dubnov, G. Assayag, and G. Bejerano. Automatic modeling of music style. In *Proceedings of the International Computer Music Conference*, pages 447–453, Havana, Cuba, 2001. International Computer Music Association.

15. C. Alamkan, W. Birmingham, and M. Simoni. Stylistic structures: an initial investigation of the stochastic generation of music. Technical Report CSD-TR-395-99, Electrical engineering and computer science department, University of Michigan, 1999.
16. K. Lemström and J. Tarhio. Searching monophonic patterns within polyphonic sources. In *Proc. Content-based multimedia information access (RIAO 2000)*, pages 1261–1279, 2000.
17. D. Ponsford, G. Wiggins, and C. Mellish. Statistical learning of harmonic movement. *Journal of New Music Research*, 28(2), 1999.
18. D. Huron. *Music Research using Humdrum: A user's guide*.
19. D. Conklin and I. Witten. Multiple viewpoint systems for music prediction. *Journal of New Music Research*, 24(1):51–73, 1995.
20. K. Höthker, D. Hörnel, and C. Anagnostopoulou. Investigating the influence of representations and algorithms in music classification. *Computers and the Humanities*, 35:65–79, 2001.
21. P. Brown, V. Della Pietra, P. deSouza, J. Lai, and R. Mercer. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, 1992.
22. D. Gusfield. *Algorithms on strings, trees, and sequences*. Cambridge University Press, 1997.
23. D. Lewin. Intervallic relations between two collections of notes. *Journal of Music Theory*, 3(2):298–301, 1959.
24. D. Lewin. *Generalized Musical Intervals and Transformations*. Yale University Press, 1987.
25. D. Meredith, G. Wiggins, and K. Lemström. Pattern induction and matching in polyphonic music and other multidimensional datasets. In *Proc. Conference on Systemics, Cybernetics and Informatics*, volume X, pages 61–66, 2001.
26. M. Farbood and B. Schoner. Analysis and synthesis of Palestrina-style counterpoint using Markov chains. In *Proceedings of the International Computer Music Conference*, pages 471–474, Havana, Cuba, 2001. International Computer Music Association.
27. A. Marsden. Representing melodic patterns as networks of elaborations. *Computers and the Humanities*, 35:37–54, 2001.

Discovering Musical Structure in Audio Recordings

Roger B. Dannenberg and Ning Hu

Carnegie Mellon University, School of Computer Science, Pittsburgh, PA 15217, USA
{rbd, ninghu}@cs.cmu.edu

Abstract. Music is often described in terms of the structure of repeated phrases. For example, many songs have the form AABA, where each letter represents an instance of a phrase. This research aims to construct descriptions or explanations of music in this form, using only audio recordings as input. A system of programs is described that transcribes the melody of a recording, identifies similar segments, clusters these segments to form patterns, and then constructs an explanation of the music in terms of these patterns. Additional work using spectral information rather than melodic transcription is also described. Examples of successful machine “listening” and music analysis are presented.

1 Introduction

Machine recognition of music is an important problem, with applications in databases, interactive music systems, and musicology. It would not be an exaggeration to say that music recognition and understanding is a central problem of computer music research.

Recognition and understanding are not well-defined terms in the field of music, largely because we have no general theory of semantics for music. Music often seems to have an internal logic and certainly exhibits rich structures, but we have no formal descriptions of meaning as in many other domains. For example, a speech understanding system or a natural language translation system can be evaluated objectively in terms of how well it preserves semantic information across a change of representation. Alternatively, internal representations of meaning can be examined. In contrast, music has no accepted “meaning” or semantic representation. Without a formal semantic theory, the concept of “music understanding” is on shaky ground.

One path for research is to pursue analogies to semantic theories from other fields. For example, if “recognition” is defined in terms of translation, as in most speech recognition, then the analogy to music might be music transcription, including pitch recognition, beat tracking, bass-line extraction, etc. If understanding is defined in terms of parsing, then we can pursue the parsing of music into phrases, voices, sections, etc.

Another path is to take music more on its own terms. When we listen to music, there is no question that we identify repeated patterns. It is also clear that music is full of relationships and analogies. For example, a melodic fragment can be repeated at some tonal or chromatic transposition, a melody can be repeated by a different instrument or with different orchestration, and a passage can be echoed with a change

in dynamics. Multiple relationships are common within a composition, and they can also occur between elements of a composition and elements outside of the composition. For example, when a melody contains a scale, we can easily recognize the scale even if we have not heard the melody before.

Structures formed by relationships are themselves recognizable. For example, 12-bar blues or an AABA song forms in popular music and jazz say nothing about the specific content of the music, but only about where we expect to find similarities and relationships. The fact that we describe music in this way is an important clue about the very nature of music.

We believe that musical relationships form the basis of music, and that an important component of “music understanding” is the identification of these relationships. The simplest and most salient relationship is a literal repetition of a segment of music at a later time. Other relationships involve repetition with a change of one or more aspects of the music, such as pitch or tempo. Higher order transformations are also possible.[13] Consider a phrase that is transposed upward by a step, and then the result is transposed upward by another step. In this case, it is not just the intervals but also the transposition that is repeated.

In this research, we begin with the premise that an important part of music understanding is the identification of repetition within the music. Repetition generates structure, but the recovery of structure is not trivial. We describe some new methods designed for this purpose. One important aspect of the work described here is that it starts with audio recordings of music rather than symbolic ones. While symbolic representations would be natural for this kind of work, we believe that symbolic representations oversimplify the problem. Much of our interest lies in discovering how it is possible to recover structure from the least-structured representation of music: sound itself. We are especially interested in applications to audio databases, another reason to work with audio representations.

This work began using simple monophonic pitch estimation to extract data from audio where a single voice is prominent. More recently, we have had some success with a chroma-based representation on more polyphonic or chordal music, where (current) monophonic pitch estimation techniques are not useful.

In the next section, we highlight some related work. Following that, we describe a system of programs, SAM1, that performs a structural analysis of music from audio input. In the section “Polyphonic Music,” we describe experiments to use spectra rather than melodies to analyze music, and we show some early results. We finish with a discussion and conclusions.

2 Related Work

The idea that similarity in music is an important aspect of musical structure is not new. Any standard text on music theory will introduce the ideas of repetition at different time scales, and of common forms such as rondo or sonata allegro, and these forms will be described largely in terms of what and when elements of the music are repeated or transformed.

Finding repeated phrases or sequences for various purposes is the goal of numerous studies. David Cope has searched music for short patterns, or *signatures*, as a way to identify characteristics of compositions and composers.[5] Conklin and Anagnosto-

poulou describe a pattern-discovery program for music that uses a probabilistic approach to separate significant patterns from chance ones. [4] Stammen and Pennycook used dynamic programming to search for repeated contours in jazz improvisations [19], and Rolland and Ganascia describe work using dynamic programming to find patterns in musical scores.[17]

Simon and Sumner described music listening as pattern formation, and introduced the idea that music can be encoded in terms of operations and transformations.[18] They suggest that listeners construct encodings when they listen to music, and that the encoding chosen by a listener will tend toward the shortest encoding possible. This idea is related to data compression, and more recent studies have used techniques from data compression to encode and generate music.[8]

Michael Leyton has written about structure and form as generative; that is, structures are perceived as a series of generative transfers from one space to another. [9] This suggests that the “content” of music is not melody, harmony, and rhythm but the transfer (by Leyton’s *control* groups) of these elements (Leyton’s *fiber groups*) within a composition, forming relationships and therefore structure.

This work is an instance of music analysis by computer, of which there are many examples in the literature. Foote and Cooper [6] constructed a similarity matrix from midi and audio data, and Wakefield and Bartsch created a similar structure using spectral data.[1, 2] This has inspired some of our work.

3 Structure from Melody

A set of programs was designed to apply these ideas to actual music content. These programs, which we will collectively call SAM1, for “Structural Analysis of Music – version 1,” perform of a number of steps:

- Read or record digital audio,
- Extract a pitch contour,
- Compute a similarity matrix,
- Find clusters of similar sequences,
- Build an explanation of the music in terms of structural relationships.

These steps are described in the following subsections, using a particular recording as an illustration. Results with other recordings will be described later.

3.1 Acquire Digital Audio

An audio CD recording of John Coltrane’s “Naima” [3] was used as input to the program. The digital audio was copied directly from the CD. One of the two stereo channels contained the strongest saxophone signal, so it was extracted to a mono sound file and down-sampled to 22,050Hz for the next step.

3.2 Pitch Extraction and Segmentation

Pitch estimation was performed using an autocorrelation technique on overlapping windows. Autocorrelation [14, 15] computes the correlation between the signal and a copy of the signal shifted by different amounts of time. When the shift is equal to a multiple of fundamental periods, the correlation will be high. In theory, one simply finds the first peak in the autocorrelation. In practice, there are small peaks corresponding to strong partials above the fundamental, and there may be higher peaks at multiples of the fundamental due to noise or subharmonics. The pitch estimation algorithm employs several simple heuristics to find the peak that corresponds to the fundamental.

In some cases there is no clear peak or fundamental, or the amplitude is low indicating there may be no signal present. Pitch estimates at these places are ignored. We also attempted to identify note boundaries using amplitude information, but found this to be unreliable, even in “Naima” where most notes are clearly articulated with clear attacks. Spectral techniques might also be used for note onset detection [16], but this could be a problem in a polyphonic context. We settled on using pitch estimates to segment the signal into notes.

After pitches have been estimated, the program labels regions of relatively stable pitch as notes. The program uses median filters to reject outliers from the sequence of pitch estimates and then searches for intervals where pitch varies within a limited range of 70 cents (0.7 semitones). The overall pitch of the note is based on the median value of up to 1 second of pitch estimates. This is more reliable than estimating pitch based on the very beginning of the note where pitch is less stable. Note durations are represented in seconds, and no attempt is made to identify beats or transcribe rhythm.

3.3 Similarity Matrix

After obtaining a sequence of notes, we need to find melodic fragments that repeat within the composition. Imagine comparing the melody starting on the first note to the melody starting on the second note, then the third note, etc., until the initial melody is compared to every other location in the piece. Then, the procedure is repeated starting with the second note and comparing that to every other location, etc. This approach gives rise to a 2-dimensional matrix we call the *similarity matrix*.

The similarity matrix contains elements s_{ij} , representing the length (in seconds) of a melodic sequence starting at note i that matches a melodic sequence starting at note j . Matches are determined by a relatively simplistic algorithm that works as follows: Starting at positions i and j , look for a match between note i and note j . If they match in both pitch and approximate duration, advance to the next notes and look for a match. If not, consider different rules to construct a match:

- Consolidate notes i and $i+1$ if they are near in pitch and duration to match note j ,
- Consolidate notes j and $j+1$ to match note i ,
- Consolidate notes i and $i+1$ and match to the consolidation of notes j and $j+1$, or
- Omit i and j if they are short and match $i+1$ to $j+1$.

A “greedy” algorithm without backtracking is used. If there is no match at all, the length of the match is reported as zero. In general, these rules are intended to identify

similar melodic contours. Consolidation [11] helps to match transcriptions with spurious note onsets. The transcription also tends to skip short notes, so we allow matches to ignore short durations. Other melodic similarity metrics could certainly be substituted for this one.

The diagonal of the matrix is uninteresting because we already know a melodic sequence is similar to itself, so the diagonal is filled with zeros. Because similarity is symmetric, only half of the matrix must be computed. When similar sequences are found, the length of one is stored in the upper half and the length of the other is stored in the lower half of the matrix.

3.4 Find Clusters of Similar Phrases

The similarity matrix identifies similar phrases, but there is no explicit indication that a phrase occurs more than once. The purpose of this next step is to form clusters from pairs of similar phrases.

Unfortunately, similarity is not transitive. If phrase A is similar to phrase B, and phrase B is similar to phrase C, it is not necessarily the case that phrase A is similar to C. This is because thresholds are used in comparing pitches and durations.

To form clusters, we scan across rows of the similarity matrix (or equivalently, down columns). If there are, say, two non-zero elements in row r in columns m and n , and if these all have approximately the same values (durations), it means that there are similar melodic sequences beginning at locations r , m , and n . These three locations and their durations form a cluster.

When a cluster is identified, it implies other similarities. E.g. we expect to find a similarity between m and n , so when m and n are added to a cluster, zeros are written to locations (m,n) and (n,m) . The cluster implies other similarities as well. For example, if r and n are similar and span more than one note each, we expect to find a similarity between $r+1$ and $n+1$. These implied similarities are also zeroed in the matrix to avoid further consideration of these elements.

The result of this step is a list of clusters consisting of sets of time intervals in the overall piece. Within each cluster, all time intervals refer to similar melodic sequences at different temporal locations.

3.5 Building an “Explanation”

The final step of the program is to describe the music in terms of structural relationships, the overall goal of the program. Intuitively, we view this as an “explanation” of the music. For each moment of music, we would like to have an explanation of the form “this is an instance of *phrase-name*,” where *phrase-name* is an arbitrary name that denotes one of the clusters identified in the previous step.

To build such an explanation, we proceed from first note to last. At each point, if the note has not already been “explained,” search the clusters to find an interval that includes the note. Name the cluster, e.g. using the next name in the sequence “A,” “B,” “C,” etc. Then, for each unexplained note included in an interval in the cluster, mark the note with the cluster name.

3.6 Results from SAM1

Figure 1 illustrates intermediate and final results of the program applied to “Naima.” The input waveform is shown at the top. Next, a piano-roll transcription is shown. Because the recording contains bass, drums, and piano along with the saxophone, and because the note identification and segmentation requires many fairly stable pitch estimates, some of the short notes of the performance (which are clearly audible) are not captured in this transcription. Also, the piano solo is missed completely except for a few spurious notes. This is because the piano notes are often in chords and because the notes decay quickly and therefore do not meet the stability criterion.

Below the note are shown clusters. A thin line connects the time intervals of each cluster, and the time intervals are shown as thick lines. Finally, the bottom of the figure shows the final result. Here, we see that “Naima” begins with an AABA form, where the B part is actually $b_1b_1b_2$, so the flattened structure should be $AAb_1b_1b_2A$, which is what we see in Figure 1. Following that, the recording contains a piano solo that was mostly missed by the transcription. After the piano solo, the saxophone enters, not by repeating the entire AABA form, but on the “bridge,” the “B” part. This aspect of the structure can be seen clearly in the figure. Following the final $b_1b_1b_2A$, the last 2 measures are repeated 3 times (as shown in the figure) and followed by a rising scale of half notes that does not appear elsewhere in the performance.

Overall, the analysis is excellent, and it was an encouraging surprise to see such a clear form emerge solely from audio input without the benefit of polyphonic transcription, beat tracking, or other analyses.

To be fair, “Naima” was used to debug and refine the program, so there is some danger that the program is actually tuned to fit the data at hand. To further evaluate the program, we applied it to monophonic recordings of some simple melodies. These are the familiar Christmas tune “We Three Kings,” and a standard jazz tune “Freddie the Freeloader” composed by Miles Davis. An amateur violinist performed the first of these, and an experienced jazz trumpet player played the second (without improvisation.)

The program was applied to these recordings without any further tuning or refinement. The results of these experiments are also quite good, as shown in Figures 2 and 3. One bit of explanation is in order for Figure 3. “Freddie the Freeloader,” basically a 12-bar blues, has a first and second ending for the last two bars, so it was performed with the repeat for this experiment. The program found the similarity between the first and second halves of the performance, so the overall form might have been simply AA. However, the program also found and reported substructure within each A that corresponds to what we would ordinarily describe as the “real” structure of the piece. (It was an unexpected “feature” that the program managed to report both explanations.) In the future, it seems essential to construct hierarchical explanations to avoid similar problems.

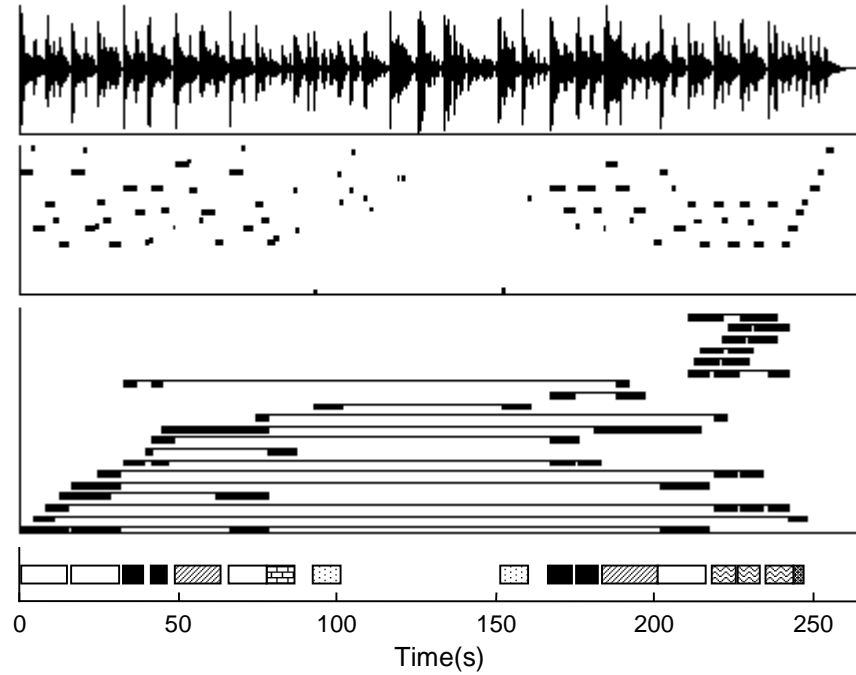


Fig. 1. Analysis of “Naima.” From top to bottom: input audio waveform, transcription in “piano roll” format, clusters of similar melodic material, analysis or “explanation” of the music. In the display of clusters, the vertical axis has no meaning except to separate the clusters. Each cluster is a thin horizontal line joining the elements of the cluster indicated by heavy lines.

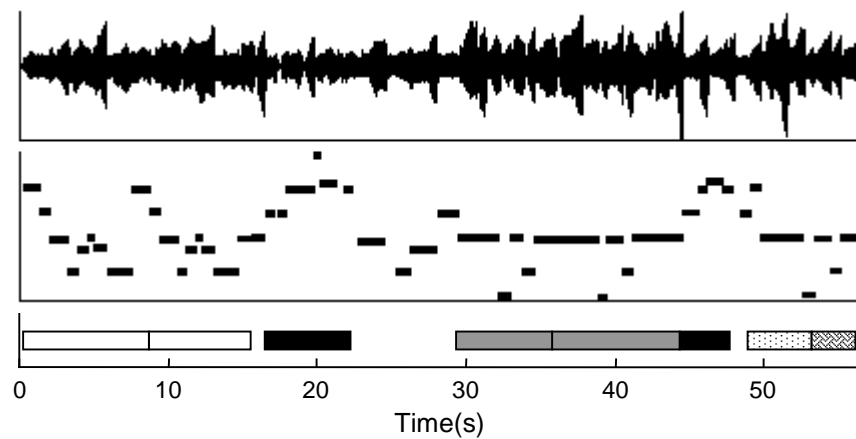


Fig. 2. Analysis of “We Three Kings.” The structure is AABCDDED, but the program found some similarity between B and E. The final D section was not matched with the other D sections.

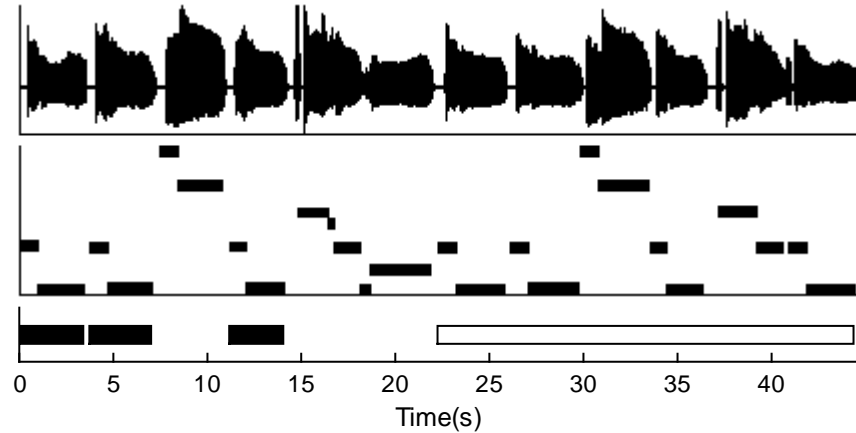


Fig. 3. Analysis of “Freddie the Freeloader,” showing three occurrences of the opening figure. The program also detected the similarity between the first and second halves of the piece, but the non-hierarchical descriptions and graphics output do not capture this aspect of the structure.

4 Analysis of Polyphonic Music

In terms of transcription, music is either monophonic or polyphonic. Pitch estimation from monophonic sources is relatively simple, while polyphonic transcription is characterized by high error rates, limited success, and continuing active research. While “Naima” is certainly polyphonic music in the sense of there being a quartet of instruments, we were able to treat it as a monophonic signal dominated by the tenor saxophone lines. This approach cannot work for many types of music.

In principle, there is no need to use melodic contour or monophonic pitch estimates for music analysis. Any local attribute or set of attributes could be used to judge similarity. There are many alternatives to using monophonic transcription, although research is still needed to test and evaluate different approaches. We have begun to explore the use of spectra, and in particular, the use of a spectrum-based vector called “chroma.” [20]

The hope is that we can identify similar sections of music by comparing short-term spectral content. With spectra and particularly with polyphonic material, it is unlikely that we will be able to separate the music into segments in the way that we obtained notes from pitch estimates. Instead, we take spectral frames of equal length as the units of comparison, inspired by earlier work on melodic matching.[10]

4.1 Chroma

Rather than using spectra directly, we use *chroma*, which are similar to spectra, but reduced in dimensionality to a vector of length 12. The elements of the vector represent the energy associated with each of the 12 pitch classes in an equal-tempered chromatic scale. Chroma are computed in a straightforward manner from spectra. Our

choice of chroma was influenced by earlier and successful work in which chroma were used to locate recurring passages of music.

Recall in the earlier work that a similarity matrix was computed to locate similar sequences of notes. With chroma, we also construct a similarity matrix to locate similar sequences of chroma. We use chroma frames of 0.25s duration. To compute the distance between two chroma, we first normalize the chroma vector to have a mean value of zero and a standard deviation of 1. Then we compute the Euclidean distance between the two vectors.

Our goal is to find whether there is a match beginning with each pair of frames i and j . Dynamic programming, e.g. time warping algorithms, could be used to find an optimal sequence alignment starting at each pair. Since dynamic programming costs $O(n^2)$ and there are n^2 pairs, the brute force search costs $O(n^4)$, although this can be optimized to $O(n^3)$ by computing an entire row of pairs at once. A song will often be divided into hundreds of chroma frames, and the $O(n^3)$ cost is prohibitive. Instead, we use a greedy algorithm to increase the computational speed, and we compute a little less than half of the similarity matrix (the diagonal is ignored), because the matrix is symmetric. (Note that the name “similarity” may be confusing because here, increasing values represent greater distance and less similarity.)

The goal of the algorithm is to find pairs of intervals $(S_{i,j}, S_{m,n})$ of frames such that $S_{i,j}$ is “similar” to $S_{m,n}$. Interval similarity is based on the distance of chroma at two locations. Call chroma distance as described earlier $D(i, j)$. Define interval $S_{i,j}$ to be a range of frame numbers from i to j , inclusive. The similarity of two intervals is the quotient of a distance function $d(S_{i,j}, S_{m,n})$ and a length function $l(S_{i,j}, S_{m,n})$. The distance function is defined in terms of a path from (i, m) to (j, n) . A path consists of a sequence of adjacent (diagonal, horizontal, or vertical) cells in the matrix. Given a path P , we can define distance d_P as the sum of the chroma distance along path P :

$$d_P = \sum_{(i,j) \in P} D(i, j)$$

The distance function d is the minimum over all paths:

$$d(S_{i,j}, S_{m,n}) = \operatorname{argmin}_P (d_P)$$

The length function is defined as the total path length, where a diagonal step has length 1 and a horizontal or vertical step has length $\sqrt{2}/2$. This is equivalent to:

$$l(S_{i,j}, S_{m,n}) = \min(j-i, n-m) + (\sqrt{2}/2)(\max(j-i, n-m) - \min(j-i, n-m))$$

The distance M between two intervals is:

$$M(S_{i,j}, S_{m,n}) = d(S_{i,j}, S_{m,n}) / l(S_{i,j}, S_{m,n})$$

In general, we want to find interval pairs such that M falls below some threshold. In practice, there are many overlapping intervals that satisfy any reasonable threshold value, so we really only want to find large intervals, ignoring intervals that are either very similar or contained within larger similar pairs.

Thus, the algorithm we use is a heuristic algorithm that searches for the longest paths starting as promising starting points where $S(i, j)$ is already below a threshold. The algorithm computes several values for each element of a matrix: the distance function d , the length l , and starting point p . The matrix cells are scanned along diagonals of constant $i+j$, moving from upper left to lower right (increasing values of $i+j$). At each point, we compute d/l based on the cell to the left $(i, j-1)$, above $(i-1, j)$, and diagonally left and above $(i-1, j-1)$. The cell giving the minimum distance is used to compute the new values of d , l , and p for location i, j . Cells are only computed if the value of d/l is below threshold.

This calculation identifies whole regions of ending points for a set of good starting points for matching intervals. To determine good ending points, this search process is also executed in reverse (decreasing $i+j$) and with paths going in the opposite direction, computing a second matrix. The starting points of this reverse matrix become the ending points of the forward matrix. It is easy to find the ending point corresponding to a starting point using the p values of the reverse matrix.

Figure 4 shows pairs obtained from an analysis of “Minuet in G.”

4.2 Clusters

After computing similar segments of music in pairs, the next step is to construct clusters from the pairs. The clustering algorithm is different from the note-based clustering described earlier. When matching notes, similar sequences match at discrete locations corresponding to note onsets, but with chroma frames, matches do not occur at clearly defined starting and ending points. Thus, the technique of scanning along a row to find all members of a cluster might not work unless we search at least several rows for near matches. The following clustering algorithm deals with near-matches in a different way.

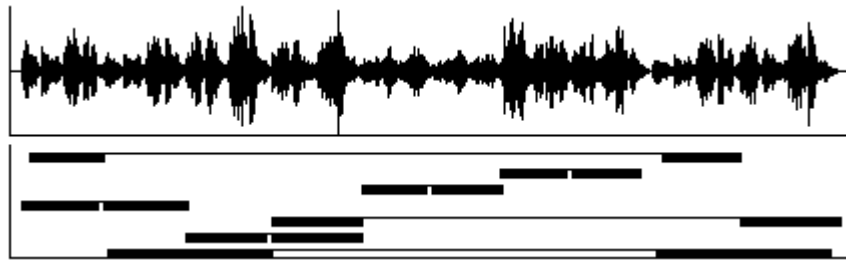


Fig. 4. Finding pairs of similar material in Beethoven’s Minuet in G. The horizontal thin lines connect elements of a pair.

To form clusters from pairs, iteratively remove clusters from the set of pairs as follows: On each iteration, remove a pair from the set of pairs (this gives the first two elements of a new cluster) and search the remaining pairs for matches. Two pairs match if one element of the first pair is approximately the same as either element of the second (recall that elements here are simply time intervals). If a match is found, add the unmatched element of the pair to the cluster and remove the pair from the set of pairs. Continue outputting new clusters until the set of pairs is empty.

This algorithm has one problem. Sometimes, there are pairs that differ in size and therefore do not match. Consider the pairs in Figure 5. Segment A matches B, which is identical to the first part of C. Since C matches D, the first part of D must also match A. It is clear that there are three segments that match A, but since element C is much larger than B, the pairs do not match. The algorithm is extended by handling this as an alternate case. Since C is larger than B but contains it, we compute which portion of C matches B, then add the corresponding portion of D to the cluster.

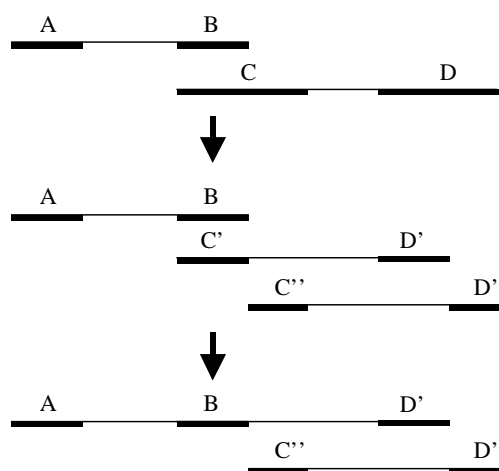


Fig. 5. When forming clusters, two pairs may share overlapping elements (B and C) that do not match because of length. To form a cluster of 3 elements that all match A, we split C to match B (which in turn matches A) and split D in proportion to C.

Hierarchical clustering techniques might be useful in dealing with approximate matches and provide a more theoretically sound basis for clustering melodic segments. An interesting twist here is that data can be modified as in Figure 5 to form better clusters. One might also get better results using an iterative approach where matches suggested by clustering provide evidence to be used in the signal analysis and similarity phases.

4.3 Results Using Chroma

Once clusters are computed, we can perform the explanation step as before. Figure 6 illustrates an explanation (or structural analysis) of Beethoven's Minuet in G, analyzed from an audio recording of an acoustic piano performance. The analysis clearly illustrates the structure of the piece.

Other pieces, with more variety and less exact repetition, have not produced such definitive results. For example, a pop song with a great deal of repetition is shown in Figure 7. It is encouraging that much of the repetitive structure emerged, but those repetitions are rather exact. Where the repetition includes some variation or improvisation, our system missed the similarity because even the chroma representation is quite different in each variation. This is due to changes in timbre, orchestration, and vocal improvisation. Better representations will be needed to recognize these variations in polyphonic music. Further work is also needed to deal with the fact that similar sections do not have definitive starting and ending points. When a segment in one cluster overlaps a segment in another cluster, it is not clear whether the two segments represent the same phrase of music or whether one is a sub-phrase of the other. This has implications for the structure of the music.

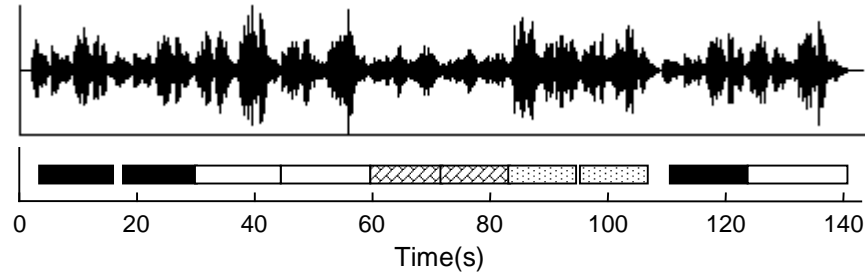


Fig. 6. Analysis of Beethoven's Minuet in G is shown below the input waveform. Notice the clear analysis at the bottom: AABBCDDAB.

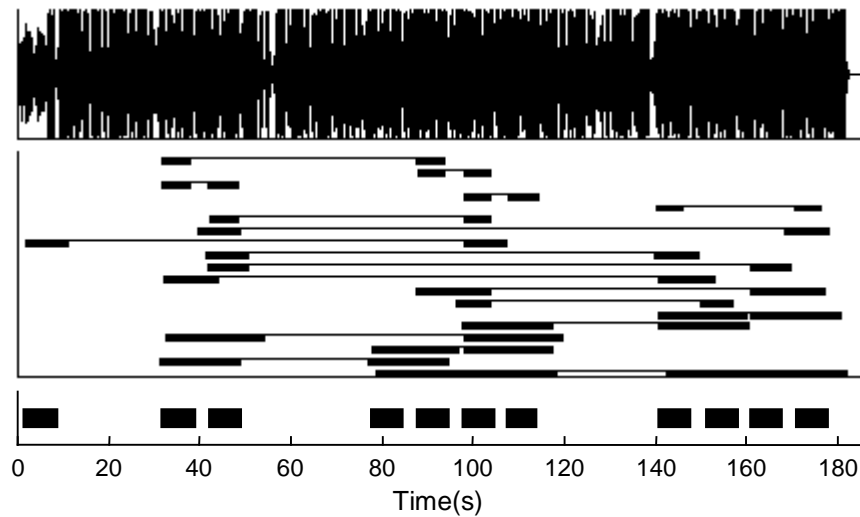


Fig. 7. Analysis of a pop song. There is frequent repetition of a short pattern within the song as shown in the analysis. Note the ambiguity of starting and ending times in the pairs (middle of figure). Our clustering and analysis failed to find much structure other than many repetitions of one motive. For example, the recording has a melody from about 10 to 20s and a variation at approximately 50 to 70s.

5 Discussion

It seems quite clear that an important aspect of music listening is to identify relationships and structure within the music. The most basic, salient, and common relationship is the repetition of a phrase. Our goal is to build automated “music listeners” that identify musical structure by finding relationships. There are several motivations behind this work. First, building machines that mimic human behavior is intrinsically interesting. As music understanding seems to be a uniquely human

experience, automating aspects of this skill is particularly fascinating. Second, we can learn more about music and about music perception by studying listening models. Our models are not intended as faithful perceptual models, but they can still tell us something about structure, coding, redundancy, and information content in musical signals. For example, this work sheds some light on whether general musical knowledge is required to understand musical structure, or is the structure implied by the music itself? Third, our original motivation was the problem of music meta-data creation. Audio databases have very little structure and there is very little that can be searched unless the audio is augmented with descriptions of various kinds. Text fields naming composers, performers, titles, instrumentation, etc. are a standard way to make audio searchable, but there are many “musical” aspects of music that cannot easily be described by text. A program that “listens” to music seems much more likely to find attributes that are of interest to humans searching a database. Attributes including common themes, salient or surprising chord progressions, and overall form might be derived automatically. These attributes should also be useful to obtain or validate other information about tempo, genre, orchestration, etc.

The significance of our work is that we have demonstrated through examples how an automated process can find pattern and structure in music. We believe that a primary activity in music listening is the identification of pattern and structure, and that this program therefore exhibits a primitive form of music listening. The input to the program is audio, so there is no “hidden” information provided through symbolic music notation, pre-segmented MIDI representation, or even monophonic audio where sources are cleanly separated.

The quality of our analyses is variable. We suspect that is true for human listeners as well, but we have no metric for difficulty or experimental data on human listeners. Furthermore, the “correct” analysis is often ambiguous. There is certainly room for much experimentation in this area. Given the inherently analog and “noisy” nature of audio input, we are pleasantly surprised that the analysis of at least some music corresponds so clearly with our intuition about what is correct.

Although some form of quantitative evaluation of our approach might be interesting, our goal thus far has been to identify techniques that show promise and to demonstrate that some analysis procedure can actually succeed using audio input. As indicated in Section 3.6, we performed a successful analysis of “Naima” after tuning various parameters and refining our algorithms. We also successfully analyzed two other pieces with no further tuning to demonstrate that the algorithms are robust enough to handle different pieces and instruments. On the other hand, we have run into difficulties with polyphonic transcription data, and it is clear that our techniques are not general enough for a wide spectrum of musical styles. Rather than evaluate this early stage of research, we believe it is more appropriate to experiment with different techniques and compare them.

In the future, we plan to look at other techniques for analysis. Polyphonic transcription is unable to *reliably* transcribe typical audio recordings, but the results seem good enough to determine if passages are similar or dissimilar, and the discrete nature of the output might be easier to work with than continuous spectra. Intermediate levels of transcription, such as Goto’s [7] work on bass and melody extraction could also be used.

Another direction for future work is a better theoretical model of the problem. How do we know when an “explanation” is a good one, and how do we evaluate

ambiguous choices? Ideally, we would like a probabilistic framework in which evidence for or against different explanations could be integrated according to a sound (no pun intended) theory.

So far, we have only looked for repetition. In many of our examples, there are instances of transposition and other relationships. These are also important to music understanding and should be identified. It appears that these relationships are often less redundant than outright repetition. After all, transposition can occur at different intervals, and transposition may be tonal or chromatic. Also, transpositions often involve just two or three notes in sequence, i.e. one or two intervals. A good model will be needed to decide when a descending second is a “motive” and when it is just a descending second. The distinction often depends upon rhythmic and harmonic context, which means more sophisticated analysis is required to detect transpositional structure. It would also be interesting to look for rhythmic motives, as in the work of Mont-Reynaud. [12] This is an area for further research.

Conclusions

We have shown how automated systems can listen to music in audio form and determine structure by finding repeated patterns. This work involves the conversion of audio into an intermediate form that allows comparisons. We have used monophonic pitch estimation and transcription as well as spectral information as an intermediate representation. With transcription, the intermediate form is a sequence of notes, and we developed a note-based pattern discovery algorithm to find similar subsequences. With spectra, the intermediate form is a sequence of frames of equal duration, and the pattern search uses a “ridge following” algorithm to find correspondences in the music starting at two different points in time. In both cases, pairs of music segments found to be similar are formed into clusters. These clusters represent patterns that recur in the music. The music is then “explained” by identifying how the music can be constructed from a sequence of these patterns. In spite of the fact that the music is originally in audio form, complete with noise, reverberation, drums, and other complications, our algorithms are able to do a very good job of extracting structure in many cases. Familiar structures such as AABA and AABBC... are found in our examples. The structural description also reports the location of the patterns, so it is a simple matter for a user to locate all occurrences of the “A” pattern, for example.

Our results so far are quite encouraging. Many inputs are analyzed almost without any error. As would be expected, there are plenty of examples where analysis is not so simple. In the future, we hope to develop better theoretical models, refine our analysis techniques, and identify other relationships such as rhythmic patterns and transpositions.

Acknowledgements. This work is supported in part the National Science Foundation, award number #0085945.

References

- [1] Bartsch, M. and Wakefield, G.H., To Catch a Chorus: Using Chroma-Based Representations For Audio Thumbnailing. in *Proceedings of the Workshop on Applications of Signal Processing to Audio and Acoustics*, (2001), IEEE.
- [2] Birmingham, W.P., Dannenberg, R.B., Wakefield, G.H., Bartsch, M., Bykowski, D., Mazzoni, D., Meek, C., Mellody, M. and Rand, W., MUSART: Music Retrieval Via Aural Queries. in *International Symposium on Music Information Retrieval*, (Bloomington, Indiana, 2001), 73-81.
- [3] Coltrane, J. *Naima Giant Steps*, Atlantic Records, 1960.
- [4] Conklin, D. and Anagnostopoulou, C., Representation and Discovery of Multiple Viewpoint Patterns. in *Proceedings of the 2001 International Computer Music Conference*, (2001), International Computer Music Association, 479-485.
- [5] Cope, D. *Experiments in Musical Intelligence*. A-R Editions, Inc., Madison, Wisconsin, 1996.
- [6] Foote, J. and Cooper, M., Visualizing Musical Structure and Rhythm via Self-Similarity. in *Proceedings of the 2001 International Computer Music Conference*, (Havana, Cuba, 2001), International Computer Music Association, 419-422.
- [7] Goto, M., A Predominant-F0 Estimation Method for CD Recordings: MAP Estimation using EM Algorithm for Adaptive Tone Models. in *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing*, (2001), IEEE, V-3365-3368.
- [8] Lartillot, O., Dubnov, S., Assayag, G. and Bejerano, G., Automatic Modeling of Musical Style. in *Proceedings of the 2001 International Computer Music Conference*, (2001), International Computer Music Association, 447-454.
- [9] Leyton, M. *A Generative Theory of Shape*. Springer, Berlin, 2001.
- [10] Mazzoni, D. and Dannenberg, R.B., Melody Matching Directly From Audio. in *2nd Annual International Symposium on Music Information Retrieval*, (2001), Indiana University, 17-18.
- [11] Mongeau, M. and Sankoff, D. Comparison of Musical Sequences. in Hewlett, W. and Selfridge-Field, E. eds. *Melodic Similarity Concepts, Procedures, and Applications*, MIT Press, Cambridge, 1990.
- [12] Mont-Reynaud, B. and Goldstein, M., On Finding Rhythmic Patterns in Musical Lines. in *Proceedings of the International Computer Music Conference 1985*, (Vancouver, 1985), International Computer Music Association, 391-397.
- [13] Narmour, E. Music Expectation by Cognitive Rule-Mapping. *Music Perception*, 17 (3). 329-398.
- [14] Rabiner, L. On the use of autocorrelation analysis for pitch detection. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-25 (1). 24-33.
- [15] Roads, C. Autocorrelation Pitch Detection. in *The Computer Music Tutorial*, MIT Press, 1996, 509-511.
- [16] Rodet, X. and Jaillet, F., Detection and Modeling of Fast Attack Transients. in *Proceedings of the 2001 International Computer Music Conference*, (2001), International Computer Music Association, 30-33.
- [17] Rolland, P.-Y. and Ganascia, J.-G. Musical pattern extraction and similarity assessment. in Miranda, E. ed. *Readings in Music and Artificial Intelligence*, Harwood Academic Publishers, 2000, 115-144.
- [18] Simon, H.A. and Sumner, R.K. Pattern in Music. in Kleinmuntz, B. ed. *Formal Representation of Human Judgment*, Wiley, New York, 1968.
- [19] Stammen, D. and Pennycook, B., Real-Time Recognition of Melodic Fragments Using the Dynamic Timewarp Algorithm. in *Proceedings of the 1993 International Computer Music Conference*, (Tokyo, 1993), International Computer Music Association, 232-235.
- [20] Wakefield, G.H., Mathematical Representation of Joint Time-Chroma Distributions. in *International Symposium on Optical Science, Engineering, and Instrumentation, SPIE'99*, (Denver, 1999).

Real Time Tracking and Visualisation of Musical Expression

Simon Dixon, Werner Goebel, and Gerhard Widmer

Austrian Research Institute for Artificial Intelligence, Schottengasse 3, A-1010 Vienna, Austria.
{simon,wernerg,gerhard}@ai.univie.ac.at

Abstract. Skilled musicians are able to shape a given piece of music (by continuously modulating aspects like tempo, loudness, etc.) to communicate high level information such as musical structure and emotion. This activity is commonly referred to as expressive music performance. The present paper presents another step towards the automatic high-level analysis of this elusive phenomenon with AI methods. A system is presented that is able to measure tempo and dynamics of a musical performance and to track their development over time. The system accepts raw audio input, tracks tempo and dynamics changes in real time, and displays the development of these expressive parameters in an intuitive and aesthetically appealing graphical format which provides insight into the expressive patterns applied by skilled artists. The paper describes the tempo tracking algorithm (based on a new clustering method) in detail, and then presents an application of the system to the analysis of performances by different pianists.

1 Introduction

An expert musical performer is able to shape a given piece of music to communicate high level information such as musical structure and emotion. That is, the artist goes beyond what is prescribed in the written score and modifies, gradually or abruptly, the tempo or loudness or other parameters at certain places in the piece in order to achieve certain musical and emotional effects. This activity is commonly referred to as *expressive music performance*. Expressive performance is an element of central importance in art music and especially in classical music, where the performing artists have (or take) a lot of freedom in expressing their interpretation of the music and their individuality. At the same time, expressive performance is still a poorly understood phenomenon, both from a musical and a cognitive perspective. No formal models exist that would explain, or at least quantify and characterise, aspects of commonalities and differences in performance style.

This paper presents a step towards the automatic high-level analysis of this elusive phenomenon with Artificial Intelligence methods. We restrict our attention to two of the most important expressive dimensions: fluctuations in *tempo* and *loudness (dynamics)*. A system is presented that is able to measure tempo and dynamics of a musical performance and to track their development over time. The system accepts raw audio input (e.g., from a microphone), tracks tempo and dynamics changes in real time, and displays the development of these expressive parameters in an intuitive and aesthetically appealing

graphical format which provides insight into the expressive patterns applied by skilled artists.

Measuring and tracking *dynamics* is rather straightforward. The (perceived) loudness of the music can be derived from the audio signal by applying well-known signal processing techniques and psychoacoustic principles. The difficult part is inferring the basic *tempo*, and tracking changes in tempo in real time. The main problems are detecting the onsets of notes in the raw audio signal (event detection), inferring the basic rate of beats or tempo and the most plausible metrical level (tempo induction), and the real time adaptation of the tempo hypotheses in response to newly incoming information (tempo tracking).

The main technical contribution of this paper is a real time algorithm that finds the tempo of a musical performance, keeping track of multiple hypotheses, rating and updating each of the hypotheses dynamically, and allowing the user to interactively switch between hypotheses (e.g., when the system has obviously chosen a wrong metrical level). At the heart of the tempo induction and tracking system is a fast on-line clustering algorithm for time intervals.

In the following, we describe the tempo tracking and clustering algorithm in detail, and then present an application of the algorithms in a real time visualisation system for expressive music performance. An example visualisation of two pianists playing the same piece demonstrates what kind of direct insight into expressive performance can be facilitated by this kind of system.

2 Real Time Tempo Tracking

Most music has as its rhythmic basis a series of pulses, spaced approximately equally in time, from which the timing of all musical events can be measured. This phenomenon is called the *beat*, and the individual pulses are also called beats. The rate at which beats occur defines the *tempo*, a value which varies over time. Sometimes a multiple or divisor of the tempo is perceived as an alternative tempo; these different rates are called *metrical levels*.

The task of a tempo induction and tracking system at any moment during its operation is to infer, from the observed inter-note time intervals, possible beat rates and select the one that most likely represents the perceived tempo of the piece. This is performed by a clustering algorithm which groups similar time intervals between note onsets, forming clusters which correspond to musical time units, such as half notes, quarter notes and dotted quarter notes. In a mechanical performance, these time intervals would be precisely integer or simple integer fraction multiples of the time between two consecutive beats. But in expressive performance, the categories are blurred and change over time, so the clustering algorithm must be robust to noise and able to adapt dynamically to drift in the cluster centres.

The architecture of the tempo tracker is shown in figure 1. The input signal is preprocessed in several stages to detect the onsets of musical notes (events), and this information is used by the multiple tempo tracking subsystem to create a set of tempo hypotheses which are updated dynamically as further input data arrives. The highest-ranking tempo hypothesis is given as output, but the ranking can be overridden by the user selecting

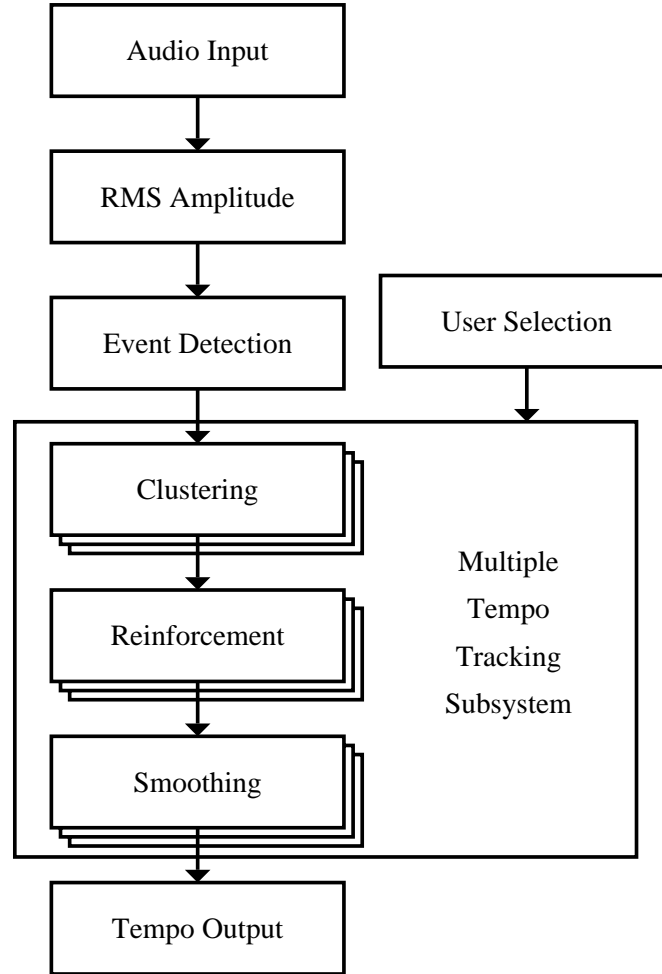


Fig. 1. Architecture of tempo tracking system

a different metrical level. In the remainder of this section, we describe the successive stages of processing in detail.

2.1 Audio Processing

The audio input is read from the soundcard or from a file, in linear PCM format. If the input has more than one channel, a single channel signal is created by averaging all channels. The resulting signal is denoted $x[n]$ and its sampling rate R .

The audio data is processed in blocks by a smoothing filter which calculates the RMS amplitude $A[k]$, where k is the integer time index expressed as a multiple of the block

size. A block size of 10ms ($b = 0.01R$ samples) is used for input and processing of the signal, and this determines the time resolution of the system. The amplitude values are calculated by smoothing across a number ($h = 4$) of blocks and passed to the event detection module.

$$A[k] = \left(\frac{1}{hb} \sum_{i=k-b}^{(k+h)b-1} x[i]^2 \right)^{\frac{1}{2}} \quad (1)$$

The event detection module finds the slope $S[k]$ of the smoothed amplitude using an m -point linear regression. It then calculates the set $P[k]$ of local peaks of $S[k]$ which are above given thresholds T_1 of amplitude and T_2 of slope, where a local peak is defined to be a point which is maximal among the l points either side of it.

$$S[k] = \frac{4 \sum_{i=k-m+1}^k \frac{ibA[i]}{R} - \left(\sum_{i=k-m+1}^k A[i] \right) \left(\sum_{i=k-m+1}^k \frac{ib}{R} \right)}{4 \left(\sum_{i=k-m+1}^k A[i]^2 \right) - \left(\sum_{i=k-m+1}^k A[i] \right)^2} \quad (2)$$

$$P[k] = \left\{ k - m + 1 \mid S[k] = \max_{i=k-l}^{k+l} (S[i]) \text{ and } A[k] > T_1 \text{ and } S[k] > T_2 \right\} \quad (3)$$

These local peaks are taken to be note onset times, which are the main input to the multiple tempo tracking module, the most complex part of the system. Although a relatively simple time domain algorithm is used for event detection, it has been shown previously [3] that the accuracy is sufficient for successful extraction of tempo.

2.2 Multiple Tempo Tracking Subsystem

Clustering. The tempo induction and tracking system calculates the time intervals between pairs of recent events (*inter-onset intervals*, or *IOIs*) and uses a clustering algorithm (figure 2) to find significant clusters of IOIs, which are assumed to represent musical units. These clusters form the bases of the tempo hypotheses generated by the system. The clustering algorithm maintains a limited memory ($M = 8$ seconds) of onset times in the set $P'[k]$, and begins processing by calculating all IOIs between pairs of onsets in its memory, weighting the intervals by the geometric mean of the amplitudes of the onsets, and summing across equally spaced onset pairs, to give the sums $I[k, i]$ for each IOI i calculated at time index (block number) k , as shown in figure 3(a).

$$P'[k] = \{j \in P[k] \mid k - j \leq \frac{MR}{b}\} \quad (4)$$

$$I[k, i] = \sum_{\substack{i_1, i_2 \in P'[k] \\ \text{with } i = i_1 - i_2}} \sqrt{A[i_1 + m - 1]A[i_2 + m - 1]} \quad (5)$$

```

For each new onset
  For times  $t$  from 100ms to 2500ms in 10ms steps
    Find pairs of onsets which are  $t$  apart
    Sum the mean amplitude of these onset pairs
  Loop until all time points are used
  For times  $t$  from 100ms to 2500ms in 10ms steps
    Calculate window size  $s$  as function of  $t$  (see (6))
    Find average amplitude of IOIs in window  $[t, t + s]$ 
    Store  $t$  which gives maximum average amplitude
  Create a cluster containing the stored maximum window
  Mark the IOIs in the cluster as used
For each cluster
  Find related clusters (multiples or divisors)
  Combine related clusters using weighted average
  Match combined clusters to tempo hypotheses and update

```

Fig. 2. Algorithm for clustering of inter-onset intervals

At each time k , the inter-onset intervals $I[k, j]$, limited to the range 0.1s to 2.5s ($\frac{0.1R}{b} \leq j \leq \frac{2.5R}{b}$) are clustered using an iterative best-first algorithm given in figure 2, which sequentially finds the clusters with the greatest average amplitude, without reusing the data for more than one cluster.

The size $s[i]$ of the windows used in clustering is calculated as a function of the minimum IOI i in the cluster:

$$s[i] = \lfloor \frac{i}{30} + 8 \rfloor \quad (6)$$

The best clusters are given by maxima in the average weight $w[k, i]$, as shown in figure 3(b). The starting indices of the best clusters at time k are denoted $a[k, 1], a[k, 2], \dots$

$$w[k, i] = \frac{\sum_{j=i}^{i+s[i]} I[k, j]}{s[i] + 1} \quad (7)$$

$$a[k, 1] = \operatorname{argmax}_i w[k, i] \quad (8)$$

The centroid $t[k, 1]$ of the cluster at $a[k, i]$ is calculated as the weighted average of the component IOIs. The initial weight assigned to the cluster is denoted $v[k, 1]$.

$$t[k, 1] = \frac{b \sum_{j=a[k, 1]}^{a[k, 1] + s[a[k, 1]]} j I[k, j]}{R \sum_{j=a[k, 1]}^{a[k, 1] + s[a[k, 1]]} I[k, j]} \quad (9)$$

$$v[k, 1] = w[k, a[k, 1]] \quad (10)$$

The next best clusters $t[k, i]$ and their weights $v[k, i]$ for $i = 2, 3, \dots$ are calculated by setting to 0 the values of $I[k, j]$ which have been used in a previous cluster $t[k, j]$ (where $j < i$) and repeating the above calculations.

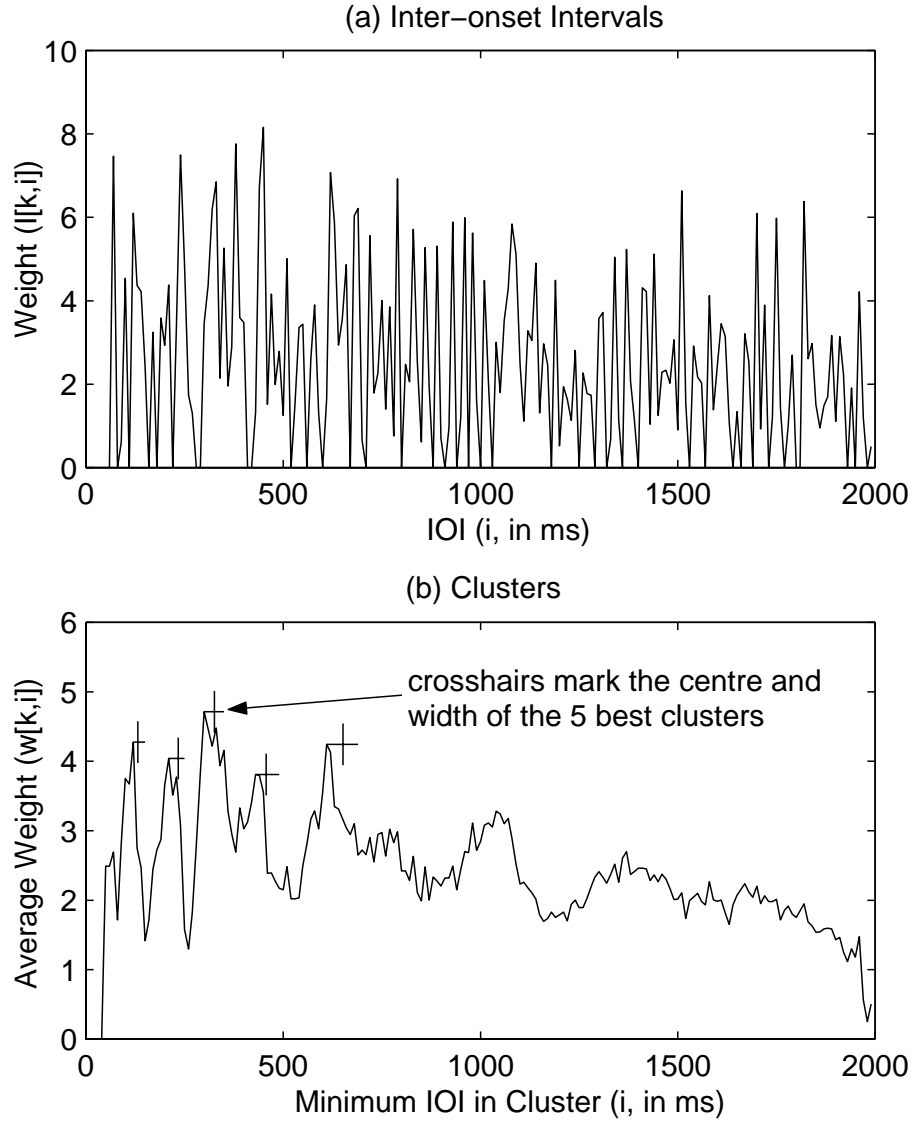


Fig. 3. (a) Example showing weighted inter-onset intervals $I[k, i]$ before clustering, for a fixed time k , and (b) Average weights $w[k, i]$ after clustering, showing the 5 best clusters as crosshairs centred at $(t[k, l], v[k, l])$ for $l = 1, 2, 3, 4, 5$, with leftmost point at $a[k, l]$ and horizontal length $s[a[k, l]]$.

Reinforcement. It is usually the case in traditional Western music that time intervals are approximately related by small integer ratios, and therefore the cluster times $t[k, i]$ are expected also to reflect this property. In other words, the cluster times are not independent;

they represent related musical units such as quarter notes and half notes. The tempo inducer exploits this property by recalculating the cluster times and weights based on the combined information given by sets of related clusters. Two clusters are said to be related if the ratio of their time intervals $\frac{t[k,i]}{t[k,j]}$ is close to an integer. More formally, the set of clusters $r[k, i]$ related to $t[k, i]$ is defined as follows.

$$f(x, y) = \begin{cases} y/x & x \leq y \\ x/y & x > y \end{cases} \quad (11)$$

$$d(x, y) = |f(x, y) - \text{round}(f(x, y))| \quad (12)$$

$$d'(x, y) = \frac{d(x, y)}{f(x, y)} \quad (13)$$

$$r[k, i] = \left\{ t[k, j] \mid d'(t[k, i], t[k, j]) < 0.1 \text{ and } 2 \leq \text{round}(f(t[k, i], t[k, j])) \leq 8 \right\} \quad (14)$$

The errors of individual cluster times are reduced by bringing the related times closer to the ideal integer ratios. To achieve this, the related clusters are scaled to the same metrical level and a weighted average of the scaled clusters is calculated. The weighting favours longer time intervals, which tend to have a lower relative error. Formally, the updated clusters $t'[k, i]$ (with weights $v'[k, i]$) are calculated as follows.

$$f'(x, y) = \begin{cases} y/x & x \leq y \\ 1 & x > y \end{cases} \quad (15)$$

$$t'[k, i] = \frac{\sum_{t[k, j] \in r[k, i]} t[k, j] v[k, j] / f'(t[k, i], t[k, j])^2}{\sum_{t[k, j] \in r[k, i]} (v[k, j] / f(t[k, i], t[k, j]))} \quad (16)$$

$$v'[k, i] = \sum_{t[k, j] \in r[k, i]} (v[k, j] / f(t[k, i], t[k, j])) \quad (17)$$

Smoothing. Tempo as a percept arises from the timing of many notes; local changes in timing do not unambiguously imply a tempo change. In order that the system is not disrupted by local timing irregularities, like the delay of a single note, the tempo tracker performs smoothing on the tempo hypotheses generated above. The tempo hypotheses $t'[k, i]$ at time step k , ranked by the corresponding weights $v'[k, i]$, are combined with historical values from the previous time step $k - 1$ to give updated hypotheses $t''[k, i]$. Each current hypothesis $t'[k, i]$ is matched to the nearest $t''[k - 1, j]$ (if a sufficiently near one exists), and updated according to a formula which causes old values to decay exponentially as they are replaced by new. If the decay factor is γ , the updated value is:

$$t''[k, i] = \gamma t''[k - 1, j] + (1 - \gamma) t'[k, i] \quad (18)$$

Although it would be possible to keep track of all hypotheses, it is sufficient for the system to keep track of the best 10 hypotheses; no change in behaviour is noted when more hypotheses are tracked. Sometimes hypotheses are found to be duplicating the same metrical level, in which case they are merged into a single hypothesis.

3 Application: A Worm with EARS

The above algorithm, together with an algorithm that computes the dynamics (loudness) of a performance from the audio signal, has been implemented in a system that tracks the tempo and dynamics in a given performance and shows the parameters (current values plus part of their history) in an animated display. The idea for this kind of visual representation was originally developed by the musicologist Jörg Langner [7]. As with tempo (equation 18), the dynamics trajectory is smoothed over the past via an exponential decay function. The system takes its input from an audio file or directly from the sound card and works in real time. For reasons that are evident (see figure 4), we call it the *Performance Worm*.

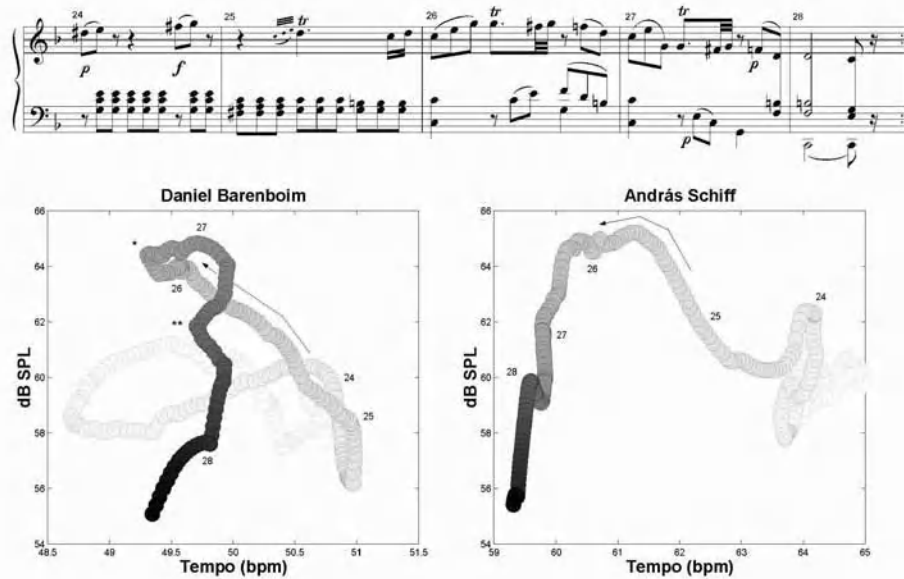


Fig. 4. Expression trajectories over the last bars (mm.24–28) of the Mozart piano sonata K.279, second movement, first section, as played by Daniel Barenboim (left) and András Schiff (right). *x* axis: tempo in beats per minute; *y* axis: dynamics ('loudness') in decibel. The darkest point represents the current instant (third beat of m.28), while instants further in the past appear fainter.

The Worm works interactively. The user can dynamically modify presentation parameters (e.g., rescale the axes) and, what is more, switch between tempo hypotheses, i.e., force the tempo tracker to re-weight its clusters and move to a higher or lower metrical level.

In this section, we briefly present an example of the Worm’s output to give an impression of the kind of insight offered by such an animation. Figure 4 shows a snapshot of the Worm as it tracks the performances of the same piece by two famous pianists. The piece is W.A. Mozart’s piano sonata K.279 (C major), first section of second movement (*Andante*), and the performances analysed are by Daniel Barenboim and András Schiff.¹ In the plots, the x axis represents the tempo in beats per minute (bpm), the y axis the loudness in terms of sound pressure level (measured in decibels). The darkest point represents the current instant, while instants further in the past appear fainter. We see the respective trajectories as they look when the performers have reached the very end of the section (beat 3 of measure 28). In the following, we will look at what these trajectories tell us about how the pianists played the last five bars of the piece (measures 24–28); the relevant part of the musical score is shown in figure 4 (top), and the positions in the trajectory that correspond to the beginnings of bars (24–28) have been manually marked in the plots, to make the following discussion easier. Note that the Worm knows nothing about the score or bar numbers.

Many interesting patterns emerge that reveal both commonalities and differences between the two performers. Some of these are clearly audible in the recording, others are hard to pinpoint and name when we hear the piece, but clearly emerge from the graphical representation (and contribute to the overall impression of the performance).

The most fundamental *similarity* between the two trajectories is the general leftward tendency in the tempo dimension (i.e., a slowing down) over the last 5 bars, combined with a strong downward movement (a reduction of loudness or *decrescendo*) over the last two bars. This is a well-known performance strategy for closing a piece; everything else would probably sound unmusical to us.

More interesting are the differences. The first striking difference is that Schiff plays the entire section much faster (above 60 bpm) than Barenboim (around 50 bpm). In fact, Schiff’s trajectory lingers around the 65 bpm mark for most of the piece and only moves left towards the 60 bpm mark in the final 5 bars of the piece — a very strong way of closing the piece by a long-term, gradual final *ritardando* that clearly sets the ending apart from the rest of the performance. Barenboim, on the other hand, spends most of his time (at least the last 8 or 9 bars) in the range of 50 bpm and also ends the piece there.

Another difference, again in the tempo dimension, is in the structuring of this final slowing down. Starting from m. 24, Schiff applies a continuous *ritardando* with no interruptions — his trajectory moves steadily to the left —, while Barenboim adds micro-structure to his *ritardando* by interjecting two little speedups at two musically similar points: the third beat of m. 26 (*) and the third beat of m. 27 (**).

In the dynamics dimension, an interesting difference is in the placement of the last rise in volume (*crescendo*) before the closing drop: Barenboim performs this *crescendo* during the *trill* in beat 2 of m. 25, while Schiff builds up the most of the volume (and

¹ Daniel Barenboim, Mozart: The Piano Sonatas, EMI Classics, 767 294-2, recorded 1985; András Schiff, Mozart: The Piano Sonatas, DECCA, 443 717-2, recorded 1980.

the tension) before the trill (see the arrows in figure 4, which indicate the durations of the trill). On the other hand, both artists combine this buildup in volume with a marked slowing down (the trajectories move from lower right to upper left). This might again represent a common performance strategy.

Many more interesting observations could be made. But this is not the place for an exhaustive musical analysis. What the example is meant to illustrate is how this approach to visualisation provides an intuitive view of a number of high-level aspects of expressive music performance. With a little experience, one immediately sees many interesting and typical features and patterns in a given trajectory. That makes the Worm an extremely useful tool for musical performance analysis.

4 Discussion

We have described a tempo tracking algorithm that extracts potential note onsets from audio input and estimates the current tempo of a piece of music in real time, and the application of this algorithm in a system for the visualisation of musical expression.

Early work in tempo and beat tracking focussed mainly on the converse of expression extraction, that is rhythm parsing, where deviations from metrical time were either not considered (e.g. [9]) or treated as noise (e.g. [10,2,11,1]), and the systems processed symbolic data off-line. More recently, several beat tracking systems have been developed which work with audio input (e.g. [12,4]) or run in real time (e.g. [8]) or both (e.g. [5, 6]). Compared with the real time audio beat tracking work of [5,6], our tempo tracking algorithm performs a simpler task, that of finding the tempo but not necessarily the beat. However, our work is not restricted to a particular musical style or tempo, whereas Goto's work is restricted to function only for popular music in 4/4 time, with a tempo range of 61–120 beats per minute, where either the drum patterns or the harmonic changes match assumed rhythmic patterns typical of pop music.

The values of parameters used in this paper were determined empirically, and are not necessarily optimal. In general, optimal values for parameters will depend on the data being processed. For example, the onset detection parameters depend on the instruments used; current values assume a reasonably sharp onset at the beginning of each tone. The tempo tracking parameters can be adjusted to suit different levels of rhythmic complexity and extents of tempo variation; the values cited here appear to work well with small to moderate tempo variations and any level of rhythmic complexity. We note that these two characteristics are interdependent, since by allowing greater variation in tempo, the system becomes more sensitive to rhythmic complexity, that is, more likely to interpret a complex rhythm as a change (or series of changes) in tempo. In further work, we intend to extend the system to allow input of high level (e.g. score) information, to make the system less dependent on parameter settings.

The expression tracking and visualisation system has a variety of interesting applications. For instance, it could be used for didactic purposes, for the visualisation and analysis of students' performances (e.g., in conservatories). Another interesting practical application would be in the automatic synchronisation of the music with other presentation aspects like lighting, videos, animations, etc. in live stage productions. Here, real time capabilities are essential. Note that our algorithms are by no means restricted to classical (or even tonal) music. In fact, they make no assumptions whatsoever regarding

the type of music or instruments they are dealing with, except that the notion of ‘tempo’ has to be applicable (i.e., there has to be some regular, recognisable rhythmic element).

A third application — and that is what we are using it for — is in the visualisation and analysis of the performance style of famous artists. We are currently starting a large-scale study on the typical style of various famous pianists, in an attempt to quantify and characterise at least some aspects of what has so far been discussed by musicologists and music lovers only in rather vague and aesthetic terms: what is it that distinguishes one great artist from another — what makes a Horowitz a Horowitz, to speak with [13]. This, we hope, will make an interesting contribution of AI to the world of music.

Acknowledgements. This research is part of the project Y99-INF, sponsored by the Austrian Federal Ministry of Education, Science and Culture (BMBWK) in the form of a START Research Prize. The BMBWK also provides financial support to the Austrian Research Institute for Artificial Intelligence.

References

- [1] A.T. Cemgil, B. Kappen, P. Desain, and H. Honing, ‘On tempo tracking: Tempogram representation and Kalman filtering’, in *Proceedings of the 2000 International Computer Music Conference*, pp. 352–355, San Francisco CA, (2000). International Computer Music Association.
- [2] P. Desain and H. Honing, ‘Quantization of musical time: A connectionist approach’, *Computer Music Journal*, **13**(3), 56–66, (1989).
- [3] S. Dixon, ‘Automatic extraction of tempo and beat from expressive performances’, *Journal of New Music Research*, **30**(1), 39–58, (2001).
- [4] S. Dixon and E. Cambouropoulos, ‘Beat tracking with musical knowledge’, in *ECAI 2000: Proceedings of the 14th European Conference on Artificial Intelligence*, pp. 626–630, Amsterdam, (2000). IOS Press.
- [5] M. Goto and Y. Muraoka, ‘A real-time beat tracking system for audio signals’, in *Proceedings of the International Computer Music Conference*, pp. 171–174, San Francisco CA, (1995). International Computer Music Association.
- [6] M. Goto and Y. Muraoka, ‘Real-time beat tracking for drumless audio signals’, *Speech Communication*, **27**(3–4), 331–335, (1999).
- [7] J. Langner and W. Goebel, ‘Representing expressive performance in tempo-loudness space’, in *Proceedings of the ESCOM 10th Anniversary Conference on Musical Creativity*, Liège, Belgium, (2002).
- [8] E.W. Large and J.F. Kolen, ‘Resonance and the perception of musical meter’, *Connection Science*, **6**, 177–208, (1994).
- [9] C.S. Lee, ‘The perception of metrical structure: Experimental evidence and a model’, in *Representing Musical Structure*, eds., P. Howell, R. West, and I. Cross, 59–127, Academic Press, San Diego CA, (1991).
- [10] H.C. Longuet-Higgins, *Mental Processes*, MIT Press, Cambridge MA, 1987.
- [11] D. Rosenthal, ‘Emulation of human rhythm perception’, *Computer Music Journal*, **16**(1), 64–76, (1992).
- [12] E.D. Scheirer, ‘Tempo and beat analysis of acoustic musical signals’, *Journal of the Acoustical Society of America*, **103**(1), 588–601, (1998).
- [13] G. Widmer, ‘What is it that makes it a Horowitz? Empirical musicology via machine learning’, in *Proceedings of the 12th European Conference on Artificial Intelligence (ECAI-96)*, Chichester, UK, (1996). Wiley & Sons.

Automatic Classification of Drum Sounds: A Comparison of Feature Selection Methods and Classification Techniques

Perfecto Herrera, Alexandre Yeterian, and Fabien Gouyon

Universitat Pompeu Fabra
Pg. Circumval·lació 8
08003 Barcelona, Spain
+34 935422806
{pherrera, ayeter, fgouyon}@iua.upf.es

Abstract. We present a comparative evaluation of automatic classification of a sound database containing more than six hundred drum sounds (kick, snare, hihat, toms and cymbals). A preliminary set of fifty descriptors has been refined with the help of different techniques and some final reduced sets including around twenty features have been selected as the most relevant. We have then tested different classification techniques (instance-based, statistical-based, and tree-based) using ten-fold cross-validation. Three levels of taxonomic classification have been tested: membranes versus plates (super-category level), kick vs. snare vs. hihat vs. toms vs. cymbals (basic level), and some basic classes (kick and snare) plus some sub-classes –i.e. ride, crash, open-hihat, closed hihat, high-tom, medium-tom, low-tom- (sub-category level). Very high hit-rates have been achieved (99%, 97%, and 90% respectively) with several of the tested techniques.

1. Introduction

Classification is one of the processes involved in audio content description. Audio signals can be classified according to miscellaneous criteria. A broad partition into speech, music, sound effects (or noises), and their binary and ternary combinations is used for video soundtrack descriptions. Sound category classification schemes for this type of materials have been recently developed [1], and facilities for describing sound effects have even been provided in the MPEG-7 standard [2]. Usually, music streams are broadly classified according to genre, player, mood, or instrumentation. In this paper, we do not deal with describing which instruments appear in a musical mixture. Our interest is much more modest as we focus only in deriving models for discrimination between different classes of isolated percussive sounds and, more specifically in this paper, of acoustic “standard” drum kit sounds (i.e. not electronic, not Latin, not brushed, etc.). Automatic labelling of instrument sounds has some obvious applications for enhancing sampling and synthesis devices’ operating systems in order to help sound designers to categorize (or suggesting names for) new patches and samples. Additionally, we assume that some outcomes of research on this subject will be used for the more ambitious task of describing the instrumentation in a musical recording, at least of the “rhythm loop” type.

Previous research in automatic classification of sound from music instruments has focused in instruments with definite pitch. Classification of string and wind instrument sounds has been attempted using different techniques and features yielding to varying degrees of success (see [3] for an exhaustive review). Classification of percussive instruments, on the other hand, has attracted little interest from researchers. In one of those above cited studies with pitched sounds, Kaminskyj [4] included three pitched percussive categories (glockenspiel, xylophone and marimba) and obtained good classification results (ranging from 75% to 100%) with a K-NN algorithm. Schloss [5] classified the stroke type of congas using relative energy from selected portions of the spectrum. He was able to differentiate between high-low sounds and open, muffled, slap and bass sounds. Using a K-means clustering algorithm, Bilmes [6] also was able to differentiate between sounds of three different congas. McDonald [7] used spectral centroid trajectories as classificatory features of sounds from percussive instruments. Sillanpää [8] used a representation of spectral shape for identification of the basic five categories of a drum kit: bass drum, snares, toms, hihats, and cymbals. His research was oriented towards transcription of rhythm tracks and therefore he additionally considered the case of identification of several simultaneous sounds. A database of 128 sounds was identified with 87% of accuracy for the case of isolated sounds. Performance dramatically dropped when there were two or three simultaneous sounds (respectively 49% and 8% for complete identification, though at least one of the sounds in the mixture was correctly identified all the times). In a subsequent study [9], the classification method used energy, Bark-frequency and log-time resolution spectrograms, and a fuzzy-c clustering of the original feature vectors into four clusters for each sound class. Weighted RMS-error fitting and an iterative spectral subtraction of models was used to match the test sounds against learnt models. Unfortunately, no systematic evaluation was presented this time. Goto and Murakoa [10] also studied drum sound classification in the context of source separation and beat tracking [11]. They implemented an “energy profile”-based snare-kick discriminator, though no effectiveness evaluation was provided. As a general criticism, in the previous research there is a lack of systematic evaluation of the different factors involved in automatic classification, and the databases are small to draw robust conclusions. A more recent study on this subject in the context of basic rhythmic pulse extraction [12] intended to be systematic, but also used a small database and a reduced set of descriptors.

Research on perceptual similarity of sounds is another area that provides useful information for addressing the problem of automatic classification of drum sounds. In perceptual studies, dis-similarity judgments between pairs of sounds are elicited from human subjects. With multidimensional scaling techniques, researchers find the dimensions that underlie to dis-similarity judgments. Even further, with proper comparison between those dimensions and physical features of sounds, it is possible to discover the links between perceptual and physical dimensions of sounds [13], [14], [15], [16]. A three dimensional perceptual space for percussive instruments (not including bells) has been hypothesized by Lakatos [17] (but also see [18]). This percussive perceptual space spans three related physical dimensions: log-attack time, spectral centroid and temporal centroid. Additional evidence supporting them has been gathered during the multimedia content description format standardization process (MPEG-7) and, consequently, they have been included in MPEG-7 as descriptors for timbres [19]. Graphical interactive testing environments that are linked

to specific synthesis techniques [20] seem to be a promising way for building higher-dimensional perceptual spaces.

From another area of studies, those focusing on characteristics of beaten objects, it seems that information about the way an object is hit is conveyed by the attack segment, whereas the decay or release segment conveys information about the shape and material of the beaten object [21]. Repp [22] found that different hand-clapping styles (palm-to-palm versus fingers-to-palm) correlated with different spectral envelope profiles. Freed [23] observed that the attack segment conveyed enough information for the subjects to evaluate the hardness of a mallet hit. Four features were identified as relevant for this information: energy, spectral slope, spectral centroid and the time-weighted average centroid of the spectrum. Kaltzky et al. [24] have got experimental results supporting the main importance of the decay part (specifically the decay rate) of a contact sound in order to identify the material of the beaten object.

In the next sections we will present the method and results of our study on automatic identification of drum sounds. First we will discuss the features we initially selected for the task and the ways for using the smallest set without compromising classification effectiveness. Some techniques consider relevance of descriptors without considering the classification algorithm in which they are being issued, but there are also attribute selection techniques that are linked to specific classification algorithms. We will compare both approaches with three different classification approaches: instance-based, statistical-based, and tree-based. Classification results for three taxonomic levels (super-category, basic level classes, and sub-categories) of drum-kit instruments will then be presented and discussed.

2. Method

2.1 Selection of Sounds

A database containing 634 sounds was set up for doing this study. Distribution of sounds into categories is shown in Table 1. Sounds were drawn from different commercial sample CD's and CD-ROMs. The main selection criteria were that they belonged to acoustic drums with as little reverberation as possible, and without any other effect applied to them. Also different dynamics and different physical instruments were looked for. Specific playing techniques yielding dramatic timbral deviations from a "standard sound" such as brushed hits or rim-shots were discarded.

Table 1. Categories used and number of sounds (inside parentheses) included in each category

Super-category	Basic-level	Sub-category
Membranes (380)	Kick (115)	Kick (115)
	Snare (150)	Snare (150)
	Tom (115)	Low (42)
		Medium (44)
		High (29)
Plates (263)	Hihat (142)	Open (70)
		Closed (72)
	Cymbal (121)	Ride (46)
		Crash (75)

2.2 Descriptors

We considered descriptors or features belonging to different categories: attack-related descriptors, decay-related descriptors, relative energies for selected bands and, finally, Mel-Frequency Cepstral Coefficients and variances. An amplitude-based segmentator was implemented in order to get an estimation of the attack-decay boundary position, for then computing those descriptors that used this distinction. Analysis window size for the computation of descriptors was estimated after computation of Zero-Crossing Rate.

2.2.1 Attack-Related Descriptors

Attack Energy (1), Temporal Centroid (2), which is the temporal centre of gravity of the amplitude envelope, Log Attack-Time (3), which is the logarithm of the length of the attack, Attack Zero-Crossing Rate (4), and TC/EA (5), which is the ratio of the Temporal Centroid to the length of the attack.

2.2.2 Decay-Related Descriptors

Decay Spectral Flatness (6) is the ratio between the geometrical mean and the arithmetical mean (this gives an idea of the shape of the spectrum, if it's flat, the sound is more "white-noise"-like; if flatness is low, it will be more "musical"); Decay Spectral Centroid (7), which is the centre of gravity of the spectrum; Decay Strong Peak (8), intended to reveal whether the spectrum presents a very pronounced peak (the thinner and the higher the maximum of the spectrum is, the higher value takes this parameter); Decay Spectral Kurtosis (9), the 4th order central moment (it gives clues about the shape of the spectrum: "peaky" spectra have larger kurtosis than scattered or outlier-prone spectra.), Decay Zero-Crossing Rate (10); "Strong Decay" (11), a feature built from the non-linear combination of the energy and temporal centroid of a frame (a frame containing a temporal centroid near its left boundary and strong energy is said to have a "strong decay"); Decay Spectral Centroid Variance (12); Decay Zero-Crossing Rate Variance (13); and Decay Skewness (14), the 3rd order central moment (it gives indication about the shape of the spectrum in the sense that asymmetrical spectra tend to have large skewness values).

2.2.3 Relative Energy Descriptors

By dividing the spectrum of the decay part into 8 bands of frequency, the energy lying in them was calculated, and then the relative energy percent for each band was computed. These bands were basically chosen empirically, according to the observations of several spectra from relevant instruments. The boundaries were fixed after several trials in order to get significant results, and were the following: 40-70 Hz. (15), 70-110 Hz. (16), 130-145 Hz. (17), 160-190 Hz. (18), 300-400 Hz. (19), 5-7 KHz. (20), 7-10 KHz. (21), and 10-15 KHz. (22).

2.2.4 Mel-Frequency Cepstrum Coefficients

MFCC's have been usually used for speech processing applications, though they have shown usefulness in music applications too [25]. As they can be used as a compact

representation of the spectral envelope, their variance was also recorded in order to keep some time-varying information. 13 MFCC's were computed over the whole signal, and their means and variances were used as descriptors. In order to interpret the selected sets of features in section 3, we will use the numeric ID's 23-35 for the MFCC means, and 36-48 for the MFCC variances.

2.3 Classification Techniques

We have selected three different families of techniques to be compared¹: instance-based algorithms, statistical modelling with linear functions, and decision tree building algorithms. The *K-Nearest Neighbors* (K-NN) technique is one of the most popular for instance-based learning and there are several papers on musical instrument sound classification using K-NN [26], [27], [28] [4]. As a novelty in this research context, we have also tested another instance-based algorithm called *K** (pronounced "K-star"), which classifies novel examples by retrieving the nearest stored example using an entropy measure instead of an Euclidean distance. Systematic evaluations of this technique using standard test datasets [29] showed a significant improvement of performance over the traditional K-NN algorithm.

Canonical discriminant analysis is a statistical modelling technique that classifies new examples after deriving a set of orthogonal linear functions that partition the observation space into regions with the class centroids separated as far as possible, but keeping the variance of the classes as low as possible. It can be considered like an ANOVA (or MANOVA) that instead of continuous to-be-predicted variables uses discrete (categorical) variables. After a successful discriminant function analysis, "important" variables can be detected. Discriminant analysis has been successfully used by [30] for classification of wind and string instruments.

C4.5 [31] is a decision tree technique that tries to focus on relevant features and ignores irrelevant ones for partitioning the original set of instances into subsets with a strong majority of one of the classes. Decision trees, in general, have been pervasively used for different machine learning and classification tasks. Jensen and Arnspang [32] or Wiczorkowska [33] have used decision trees for musical instrument classification. An interesting variant of C4.5, that we have also tested, is PART (partial decision trees). It yields association rules between descriptors and classes by recursively selecting a class and finding a rule that "covers" as many instances as possible of it.

2.4 Cross-Validation

For the forthcoming experiments the usual ten-fold procedure was followed: 10 subsets containing a 90% randomly selected sample of the sounds were selected for learning or building the models, and the remaining 10% was kept for testing them. Hit-rates presented below have been computed as the average value for the ten runs.

¹ The discriminant analysis was run with SYSTAT (<http://www.spssscience.com/SYSTAT/>), and the rest of analyses with the WEKA environment (www.cs.waikato.ac.nz/~ml/).

3. Results

3.1 Selection of Relevant Descriptors

Two algorithm-independent methods for evaluating the relevance of the descriptors in the original set have been used: Correlation-based Feature Selection (hence CFS) and ReliefF. CFS evaluates subsets of attributes instead of evaluating individual attributes. A “merit” heuristic is computed for every possible subset, consisting of a ratio between how predictive a group of features is and how much redundancy or inter-correlation there is among those features [34]. Table 2 shows the CFS-selected features in the three different contexts of classification we are dealing with. Note that a reduction of more than fifty percent can be achieved in the most difficult case, and that the selected sets for basic level and for sub-category classification show an important overlap.

Table 2. Features selected by the CFS method

Super-category	[21, 4, 22]
Basic-level	[2, 4, 5, 6, 7, 9, 10, 14, 15, 16, 17, 18, 20, 21, 22, 26, 27, 30, 39]
Sub-category	[1, 2, 3, 4, 5, 6, 7, 9, 10, 14, 15, 16, 17, 18, 19, 20, 21, 22, 26, 30, 39]

ReliefF evaluates the worth of an attribute by repeatedly sampling an instance and considering the value of the given attribute for the nearest instance of the same and for the nearest different class [34]. Table 3 shows the ReliefF-selected features in the three different contexts. Note that the list is a ranked one –from most to least relevant– and that we have matched the cardinality of this list to the one yielded by the previous method, in order to facilitate their comparisons.

Table 3. Features selected by the ReliefF method

Super-category	[9, 14, 7]
Basic-level	[9, 14, 7, 19, 10, 17, 4, 25, 18, 6, 15, 21, 20, 16, 24, 26, 30, 31, 13]
Sub-category	[9, 14, 19, 7, 17, 10, 4, 25, 16, 15, 18, 6, 21, 20, 24, 30, 26, 31, 13, 28, 2]

Comparing the two methods it can be seen that all selected subsets for basic-level or for sub-category share more than 60% of features (but surprisingly they do not coincide at all when the target is the super-category). It is also evident that they include quite a heterogeneous selection of descriptors (some MFCC’s, some energy bands, some temporal descriptors, some spectral descriptors...).

Contrasting with the previous “filtering” approaches, we also tested a “wrapping” approach for feature selection [35]. This means that features are selected in connection with a given classification technique which acts as a wrapper for the selection. Canonical Discriminant Analysis provides numerical indexes in order to decide about the relevance of a feature (but after analysis, not prior to it) as for example the F-to-remove value, or the descriptor’s coefficients inside the canonical

functions. For feature selection inside CDA it is usual to follow a stepwise (usually backwards) procedure. This strategy, however, only grants a locally optimal solution, so that an exhaustive (but sometimes impractical) search of all the combinations is recommended [36]. In our case, we have proceeded with a combination of backward stepwise plus some heuristic search. Table 4 shows the selected subsets, with the features ranked according the F-to-remove value (the most relevant first). A difference related to the filtering approaches is that with CDA the selected sets are usually larger. A large proportion of the selected features, otherwise, match those selected with the other methods.

Table 4. Features selected after Canonical Discriminant Analyses

Super-category	[4, 13, 19, 20, 37, 39]
Basic-level	[15, 9, 4, 20, 14, 2, 13, 26, 27, 3, 19, 8, 21, 39, 6, 11, 38]
Sub-category	[16, 15, 3, 9, 2, 17, 20, 13, 14, 19, 27, 26, 39, 7, 12, 10, 8, 37, 38, 4, 21, 22, 25, 33, 30, 29, 5, 24, 28, 45, 36, 34]

3.2 Classification Results

We tested the three algorithms using the different subsets discussed in the previous section. Three different levels of classification were tested: super-category (plates versus membranes), basic-level (the five instruments) and sub-category (kick and snare plus some variations of the other three instruments: open and closed hihat, low, mid and high tom, crash and ride cymbal). Tables 5, 6 and 7 summarize the main results regarding hit rates for the three different classification schemes we have tested. Rows contain the different algorithms and columns contain the results using the different sets of features that were presented in the previous section. For the C4.5, the number of leaves appears inside parentheses. For PART, the number of rules appears inside parentheses. The best method for each feature set has been indicated with bold type and the best overall result appears with grey background.

Table 5. Super-category classification hit rates for the different techniques and feature selection methods

	All features	CFS	ReliefF	CDA
K-NN (k=1)	99.2	97.9	93.7	96.7
K*	98.6	97.8	94.8	96.7
C4.5	97.2 (8)	98.6 (8)	94.8 (12)	95.1(14)
PART	98.4 (5)	98.2 (6)	94.4 (6)	95.1(9)
CDA	99.1	94.7	88.1	99.3

Table 6. Basic-level classification hit rates for the different techniques and feature selection methods

	All features	CFS	ReliefF	CDA
K-NN (k=1)	96.4	95	95.6	95.3
K*	97	96.1	97.4	95.8
C4.5	93 (20)	93.3(21)	92.2(23)	94.2(18)
PART	93.3 (12)	93.(11)	93.1(11)	93.6(12)
CDA	92	93	91	95.7

Table 7. Sub-category classification hit rates for the different techniques and feature selection methods

	All features	CFS	ReliefF	CDA
K-NN (k=1)	89.9	87.7	89.4	87.9
K*	89.9	89.1	90.1	90.7
C4.5	82.6 (40)	83 (38)	81 (45)	85(43)
PART	83.3 (24)	84.1(27)	81.9 (29)	84.3(27)
CDA	82.8	86	82	86.6

A clear interaction effect between feature selection strategy and algorithm family can be observed: for instance-based algorithms ReliefF provides the best results while for the decision-trees the best results have been obtained with CFS. In the case of decision trees, selecting features with CFS is good not only for improving hit-rates but also for getting more compact trees, (i.e. with a small number of leaves and therefore smaller in size). As expected, the CDA-selected features have yielded the best hit-rates for the CDA, but surprisingly they have also yielded the best hit-rates for most of the decision-trees.

It is interesting to compare the results obtained using feature selection with those obtained with the whole set of features. For the super-category classification it seems that all the selection procedures have operated an excessive deletion and performance has degraded up to 4% when using a selected subset. Note however that in this classification test the best overall result (CDA features with CDA classification) outperforms any of the figures obtained with the whole subset. For the basic-level and sub-category tests, the reduction of features degrades the performance of instance-based methods (but less than 1%), whereas it improves the performance of the rest.

After comparing families of algorithms it is clear that differences between them increase as the task difficulty increases. It is also evident that the best performance is usually found in instance-based ones (and specifically K* yields slightly better results than a simple K-NN), whereas tree-based yield the worst figures and CDA lies in between. Although decision trees do not provide the best overall performance, they have an inherent advantage over instance-based: expressing relationships between features and classes in terms of conditional rules. Table 8 exemplifies the type of rules that we get after PART derivation.

Table 8. Some of the PART rules for classification at the "basic-level". Correctly and wrongly classified instances are shown inside parentheses. We have left out some less general rules for clarity

SKEWNESS > 4.619122 AND B40HZ70HZ > 7.784892 AND MFCC3 <= 1.213368: Kick (105.0/0.0)	
KURTOSIS > 26.140138 AND TEMPORALCE <= 0.361035 AND ATTZCR > 1.478743: Tom (103.0/0.0)	SPECCENTROID > 11.491498 AND B1015KHZ > 0.791702: HH (100.0/2.0)
B710KHZ <= 0.948147 AND KURTOSIS <= 26.140138 AND ATTZCR <= 22.661397: Snare (133.0/0.0)	SKEWNESS <= 4.485531 AND B160HZ190HZ <= 5.446338 AND MFCC3VAR > 0.212043 AND MFCC4 > -0.435871: Cymbal (110.0/3.0)

Regarding CDA, an examination of the canonical scores plots provides some graphical hints about the performance of the four canonical discriminant functions needed for the basic-level case: the first one separates toms+kicks from hihats+cymbals, the second one separates the snare from the rest, the third one separates cymbals from hihats, and the fourth one separates toms from kicks. It should be noted that in the other cases it is more difficult to assign them a clear role.

Inspecting the confusion matrix for the instrument test, most of the errors consist in confusing cymbals with hihat, and tom with kick (and their inverse confusions, though with a lesser incidence). For the sub-instrument test, 60% of the misclassifications appear to be intra-category (i.e. between crash and ride, between open and closed hihat, etc.), and they are evenly distributed.

4. Discussion

We have achieved very high hit rates for the automatic classification of standard drum sounds into three different classification schemes. The fact that, in spite of using three very different classification techniques, we have obtained quite similar results could mean that the task is quite an easy one. It is true that the number of categories we have used has been small even for the most complex classification scheme. But it should also be noted that there are some categories that, at least from a purely perceptual point of view, do not seem to be easily separated (for example, low-toms from some kicks, or some snares from mid-toms or from some crash cymbals). Therefore, a contrasting additional interpretation for this good performance is to consider that our initial selection of descriptors was good. This statement gets support by the fact that the all-feature results are not much worse than results after feature selection. In the case of having a bad initial set, those bad features would have contributed to worsen the performance. As it has not been the case, we can conclude that from a good set of initial features, some near-optimal sets have been identified with the help of filtering or wrapping techniques. Most of the best features found can be considered as spectral descriptors: skewness, kurtosis, centroid, MFCC's. We included a very limited number of temporal descriptors, but, as expected, apart from ZCR, they do not seem to be needed for precise instrument classification.

In the section of improvements for subsequent research we may list the following: (1) A more systematic approach to description in terms of energy bands (for example, using Bark measures); (2) Evaluation of whole-sound descriptors against attack-decay decomposed descriptors (i.e. the ZCR); (3) Non-linear scaling of some feature dimensions; (4) Justified deletion of some observations (after analyzing the models, it seems that some outliers that contribute to the increment of the confusion rates should be considered as “bad” examples for the model because of audio quality or wrong class adscription).

5. Conclusions

In this study, we have performed a systematic study of the classification of standard drum sounds. After careful selection of descriptors and its refinement with different techniques, we have achieved very high hit-rates in three different classification tasks: super-category, basic-level category, and sub-category. In general, the most relevant descriptors for them seem to be ZCR, kurtosis, skewness, centroid, relative energy in specific bands, and some low-order MFCC's. Performance measures classification techniques have not yielded dramatic differences between classification techniques and therefore selecting one or another is clearly an application-dependent issue. We believe, though, that relevant performance differences will arise when more classes are included in the test, as we have planned for a forthcoming study. Regarding classification of mixtures of sounds, even if it is not yet clear if the present results will be useful, we have gathered interesting and relevant data in order to characterize different classes of drum sounds.

Acknowledgments. The research reported in this paper has been partially funded by the EU-IST project CUIDADO.

References

- [1] Zhang, T. and Jay Kuo, C.-C.: Classification and retrieval of sound effects in audiovisual data management. In Proceedings of 33rd Asilomar Conference on Signals, Systems, and Computers (1991)
- [2] Casey, M.A.: MPEG-7 sound recognition tools. IEEE Transactions on Circuits and Systems for Video Technology, 11, (2001) 37-747
- [3] Herrera, P., Amatriain, X., Batlle, E., Serra, X.: A critical review of automatic musical instrument classification. In Byrd, D., Downie, J.S., and Crawford, T (Eds.), Recent Research in Music Information Retrieval: Audio, MIDI, and Score Kluwer Academic Press, in preparation.
- [4] Kaminskyj, I.: Multi-feature Musical Instrument Sound Classifier. In Proceedings of Australasian Computer Music Conference (2001)
- [5] Schloss, W.A.: On the automatic transcription of percussive music -from acoustic signal to high-level analysis. STAN-M-27. Stanford, CA, CCRMA, Department of Music, Stanford University (1985)
- [6] Bilmes, J.: Timing is the essence: Perceptual and computational techniques for representing, learning and reproducing expressive timing in percussive rhythm. MSc, Thesis. Massachusetts Institute of Technology, Media Laboratory. Cambridge, MA. (1993)

- [7] McDonald, S. and Tsang, C.P.: Percussive sound identification using spectral centre trajectories. In Proceedings of 1997 Postgraduate Research Conference (1997)
- [8] Sillanpää, J.: Drum stroke recognition. Tampere University of Technology. Tampere, Finland (2000)
- [9] Sillanpää, J., Klapuri, A., Seppänen, J., and Virtanen, T.: Recognition of acoustic noise mixtures by combined bottom-up and top-down approach. In Proceedings of European Signal Processing Conference, EUSIPCO-2000 (2000)
- [10] Goto, M., Muraoka, Y.: A sound source separation system for percussion instruments. Transactions of the Institute of Electronics, Information and Communication Engineers D-II, J77, 901-911 (1994)
- [11] Goto, M. and Muraoka, Y.: A real-time beat tracking system for audio signals. In Proceedings of International Computer Music Conference, 171-174 (1995)
- [12] Gouyon, F. and Herrera, P.: Exploration of techniques for automatic labeling of audio drum tracks' instruments. In Proceedings of MOSART: Workshop on Current Directions in Computer Music (2001)
- [13] Miller, J.R., Carterette, E.C.: Perceptual space for musical structures. Journal of the Acoustical Society of America, 58, 711-720 (1975)
- [14] Grey, J.M.: Multidimensional perceptual scaling of musical timbres. Journal of the Acoustical Society of America, 61, 1270-1277 (1977)
- [15] McAdams, S., Winsberg, S., de Soete, G., and Krimphoff, J.: Perceptual scaling of synthesized musical timbres: common dimensions, specificities, and latent subject classes. Psychological Research, 58, 177-192 (1995)
- [16] Toivainen, P., Kaipainen, M., and Louhivuori, J.: Musical timbre: Similarity ratings correlate with computational feature space distances. Journal of New Music Research, 282-298 (1995)
- [17] Lakatos, S.: A common perceptual space for harmonic and percussive timbres. Perception and Psychophysics, 62, 1426-1439 (2000)
- [18] McAdams, S., Winsberg, S.: A meta-analysis of timbre space. I: Multidimensional scaling of group data with common dimensions, specificities, and latent subject classes (2002)
- [19] Peeters, G., McAdams, S., and Herrera, P.: Instrument sound description in the context of MPEG-7. In Proceedings of Proceedings of the 2000 International Computer Music Conference (2000)
- [20] Scavone, G., Lakatos, S., Cook, P., and Harbke, C.: Perceptual spaces for sound effects obtained with an interactive similarity rating program. In Proceedings of International Symposium on Musical Acoustics (2001)
- [21] Laroche, J., Meillier, J.-L.: Multichannel excitation/filter modeling of percussive sounds with application to the piano. IEEE Transactions on Speech and Audio Processing, 2, (1994) 329-344
- [22] Repp, B.H.: The sound of two hands clapping: An exploratory study. Journal of the Acoustical Society of America, 81, (1993) 1100-1109
- [23] Freed, A.: Auditory correlates of perceived mallet hardness for a set of recorded percussive events. Journal of the Acoustical Society of America, 87, (1990) 311-322
- [24] Klatzky, R.L., Pai, D.K., and Krotkov, E.P.: Perception of material from contact sounds. Presence: Teleoperators and Virtual Environments, 9, (2000) 399-410
- [25] Logan, B.: Mel Frequency Cepstral Coefficients for Music Modeling. In Proceedings of International Symposium on Music Information Retrieval, ISMIR-2000. Plymouth, MA, (2000)
- [26] Martin, K.D. and Kim, Y.E.: Musical instrument identification: A pattern-recognition approach. In Proceedings of Proceedings of the 136th meeting of the Acoustical Society of America. (1998)
- [27] Fujinaga, I. and MacMillan, K.: Realtime recognition of orchestral instruments. In Proceedings of the 2000 International Computer Music Conference, (2000) 141-143

- [28] Eronen, A.: Comparison of features for musical instrument recognition. In Proceedings of 2001 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA'01) (2001)
- [29] Cleary, J.G. and Trigg, L.E.: K*: An instance-based learner using an entropic distance measure. In Proceedings of International Conference on Machine Learning, (1995) 108-114
- [30] Agostini, G., Longari, M., and Pollastri, E.: Musical instrument timbres classification with spectral features. In Proceedings of IEEE Multimedia Signal Processing Conference (2001)
- [31] Quinlan, J.R.: C4.5: Programs for machine learning. Morgan Kaufmann. San Mateo, CA, (1993)
- [32] Jensen, K. and Arnspang, J.: Binary decision tree classification of musical sounds. In Proceedings of the 1999 International Computer Music Conference. (1999)
- [33] Wiczorkowska, A.: Classification of musical instrument sounds using decision trees. In Proceedings of the 8th International Symposium on Sound Engineering and Mastering, ISSEM'99, (1999) 225-230
- [34] Hall, M.A.: Correlation-based feature selection for discrete and numeric class machine learning. In Proceedings of Seventeenth International Conference on Machine Learning (2000)
- [35] Blum, A., Langley, P.: Selection of relevant features and examples in machine learning. Artificial Intelligence, 97, (1997) 245-271
- [36] Huberty, C.J.: Applied discriminant analysis. John Wiley. New York (1994)

Some Formal Problems with Schenkerian Representations of Tonal Structure

Tim Horton

Faculty of Music, University of Cambridge, Cambridge, CB3 9DP, UK
tjh20@cam.ac.uk

Abstract. This paper is concerned with the type of data structures required for the accurate representation of tonal structure, in particular with the atomic/combinatorial nature of the non-terminal symbols within such representations. The theoretical commitments inherent in the form of Schenkerian representations will be examined, followed by a critical analysis of the consequences of these commitments for the descriptive adequacy of Schenkerian theory. Specifically, the structural phenomena examined here suggest that the form required of a successful tonal theory is a combinatorial syntax, an observation with important ramifications for the question of the computational architecture underlying our cognition of tonal music.

1 An Overview of Representations of Hierarchical Structure

That pitch structure within certain musical idioms is based upon the embellishment of structurally significant events by sets of ornamental events is a fact that has been recognised for some time (Ganassi [1], for example, provides a very early articulation of this insight). In particular, the recursive application of such a process of embellishment results in a syntactic hierarchy, whereby an event involved in the elaboration of another can itself be subject to elaboration by other ornamental events.

There are two different syntactic relations in terms of which we can formalise this insight: constituency and dependency. The relation of constituency captures the fact that the atomic elements of a complex object group together on the basis of their structural functions into component parts, which in turn group together into larger parts, and so on, until the whole object is obtained. Constituent structure is usually represented either through bracketing or through a tree diagram, indicating how units concatenate into the *strings* from which an object is composed at different levels (Fig. 1a).¹

The relation of dependency captures the fact that one element may be subordinate to another on the basis of their respective structural functions, which in turn may be subordinate to another, and so on, until the structurally most important element of the object is reached. Dependency structure is usually represented either through arrows (which in the diagram below point from a subordinate element to its superordinate element) or through a tree diagram, indicating the chain of subordination between elements (Fig. 1b). The element governing a group of subordinate elements (whether

¹ For an example of constituent structure trees within tonal theory, see Keiler [2].

these subordinates are directly dependent upon the governing element, as C is to B, or whether they are dependent upon another element that is itself dependent upon the governing element, as C is to E) is called the *head* of that group. Such a group is called a *subtree* as it consists of elements dominated by a single node and could therefore be detached as an independent tree structure.²

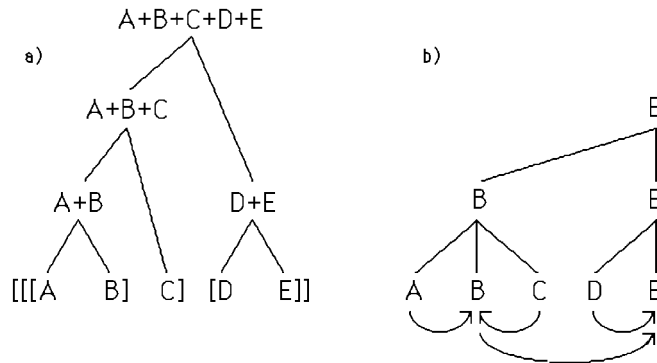


Fig. 1. Diagrams representing constituency relations (1a) and dependency relations (1b)

Before continuing, it is worth observing an important difference in the nature of these two types of syntactic tree. A constituent structure tree is combinatorial: the non-terminal nodes represent strings of elements and thus successively higher levels of the tree consist of larger and larger constituents (only the terminal nodes represent atomic elements). Each level of the tree therefore contains all of the elements that comprise the object, and, as a result, it is possible to read across the nodes and branches at any particular level and get a structural description of the entire object, albeit a partial description (at only one structural level).

A dependency tree, by contrast, is atomistic: every node of the tree, both terminal and non-terminal, represents an atomic element and thus successively higher levels of the tree consist of fewer and fewer elements. Each level of the tree therefore contains only those elements of the same dependency rank, and, as a result, it is *not* possible to read across the nodes at any particular level and obtain a structural description of the entire object.

Note also that a constituent is just a head plus all its dependents: if you concatenate the set of elements consisting of a head and the subordinate events it governs, you obtain the string of the corresponding constituent. That this is so should not be surprising, since both syntactic relations derive from the same considerations of the structural functions of elements. These syntactic relations, therefore, are essentially two sides of the same coin; whenever there is dependency there is also constituency, and vice versa. This was formally proved by Gaifman [4], who demonstrated that constituency and dependency systems are *weakly equivalent*: any object that can be described in terms of one relation can also be described in terms of the other.

In the same paper, Gaifman also proved that pure constituency and dependency systems are not *strongly equivalent*, that is, they do not assign exactly the same structural descriptions to objects. Though the constituent structure representation in

² For an example of dependency trees within tonal theory, see Lerdahl and Jackendoff [3].

Fig. 1a isolates groups of elements, each consisting of a head plus its dependents, it does not mark out which element is the head of each (for example, it does not show that B is the head of A+B). And though the dependency representation in Fig. 1b partitions the whole object into strings, in some cases where two elements are dependent upon the same head, the representation does not indicate the relative structural level at which each subordinate combines with its head (for example, it does not show that C enters into construction with B at a higher level than does A). A consequence of these two systems not being strongly equivalent, then, is that it is not possible to derive completely one representation from the other.

Each system can be enriched to make them strongly equivalent (which is how they are used in practice). In the constituent structure representation, the identity of the heads of each constituent can be encoded in the node labels. In Fig. 2a, for example, the label 'BP', stands for 'B Phrase', indicating B as the head of the constituent A+B. In the dependency representation, the structural level at which each dependent enters into construction with the head can be encoded in the relative height of the branch attachments, thus enabling each constituent to be identified as a distinct subtree. In Fig. 2b, for example, the subtree A+B is distinguished from the subtree A+B+C.

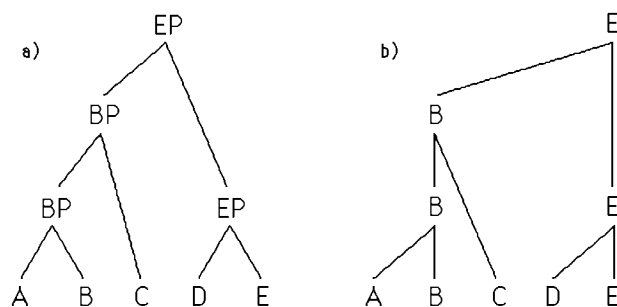


Fig. 2. Tree diagrams exemplifying the enriched constituency (2a) and dependency (2b) systems

There is an assumption behind this discussion, however, one that the talk of equivalence presupposes: note that, in order for the dependency system to encode the part-whole relations captured in the constituent structure representation, we had to introduce concatenation, that is, a combinatorial operation over the nodes of the dependency tree, in order to derive subtrees.

Both dependency and constituency systems can be equally appropriate for representing syntactic hierarchies. In practice, there may be a sufficient number of observations that need to be stated over the constituent structure of an object – commonalities in the distribution of atomic and complex parts, descriptions of processes involving the tokening of complex parts, etc. – that theorists often choose to encode constituency as a direct relation (using a constituent structure formalism), rather than as a derived one (using a dependency formalism). But this is to some extent arbitrary; because the theorist needs to refer to constituents, it's simply more convenient. What I want to emphasise, however, is that the use of the dependency formalism must invoke a combinatorial operation. In other words, although a dependency tree in this enriched system is still atomistic, containing only individual

events at each node, the *representational framework*, that is, the level of description that is taken to encode the structural properties of an object, has become combinatorial.

2 The Evolution of Schenkerian Representations

The hierarchical component of tonal structure received its fullest description in the work of the theorist Heinrich Schenker [5]. The richness of Schenker's understanding of the recursive nature of tonal structure came about from an important insight of his, which today remains perhaps the most remarkable contribution to tonal theory. Schenker realised that tonal structure is linearly derived, that is, harmonic progression at any structural level has its origin in voice leading between the superordinate events on that level. It follows that a necessary step in explaining the tonal structure of a work is to reveal the linear processes from which it is derived. In order to do this, Schenker evolved the technique of *reduction*, whereby structurally less important pitch events are successively eliminated in a series of stages to reveal the underlying voice-leading progressions.

Between 1920 and 1925, a fundamental development took place in Schenker's work, one which, though tacit in expositions of Schenkerian theory, is of enormous theoretical significance. Schenker moved from merely *explaining* tonal structure in terms of linear processes to actually using these processes as a vehicle for the *representation* of tonal structure. This shift can be seen quite clearly by comparing the notational formalism of some of his early and late analyses (Fig. 3).

In the example of Fig. 3a, each layer of the reduction consists of a succession of pitch events situated in line with their real-time locations on the musical surface; in the example of Fig. 3b, by contrast, these event successions have been transformed into large-scale contrapuntal formations that span the whole composition and that are thus understood to characterise the tonal structure of the composition at some level (a fact reflected in the systematised use of horizontal lines under the staves, indicating the extent of the prolongation of particular harmonies). Whereas the counterpoint of Schenker's early analytic diagrams was an *explanatory tool* to illustrate the underlying linear formations from which a harmonic progression has been generated, the counterpoint of Schenker's later analytic diagrams was a *medium of representation*, enabling him to describe tonal structure using the concepts of counterpoint.³

Music theorists subsequently took the notational formalism of Schenker's later analyses to be a necessary step in the expression of his insights. In doing so, however, they created something of a fallacy in the foundation of modern tonal theory, which can be stated as follows: *because tonal structure is linearly derived (and therefore an understanding of its linear derivation is an integral component of its explanation), this means that it is necessary to describe and represent tonal structure in terms of*

³ In fact, these two diagrams are not strictly analogous since the earlier one is not a systematic reduction of the sort discussed here, but rather an illustration of various different aspects of underlying voice leading at one particular structural level. Nevertheless, the point about the difference in the representational status of the entities in these early and late reductions still stands.

that linear derivation and its attendant concepts. But this simply isn't true: linear structure and tonal structure can perfectly well be described in different representations.

a)

Fig. 29
T. 11 12 13 14 15 16 17 18 19 20 21 22 23

a) F dur I/VI-1st
a moll

b) F dur I/VI-1st
a moll

c)

d)

e) F dur IIIrd

IV VII III VI⁵⁻⁶ V/II³⁻⁴ - ^{1st} 1 V 1st - ^{1st} 1st

IV V/II³⁻⁴ d moll I/VI F dur

VI

I

b)

Chopin, Étude op. 10 no. 1

a)

b)

I - VI - I - V - I

m. 1 3 5 7 9 11 13 15 17 19 21 23

(n.a.) (n.a.) (n.a.) (n.a.) (n.a.) (n.a.) (n.a.) (n.a.) (n.a.) (n.a.) (n.a.) (n.a.) (n.a.)

I - VI - I - V - I

(n.a.) (n.a.) (n.a.) (n.a.) (n.a.) (n.a.) (n.a.) (n.a.) (n.a.) (n.a.) (n.a.) (n.a.) (n.a.)

I - VI - I - V - I

(n.a.) (n.a.) (n.a.) (n.a.) (n.a.) (n.a.) (n.a.) (n.a.) (n.a.) (n.a.) (n.a.) (n.a.) (n.a.)

Fig. 3. Reductions from early (3a) and late (3b) analyses by Schenker (Fig.3a: [6], p.37; Fig.3b: [5], Figure 130/4b)

In particular, it has become an assumption of tonal theory that in order to describe the hierarchical nature of pitch structure in tonal music, you need to use a reductive system (see, for example, Salzer [7], p.10-2). But as we saw in the opening section, in order to describe the hierarchical structure of an object it is by no means necessary to remove systematically its least important parts. Reduction is a technique for illustrating the *linear* structure of music; it only became used for describing the hierarchical nature of pitch structure in tonal music once Schenker had decided to use linear structure as a vehicle for representing tonal structure.

It should be evident from the foregoing discussion that, post-Schenker, the term 'reduction' has a highly specific meaning in music theory, one that should not be

confused with the more general use of the term in philosophy and mathematics (where it refers to the characterisation or definition of the elements of one domain in terms of the vocabulary of another domain). Specifically, a reduction in music theory is a method of representing a musical surface in terms of a series of layers. Each layer is defined with reference to a particular dependency rank and consists of the subset of events from the musical surface above that dependency rank. (Thus, the higher the dependency rank that a layer is defined by, the smaller the subset of events on that layer.) The subset of events comprising each layer form a *voice-leading progression* and are held to constitute a representation of the musical surface at the structural level appropriate to that dependency rank.

3 The Consequences of Reduction for the Form of Schenkerian Representations

The decision to describe tonal structure in terms of its linear derivation has profound consequences for the form of Schenkerian representations. Most importantly, it consigns them to being atomistic: if the tonal structure of a composition at any particular hierarchical level is to be conceptualised as a voice-leading progression, then this requires that it be represented in terms of configurations of *individual events* (since voice leading is obviously a phenomenon that occurs between individual events). A result of adopting an analytic system that represents a musical surface in terms of a series of reductive layers, then, is that the part of the system that is taken to encode the structural properties of a musical work is unable to admit complex data structures.

In other words, to employ a reduction as a medium of representation is to use a dependency formalism in a particular way. We have already seen that, unlike constituent structure trees, it is not possible to read across a layer of nodes in a dependency tree and still have a representation of the entire object. This is why, in the enriched dependency system examined in the first section, the *representation* of an object included a partitioning of the dependency tree into subtrees. By contrast, a reductive representation uses the series of elements at each level of a dependency tree to generate a large-scale linear formation that is understood to characterise the entire object at that particular structural level, with the complete representation comprising a whole series of such levels.

To summarise the discussion so far, a consequence of the commitment to represent tonal structure in terms of its linear derivation is that Schenkerian representations are incapable of encoding the syntactic relation of constituency; rather, at any structural level, one single event has to represent several events.

Because the form of a music theory's representations determines the conceptual framework within which compositions are described, the atomism of Schenkerian representations has crucial implications for the sorts of concepts Schenkerian theory uses to describe tonal structure. In particular, the way in which Schenkerian theory attempts to relate information concerning part-whole relations is through the concept of *prolongation*. It is worth taking a moment to examine how this works.

Constituent structure representations capture directly the intuition that tonal compositions consist of functionally determined parts by representing such parts as

strings of events. Schenkerian theory, however, cannot do this: strings of events are combinatorial entities, which are therefore incapable of participating in linear processes. Hence, it describes each functional part in terms of the prolongation of the head itself. Indeed, this is what lies behind its portrayal of the sequence of heads at any particular hierarchical level as a large-scale linear progression spanning the entire length of a composition: since each stage of a linear progression lasts a certain period of time, a period that will obviously contain a certain number of subordinate events, such a move enables the characterisation of the musical surface, at that particular hierarchical level, as consisting of functionally determined parts. Lerdahl and Jackendoff [3] explain this aspect of Schenkerian thinking as follows:

„The largest levels of our reductions...are analogous to stages of phrase-structure analysis [that is, constituent structure analysis] as represented in linguistic trees. Thus the highest level in one of our reductions (always the tonic chord) roughly corresponds to ‘Sentence’, our second largest level (the ‘basic form’ in prolongational reduction) roughly corresponds to ‘Noun Phrase + Verb Phrase’, and so forth.“ ([3], p.287)

Prolongation is thus a paradigmatic example of how the concepts of Schenkerian theory relate to the form of its representations. The concept of prolongation is a way of trying to convey information concerning part-whole relations within an atomistic representational system (one that must be atomistic in order to serve as the basis for illustrating linear pitch structure). On the one hand, prolongation is an operation that acts upon individual pitch events; on the other, prolongation establishes the notion of the functionally determined parts of a piece. It thus enables Schenkerian theory to articulate information about constituent structure whilst retaining the commitment to describe and represent tonal structure using the concepts of counterpoint. I now want to suggest, however, that the atomism of reductive representations has highly problematic consequences for their descriptive adequacy.

4 Formal Problems with Schenkerian Representations of Tonal Structure

4.1 Coordination

A major problem with reductive representations is that they must necessarily assume that constituents have single heads. There are occasions, however, where dependency relations cannot accurately be described solely in terms of the subordination of one event to another. One such example is a neighbour-note construction (Fig. 4a). In this case, the supertonic chord is not subordinate to either one of the surrounding dominant chords, but to *both* of them. The phrase therefore has two heads and consequently needs to be characterised not in terms of the *subordination* of one event to another, but rather in terms of the *coordination* of two events, the dominant chords, with a subordinate auxiliary element, the supertonic chord, connecting them.

(In this and all subsequent figures, D=Dominant, M=Mediant, S=Subdominant, T=Tonic, XA= X Articulation, XP= X Phrase. I have chosen to construct syntactic representations over harmonic-function categories rather than scale-degree categories,

though this decision is not relevant to the discussion here, which concerns the issue of whether descriptions of syntactic structure should be reductive or combinatorial.)

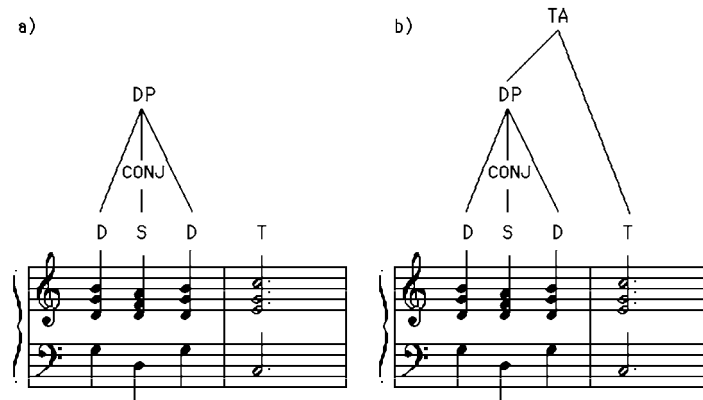


Fig. 4. A symmetrical neighbour-note construction as an example of coordination

This problem is a symptom of a more general malaise, borne out of the atomism that reductive representations necessitate: they must always take dependency to be a relation between *individual events*. Yet look at the next structural level up (Fig. 4b); because no single event can be labelled as head of the dominant phrase, it is necessary to regard the *whole set of heads* (in this case, two of them) as dependent upon the final tonic chord. Thus, it is not always possible to construe dependency purely as a relation between individual events because coordinative constructions, such as the one illustrated here, do not have single heads.

Schenkerian methodology, however, forces a dilemma on the analyst in these situations. The entities on each structural level of a Schenkerian representation must be individual events, from which contrapuntal lines can be constructed. The reduction procedure therefore imposes an arbitrary choice of head, so that a single event can be retained at the next structural level above. An accurate account of dependency relations, however, may require that an analytic system recognise combinatorial data structures.⁴

4.2 Exocentricity

The atomism of reductive representations has problematic consequences even for the description of certain constructions based solely on the subordination of one event to another.

⁴ The teleological nature of music's sequential unfolding means that coordinative constructions of the form X-CONJ-X are less common in music than in natural language. Nevertheless, such constructions do exist, and not only over small scales, like the construction just examined, but also at the highest levels of pieces, for example, Chopin's Mazurka, Op.17, No.3.

One particular assumption behind the use of reductions as a vehicle for representing hierarchical structure is that the phenomenon of prolongation entails functional equivalence: a pitch event at a higher structural level is considered functionally equivalent to the elaboration of that event at lower structural levels. This assumption is, after all, what enables the former to represent the latter.

However, to borrow some terminology from early grammatical theory [8], it is necessary to distinguish between constructions that are *endocentric* and those that are *exocentric*. An endocentric construction is one that has the same distribution as its head, that is, one that can occupy the same contexts in tonal music with the same effect. In other words, in endocentric constructions the head is functionally equivalent to the whole: it is possible to replace the whole construction with just the head and still be left with a functionally equivalent construction. Elements that are subordinate to the head in this way, called *modifiers*, are therefore optional elements of the construction, which can be omitted without changing the effect of the construction. In exocentric constructions, by comparison, the whole does not necessarily have the same distribution as its head. Although still subordinate to the head, some of the dependent elements, called *complements*, may nevertheless be obligatory in order for the construction to fulfil the relevant structural roles and to behave in that particular way. Thus, in exocentric constructions it is not possible to replace the whole construction with the head: the two are not functionally equivalent.

To take a trivial example, because the construction in Fig. 5a is endocentric, its reduction to a single tonic chord at the level above is unproblematic because the head and the whole are functionally equivalent. On the other hand, the construction in Fig. 5b is exocentric; as a result, its reduction to a single tonic chord at the level above would replace one type of constituent, a TA, with another, a T or TP. So even though the tonic is the head of the TA, the DP is a complement not a modifier, and thus it is not possible to replace the whole with the head.

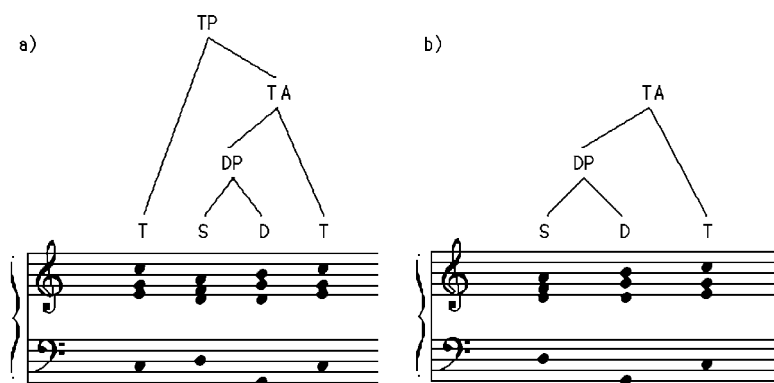


Fig. 5. Endocentric (5a) and exocentric (5b) harmonic structures

The Schenkerian analyst, however, doesn't have the luxury of worrying whether tonal constructions are endocentric or exocentric; the commitment to represent tonal structure in terms of linear formations means that complex harmonic structures *have* to be represented in terms of simpler configurations at higher structural levels.

Consequently, in order to derive the underlying voice-leading formations, Schenkerian theory is often required to treat all subordinate events as modifiers, by replacing each construction with its head.

Reductive representations thus confuse the notion of a constituent being headed, that is, consisting a head plus various other elements that are functionally dependent upon the head, with the notion of a constituent being endocentric, that is, the dependent elements being entirely optional. What constructions such as the ones examined above show is that *there are different types of dependency relation*, modification and complementation, and reduction is only an appropriate method of analysis for the former.⁵

4.3 Projection

Another aspect of syntactic structure that causes serious problems for reductive representations is how heads *project* onto the musical surface. The projection of a head at any particular structural level is just the string of elements (i.e. the constituent) that it dominates at that level.

Because a constituent structure representation describes the part-whole relations of an object, it clearly distinguishes the set of elements that constitute a particular head's projection from those that do not. This is a very important aspect of tonal structure: since it is the head that gives the constituent its peculiar functional properties, an account of constituent structure thus marks out the specific sequence of pitch events over which each head is functionally influential. In particular, because an account of constituent structure indicates unambiguously the extent of each constituent, it not only illustrates the fact that a piece consists of various functional regions, but, more importantly, *it explicitly encodes the boundaries of those functional regions*.

The atomism of reductive representations, however, makes the proper description of part-whole relations impossible because they are unable to recognise any concept of a 'whole': instead, at each level, the part has to represent the whole. As we've seen, Schenkerian analyses represent the group of events that elaborate a particular head in terms of the prolongation of the head itself. Accordingly, prolongational structure is assumed to be described by the spans of time subsumed by the voice-leading connections between heads. Lerdahl and Jackendoff [3] explain this as follows:

„Hierarchical relations among these connected or unconnected events are conveyed much the same way as in grouping analysis: slurs [between superordinate events] 'contain' other slurs...The secondary notation [that is, the musical notation illustrating the underlying voice-leading progressions]...easily conveys the 'prolongational groupings' inherent in prolongational reduction.“ ([3], p.202-3)

If a single event has to represent a whole constituent, however, an immediate problem is how to encode constituent boundaries. For if the large-scale voice leading

⁵ Various kinds of tonal constructions involve complementation, for instance, certain types of harmonic structures based upon third relations. To take one example, in constructions involving a descending chromatic third-divider between dominant and tonic, the mediant commonly elaborates the dominant as a complement (for example, bars 13-21 of the third movement of Beethoven's Violin Sonata, Op.24).

between constituent heads is used to demarcate the prolongational regions within a piece then, because voice-leading notation connects individual events, the only boundaries it can recognise are those that occur at the heads of constituents. Thus, the analyst either has to relinquish the idea that the linear formation at any structural level can be used to convey how the harmonic structure of a piece unfolds, or has to interpret the concept of prolongation in a literal sense, with the unfortunate consequence of implying that all tonal constituents are left-headed.

Indeed, it is common for the boundaries of the constituent parts of a tonal composition not to be coextensive with those regions marked out by connecting the heads of those constituents together. Thus, any analytic technique that relies upon representing the tonal structure of a composition in terms of a framework built out of its most important voice-leading events will inevitably misrepresent the structure's projectivity.

The C Major Prelude from Book I of Bach's *The Well-tempered Clavier* (1722) provides a good demonstration of this as many of the underlying voice-leading events occur at the end of the sequences of events that elaborate them. Fig. 6 illustrates the most significant large-scale linear process of the piece; Fig. 7 provides a representation of its constituent structure at the highest levels of structure. The roman numerals directly beneath the staff in Fig. 7 (labelled 'CS') show how the heads of the constituents at the relevant structural level project onto the musical surface. Further below is a Schenkerian reading (labelled 'SR') by Cadwallader and Gagné ([9], p.66), which, by contrast, interprets the various stages of the harmonic progression as coinciding with the equivalent stages in the voice-leading process, thus giving a distorted impression of the work's tonal structure.

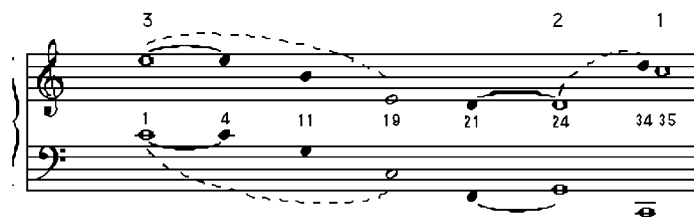


Fig. 6. The middle- and large-scale linear processes underlying the C major Prelude from Book I of *The Well-tempered Clavier* by J.S. Bach

In summary, the fact that the boundaries of tonal constituents are rarely coextensive with the real-time location of the underlying voice-leading events means that the concept of prolongation is a relatively impoverished theoretical device for describing part-whole relations.

5 Conclusion

Some structural phenomena have been examined that pose problems for Schenkerian representations of tonal structure. These problems arise from the dual role non-terminal nodes are required to play in Schenkerian representations, serving both the description of linear structure and the description of tonal structure. Elsewhere

(Horton [10]), I have argued that these two dimensions of structure require theories of a different form and that their conflation within the same system has adverse consequences for the description of both.

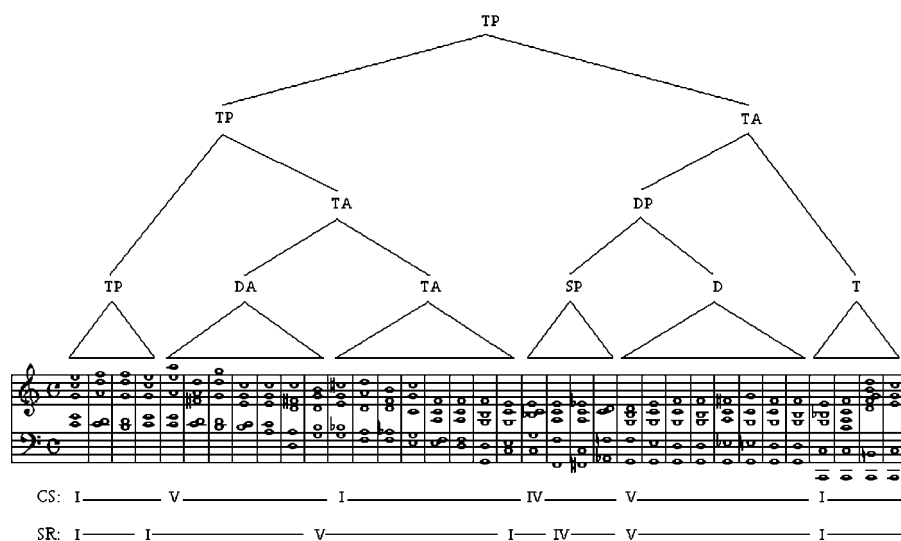


Fig. 7. A constituent structure representation of the middle- and large-scale structural levels of the C major Prelude. The first set of Roman numerals beneath the staff (CS) illustrates (for an intermediate level of structure) the projection of heads onto the musical surface as entailed by an account of the composition's constituent structure; the second set (SR) illustrates (for the same level of structure) a Schenkerian account of projectivity in terms of the prolongation of those heads

What is more, the phenomena investigated here not only seem to require the separation of linear and syntactic representations of musical structure, but also imply that a descriptively adequate tonal theory will have the form of a combinatorial syntax. In turn, this observation places important constraints upon the computational architecture that implements our cognition of tonal music; specifically, it must be one that is able to support the combinatorial algorithms involved in concatenating events into strings. Indeed, the fact that constituent parts play causally efficacious roles in music cognition implicates the existence of a computational architecture that not only is able to *encode* constituent relations between parts and wholes, but also *whose operation is sensitive* to the tokening of such parts and wholes.⁶

References

1. Ganassi, S.: *Opera Intitulata Fontegara*. (ed.) Peter, H. (trans.) Swainson, D. Robert Lienau Musikverlag, Vienna (1535/1959)

⁶ For a more detailed discussion of all of the topics covered in this paper, see Horton ([10], Chapters 7 and 8).

2. Keiler, A.: The Syntax of Prolongation, Part I. In *Theory Only* 3 3-27 (1977)
3. Lerdahl, F., Jackendoff, R.: *A Generative Theory of Tonal Music*. MIT Press, Cambridge, MA (1983)
4. Gaifman, H.: Dependency Systems and Phrase-Structure Systems. *Information and Control* 8 304-337 (1965)
5. Schenker, H.: *Free Composition*. (ed.) E. Oster. Longman, New York (1935/1979)
6. Schenker, H.: *Sonate A dur, Op. 101: Kritische Einführung und Erläuterung*. Beethoven: *Die letzten Sonaten*, (ed.) O. Jonas. Universal Edition, Vienna (1920/1972)
7. Salzer, F.: *Structural Hearing: Tonal Coherence in Music*. (2 vols.) Charles Boni, New York (1952)
8. Bloomfield, L.: *Language*. Holt, New York (1933)
9. Cadwallader, A.C., Gagné, D.: *Analysis of Tonal Music: A Schenkerian Approach*. Oxford University Press, Oxford (1998)
10. Horton, T. J.: *The Formal Structure of Tonal Theory*. Ph.D thesis, Faculty of Music, University of Cambridge (2002)

Respiration Reflecting Musical Expression: Analysis of Respiration during Musical Performance by Inductive Logic Programming

Soh Igarashi¹, Tomonobu Ozaki¹, and Koichi Furukawa²

¹ Keio Research Institute at SFC

² Graduate School of Media and Governance, Keio University
5322 Endo, Fujisawa, Kanagawa, 252-8520, Japan
{soh,tozaki,furukawa}@sfc.keio.ac.jp
<http://bruch.sfc.keio.ac.jp/>

Abstract. In this paper, we describe the analysis of respiration during musical performance by Inductive Logic Programming (ILP). For effective musical performance, it is one of the most important factors to control one's respiration in response to the aspects of the music performed. It is, however, often difficult even for experts to explain how to do so clearly. We measured respiration during cello performance by using a respiration sensor, and tried to extract rules of respiration from the data together with musical/performance background knowledge such as harmonic progression and bowing direction. As a result it was found that there was repeatability and regularity in performers' respiration pattern during musical performance, and then consistency in respiration with regard to musical structure was confirmed. It was also discovered that players tend to exhale at the beginning of new large musical structures, and inhale immediately before the change of keys.

1 Introduction

People often talk about “good” or “skilled” musical performance, and all players make every effort to play better. Generally, whether the performance is effective or not is judged by the audience listening to it. It is, however, difficult to explain the judgment criteria clearly. Furthermore, a convincing, concrete and scientific answer has not yet been discovered which answers the essential question, “how can an instrumentalist perform more effectively.” We know, however, from experience that there are some regularities or common features in effective musical performance. Expert performers are those who produce music rich with these features; they successfully navigate the unwritten constraints of musical performance[Furukawa00,Ueno01]. The problem which arises here is that experts normally are not conscious of satisfying such musical constraints. In many cases, skilled performers cannot explain what kinds of constraints we need to achieve effective musical performance. Therefore, if we can clearly verbalize these constraints, they should be very interesting and valuable for many people.

In this paper, we focus on respiration during cello performance as a constraint upon musical performance. The purpose of this research is to extract rules of

respiration from the data obtained by biomedical measurement together with musical/performance background knowledge by using Inductive Logic Programming (ILP)[Igarashi01].

2 Respiration Data Acquisition

Respiration during cello performance was measured by a belt-shaped respiration sensor. It senses the change in abdominal circumference which reflects the subject's breathing. The respiration data was sampled as a waveform, pre-processed by low pass filtering (cut off frequency being 1 Hz), and normalized. Figure 1 shows an example of a respiration waveform. This graph depicts a respiration curve. The horizontal axis represents elapsed time. Graduations on the horizontal axis indicate the corresponding number of measures of music. An ascending slope in Figure 1 indicates exhalation, and a descending slope indicates inhalation. The vertical axis depicts the voltages measured by the respiration sensor which can be interpreted as an approximate quantity of respiration. Therefore, a suddenly descending slope signifies rapid and deep inhalation, and a gently ascending slope signifies leisurely exhalation. A nearly constant slope indicates that the subject does not breathe heavily. A change in slope from ascending to descending indicates a change in respiration from exhalation to inhalation, while a change in slope from descending to ascending indicates a change from inhalation to exhalation.

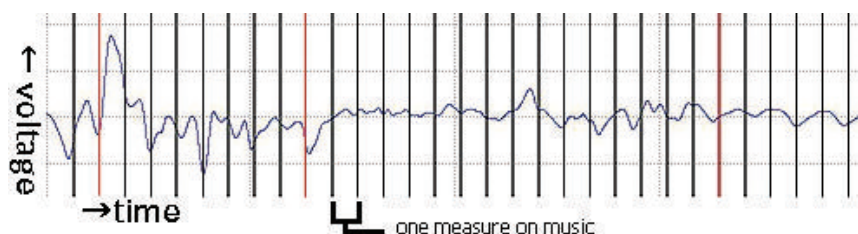


Fig. 1. A respiration curve during cello playing (example)

The subjects were four skilled cello players: students and graduates of a famous school of music in Japan. They consist of two males (subject A, B) and two females (subject C, D). Each subject has more than 10 years performance experience. Each cellist was asked to perform Luigi Boccherini's Rondo. We selected this work as it includes several passages which require the cellist to demonstrate a wide range of basic cello techniques and musical expressions. For the analysis, only the first half of the Rondo was utilized (see Figure 2). Each subject repeated the performance of this musical fragment six times, and six respiration curves were obtained for each subject. With regard to subject B, only five respiration curves were analyzed as one of the measurements had technical difficulties. For simplicity, each performer performed the piece with a uniform tempo (quarter note = 84).

3 Extracting Respiration Rules by ILP

Inductive Logic Programming (ILP)[Furukawa01] is a framework of inductive inference based on predicate logic. It is one of the most powerful methods of machine learning. ILP has been used previously for the analysis of music. [Dovey95] analyzed piano performances of works by Rachmaninoff using ILP, and determined some rules underlying them. [Baelen96] extended Dovey’s work in several directions, and attempted to discover regularities that can be used to generate MIDI (Musical Instrument Digital Interface) information derived from the musical analysis of a piece. [Widmer95,Widmer96] studied a method to learn rules of musical expression for piano performance at two different levels: the level of individual notes and that of musical structures.

3.1 Experiment 1 – Learning Respiration State

Problem Setting. During the initial step in the analysis by ILP, rule extraction of respiration states was conducted. The aim of this experiment was to learn such rules as would explain when performers exhale and inhale during musical performance. Since the representation language of ILP is predicate logic, it is difficult within an ILP framework to handle time sequential data, such as respiration curves. The data obtained as a waveform must be segmented into examples. How we conduct the segmentation is one of the most important issues in our research methodology.

Six (five in the case of subject B as mentioned above) respiration curves for each subject were analyzed. We chose to regard one beat of music as one example. As the experimental task consists of 67 measures totalling 134 beats, so each respiration curve is segmented into 134 examples (wave segments). For each $i \in [1..134]$, i -th wave segment of five (four in the case of subject B) respiration curves were averaged to build the i -th training example in the training data set. The remaining respiration curve was regarded as a test data set which contains 134 test examples.

Three classes were introduced as target concepts, *inhalation*, *exhalation*, and *no respiration*. One of the three classes was assigned to each example based upon the value difference between the first and the end point of the wave segment corresponding to the current musical beat. For respiration wave segments with an ascending slope, having a large positive value difference, class *exhalation* was assigned. For descending wave segments, having a large negative value difference, class *inhalation* was assigned. Respiration wave segments of nearly even shape, having a small value difference approaching zero, are assigned the remaining class *no respiration*. The value difference of 10 % of the peak magnitude of each respiration curve was used as a threshold.

This class assignment method has the limitation of mis-identifying examples which include both inhalation and exhalation as *no respiration*. In this study, this identifying error was ignored as the measured respiration curves indicated that most changes of respiration were synchronized with the onsets of beats. To discover a better class assignment method is a future task. Table 1 shows the number of examples within each data set associated with a particular class.

Table 1. The number of examples belonging to each class

the name of data set	<i>exhalation</i>	<i>inhalation</i>	<i>no</i>	total
subject A (training data set)	50	48	36	134
subject A (test data set)	52	53	29	134
subject B (training data set)	50	50	34	134
subject B (test data set)	62	52	20	134
subject C (training data set)	48	42	44	134
subject C (test data set)	60	50	24	134
subject D (training data set)	49	46	39	134
subject D (test data set)	47	47	40	134

As background knowledge, 15 kinds of predicates concerned with musical structure and playing styles were defined. These predicates are represented in Prolog and their interpretations are shown below:

<code>downbeat(X).</code>	... Beat X is a downbeat.
<code>upbeat(X).</code>	... Beat X is an upbeat.
<code>has_note(X,Y).</code>	... Beat X has Y notes.
<code>shape(X,Y).</code>	... The melodic line in beat X is in the shape of Y.
<code>dynamics(X,Y).</code>	... The dynamics of X are Y.
<code>harmonic_function(X,Y).</code>	... The harmonic function of beat X is Y.
<code>change_key_f(X,Y).</code>	... A key changes after beat X.
<code>change_key_l(X,Y).</code>	... A key changes before beat X.
<code>start_structure(X).</code>	... A musical structure starting with beat X.
<code>end_structure(X).</code>	... A musical structure ending with beat X.
<code>structure(X,Y).</code>	... Beat X is of structure Y.
<code>bowing(X,Y).</code>	... The bowing pattern in beat X is Y.
<code>string_transition(X)</code>	... Beat X includes a string transition.
<code>shift(X).</code>	... A left hand shift is done in beat X.

To represent the order of examples, a predicate named `nextto` was defined. `nextto(X,Y)` signifies that beat Y is next to X. An example of the real representation of an example in Prolog is shown as follows.

```

class(beat18,exhalation).      upbeat(beat18).
has_note(beat18,4).           shape(beat18,descending).
start_structure(beat18).       dynamics(beat18,piano).
harmonic_function(beat18,tonic). bowing(beat18,u).
string_transition(beat18).     nextto(beat17,beat18).
nextto(beat18,beat19).

```

Using the data set described above, induction by ILP was conducted. The ILP system used was FOIL6.4[Quinlan95]. This is a multiple class problem which

has multiple target concepts (classes). In this analysis each class was learned separately. This means that the multiple class problem was solved by dividing into several two class problems. On learning each class, positive examples of the other classes were regarded as negative examples of the class.

Results. On learning each class, many rules were obtained. The summary of our results is shown below before detailed discussion of our derived rules. Table 2 describes the summary of learned results from the training data set.

Table 2. Summary of learned results from training data set: Experiment 1

subject A	<i>ex</i>	<i>in</i>	<i>no</i>
positive coverage	98%	98%	89%
negative coverage	100%	100%	100%
accuracy	99%	99%	97%
subject B	<i>ex</i>	<i>in</i>	<i>no</i>
positive coverage	94%	84%	74%
negative coverage	100%	100%	98%
accuracy	98%	94%	92%
subject C	<i>ex</i>	<i>in</i>	<i>no</i>
positive coverage	98%	98%	89%
negative coverage	100%	100%	100%
accuracy	99%	99%	96%
subject D	<i>ex</i>	<i>in</i>	<i>no</i>
positive coverage	100%	85%	90%
negative coverage	100%	100%	96%
accuracy	100%	95%	94%

Positive coverage is defined as $\frac{p}{P}$, where P is the total number of positive examples and p is the number of positive examples covered by the rule(s). Negative coverage is defined as $\frac{N-n}{N}$, where N is the total number of negative examples and n is the number of negative examples covered by the rule(s). Accuracy is defined as $\frac{p+(N-n)}{P+N}$.

Table 2 clearly shows that nearly complete classification was achieved with the training data set, though there was not enough about the class *no* of subject B. To verify the prediction ability of these results, validation was performed by using the test data set. The result of the validation is shown in Figure 3.

Table 3 depicts the effectiveness of the rules obtained from the training data set to the test data set; for example, accuracy in Table 3 signifies what percentage of positive examples in the test data set are covered by the rules obtained from the training data set. To all results shown in Table 3 an χ^2 test was conducted, and statistical significance at the 1% significance level was confirmed except for the class *no* of some subjects. Default accuracy signifies the accuracy in the case that all positive examples are classified into the most frequent class (positive or negative in this case). Therefore it cannot be said that the derived rules are significant unless the accuracy exceeds the default value in predictive point of view. From this viewpoint, with respect to the classes *exhalation*, and *inhalation* the derived rules can be considered significant, but not for the class *no*. The weakness in class assignment about the class *no* described above can be considered as one of the reasons for this result.

We claim from what has been described thus far that there surely is some kind of regularity or pattern which should be learned in respiration during musical

Table 3. Results of validation: Experiment 1

subject A	<i>ex</i>	<i>in</i>	<i>no</i>
positive coverage	79%	74%	52%
negative coverage	88%	86%	86%
accuracy	84%	81%	78%
default accuracy	61%	60%	78%
subject B	<i>ex</i>	<i>in</i>	<i>no</i>
positive coverage	53%	54%	35%
negative coverage	79%	76%	86%
accuracy	67%	67%	78%
default accuracy	54%	61%	85%
subject C	<i>ex</i>	<i>in</i>	<i>no</i>
positive coverage	65%	68%	46%
negative coverage	84%	88%	74%
accuracy	75%	81%	69%
default accuracy	55%	63%	82%
subject D	<i>ex</i>	<i>in</i>	<i>no</i>
positive coverage	68%	51%	25%
negative coverage	79%	89%	78%
accuracy	75%	75%	62%
default accuracy	65%	65%	70%

performance. In the following section we discuss the features of such patterns, by examining the derived rules with regard to the classes *exhalation* and *inhalation*.

Discussion. First, the following rules which cover many of the examples were obtained. Though some predicates in the rules shown below are different in form from those described above, they have the same meanings. The number in square brackets which follows each rule indicates the number of examples covered by the rule.

```
class_exhalation(A) :- nextto(A,B), nextto(C,A),
                        class_inhalation(C), class_inhalation(B). [13]
```

“A player will *exhale* on a beat, when s/he *inhales* on both the following and previous beats.” obtained from subject A

```
class_inhalation(A) :- nextto(A,B), nextto(C,A),
                        class_exhalation(C), class_exhalation(B). [14]
```

“A player will *inhale* on a beat, when s/he *exhales* on both the following and previous beats.” obtained from subject C

Similar rules were obtained from all of the subjects. These rules signify that *exhalation* and *inhalation* are alternately repeated on a beat-by-beat basis. This is also confirmed by the measured respiration curves. It can be thought that these rules were obtained by allowing the other target concepts to be used as background knowledge when a target concept is learned. These are ordinary rules and not so significant as knowledge. However from another viewpoint we could say that these rules signify the validity of the knowledge representation we have employed, as this knowledge which can be easily discovered by humans was also discovered by ILP utilizing the predicate calculus knowledge representation.

Interesting rules such as the following were also obtained.

```
class_exhalation(A) :- structure_5_2(A). [2]
```

“A player will *exhale* on the beat which is the second beat of structure.5.”
obtained from subject A

```
class_inhalation(A) :- structure_1_3(A). [6]
```

“A player will *inhale* on the beat which is the third beat of structure.1.”
obtained from subject A

Many similar rules were obtained from all the subjects. These rules signify that the same kind of respiration is present in different musical beats (examples) that share the same musical structure. We consider musical structure to mean a kind of musical unit which consists of several measures. Figure 2 shows what part of music corresponds to each musical structure. This result suggests that there is consistency in respiration during musical performance with regard to musical structure. In addition to the rules described above, many rules which include predicates about musical structure were also obtained. Examples of those are given below.

```
class_exhalation(A) :- nextto(B,A), start_structure(A),  
                        harmonic_function_tonic(B). [10]
```

“A player will *exhale* on a beat, which marks the beginning of a musical structure and the harmonic function of the prior beat is tonic.” obtained from subject D

```
class_inhalation(A) :- nextto(A,B), nextto(C,A), class_exhalation(C),  
                        end_structure(B). [2]
```

“A player will *inhale* on a beat, when a musical structure ends with the next beat and s/he *exhales* on the prior beat.” obtained from subject C

We would like to focus on the following rule.

```
class_exhalation(A) :- nextto(B,A), shape_rest_note(B). [4]
```

“A player will *exhale* on a beat, where the prior beat is a musical rest.”
obtained from subject A and D

This rule does not seem especially interesting at first glance. It covers four examples, the measures marked “A” in Figure 2. The figure clearly shows that all the four beats covered by the rule are musically important parts that mark the beginning of large musical structures. The exact same rule was obtained from two subjects, subject A and D, and it was confirmed that the rule holds for other subjects as well. This fact strongly supports the validity of the derived rule that a player will exhale at the beginning of a new large musical structure.

On the other hand regarding class *inhalation*, the following interesting rule was obtained.

```
class_inhalation(A) :- change_key_f(A). [2]
```

The image shows a musical score for a piece titled "Allegretto". The score is written for a piano and features six specific musical structures highlighted with boxes and labeled "Structure 1" through "Structure 6".

- Structure 1:** Located at the beginning of the first staff, marked with a box.
- Structure 2:** Located in the second staff, marked with a box.
- Structure 3:** Located in the third staff, marked with a box.
- Structure 4:** Located in the fourth staff, marked with a box.
- Structure 5:** Located in the fifth staff, marked with a box.
- Structure 6:** Located in the sixth staff, marked with a box.

Below the score, there is a legend titled "Structure" with three symbols: A (a box), B (a bracket), and C (a box with a horizontal line). The score also includes various musical notations such as dynamics (p, f), articulation (accents), and phrasing slurs.

Fig. 2. The musical structures

“A player will *inhale* on a beat, when a key changes after the beat.”
obtained from subject A

There are two beats where a key changes after those, and it was found that 87.5% of such examples had class *inhalation* through all the subjects. Although the number of applicable examples is rather small, the rule seems to have strength. A change of key often signifies a change of musical character. Therefore it can be thought that respiration during musical performance is performed in response to the change of musical character.

3.2 Experiment 2 – Learning Breathing Points

Problem Setting. In the first experiment, we focused our attention on three states of respiration, *exhalation*, *inhalation*, and *no respiration*. As a result, information about the change of respiration state was ignored. A second experiment was conducted which accounted for the state change of respiration, particularly exhalation to inhalation. This time, only one class, *breathe*, indicating that a player breathed on the beat in question, was defined, and the class assignment was conducted manually. The examples which were not assigned to the class *breathe* were regarded as negative examples. The data set, background knowledge and other problem settings were the same as the first experiment. Table 4 shows the number of examples within each data set. In Table 4 the class *positive* indicates the class *breathe*.

Table 4. The number of examples each data set has

the name of data set	<i>positive</i>	<i>negative</i>	total
subject A (training data set)	45	89	134
subject A (test data set)	46	88	134
subject B (training data set)	40	94	134
subject B (test data set)	44	90	134
subject C (training data set)	40	94	134
subject C (test data set)	51	83	134
subject D (training data set)	46	88	134
subject D (test data set)	40	94	134

Results. First the summary of results is shown below before detailed discussion of each derived rule. Table 5 describes the summary of learned results from the training data set. The table shows that almost as good a classification of

Table 5. Summary of learned results from the training data set: Experiment 2

	subject A	subject B	subject C	subject D
positive coverage	78%	90%	88%	80%
negative coverage	100%	100%	100%	99%
accuracy	93%	97%	96%	96%

the training data set was accomplished as in the first experiment. To verify the prediction adequacy of these results, validation was performed by using the test data set. Results of the validation are shown in Table 6. It signifies the

Table 6. Results of validation: Experiment 2

	subject A	subject B	subject C	subject D
positive coverage	61%	57%	61%	80%
negative coverage	92%	86%	89%	85%
accuracy	81%	76%	78%	84%
default accuracy	66%	67%	62%	70%

effectiveness of the rules obtained from the training data set to the test data set, since the accuracy exceeds the default value. For each result shown in Table 6 an χ^2 test was conducted, and statistical significance at the 1% significance level was confirmed.

Again, our experimental findings indicate that there are clear respiratory patterns in musical performance. In the following section we discuss the features of these regularities by examining each of our derived rules respectively.

Discussion. As a notable result, rules like the following were obtained.

`breathe(A) :- structure_1_6(A). [2]`

“A player will *breathe* on a beat which is the sixth beat of structure_1.”
obtained from subject A and D

`breathe(A) :- structure_6_3(A). [2]`

“A player will *breathe* on a beat which is the third beat of structure_6.”
obtained from subject A and B

Both rules were independently obtained from two subjects, and many similar rules were obtained from all the subjects. These kinds of rules correspond to the results of the first experiment, and indicate that when a musical structure is repeated a player breathes similarly during the repetition. This result suggests that a player controls their respiration in response to the structure of the music being performed. Since to control respiration is nothing else but to design musical performance itself, it can be thought that this result reveals one of the constraints for skilled musical performance.

In addition to the rules described above, the following interesting rule was also obtained:

`breathe(A) :- change_key_f(A). [2]`

“A player will *breathe* on a beat, when a key changes after the beat.”
obtained from subject C

This rule corresponds to a result of the first experiment, and supports its validity.

4 Discussion on Phrasing

In the previous section an analysis of the state change of respiration was performed. In this section we would like to discuss musical phrasing. Generally speaking, the word phrase can be used in several different ways. In this paper, the length of a phrase is defined as the length between two successive points marking a change in respiration. For example, let us suppose that a player breathes on a beat P , and also on the third following beat; we claim a phrase of three beats length begins with beat P . Table 4 shows that the total number of breaths for each subject is nearly equal. However considering the lengths of phrases, differences among subjects are recognized. Table 7 shows the distribution with regard to the lengths of phrases for the training data set of each subject.

Table 7. The distribution with regard to the length of phrases

the length of phrase	1	2	3	4	5	6	7	8	total
subject A	0	2	17	11	12	1	1	0	44
subject B	1	9	17	8	2	1	0	2	40
subject C	0	19	14	3	3	1	2	1	43
subject D	5	18	6	12	3	2	0	0	46

Table 7 clearly shows that the subjects are rather different from one another in terms of phrasing. As a general trend we recognize that subjects C and D take shorter phrases than subjects A and B. This might be caused by gender difference. With respect to the phrasing of musical performance the following two characteristics were discovered by focusing on sections of music with especially long phrases – those lasting longer than four beats.

- There is a tendency to play one musical structure as one phrase.
- There is a tendency in a technically challenging part to prolong a phrase.

The first characteristic was observed from three subjects, subjects A, C, and D. Subject A, in particular took a phrase of 7 beats long exactly corresponding to the musical structure, the section marked “B” in Figure 2. A similar tendency was observed from subjects C and D.

The second characteristic was observed from subjects A, B, and C. The technically challenging part mentioned above is marked as section “C” in Figure 2. This observation seems to imply that in parts of technical difficulty a player does not tend to breathe often in order to avoid the influence of breathing upon body control. Therefore it can be thought that a player reduces the number of breaths by taking a long phrase in technically hard parts.

Although it is difficult to draw a clear conclusion because the data size was small, it can be thought from what has been described above that respiratory phrase analysis is one of the most promising approaches for revealing the constraints of skilled musical performance.

5 Conclusion

To summarize our findings, the following four major points can be considered:

1. Performers exhibit reproducible and repeatable respiration tendencies during musical performance.
2. There is consistency of respiration during musical performance with regard to musical structure.
3. Performers tend to exhale at the beginning of new large musical structures, and to inhale immediately before changes of key.
4. Performers tend to play one musical structure as one breathing phrase, and to take a long breathing phrase in technically hard parts.

The first point demonstrates consistency of respiration for each subject, and the others are the discovery of commonality among subjects. However, exactly the same rules were seldom obtained from multiple subjects. Yet, by interpreting the derived rules at a meta level, commonality was discovered. We will give an example to explain this meta level interpretation. As a result of the first experiment many rules relating to musical structure such as “`class_exhalation(A) :- structure_5.2(A).`” and “`class_inhalation(A) :- structure_1.3(A).`” were obtained. These rules are literally different from each other, but are common in signifying the correspondence of respiration to musical structure.

In conclusion, it can be said that by analyzing the respiration of performers during musical performance using ILP, some of the constraints concerning skilled musical performance were revealed.

6 Future Work

There are several directions for future research. First, performing descriptive learning seems to be promising. Since this research focused on classification, the derived results were not sufficient from the concept learning point of view. Employing descriptive learning seems to lead to an improvement with the readability of derived rules.

There is also room for improving data segmentation and class assignment methodologies. In this research, the respiration curves were divided equally, and one beat of music was regarded as one example. This led to the case of multiple classes included in one example. To discover more refined segmentation and class assignment methods is an important future task.

In addition to what has been described thus far, making a comparison between experts and novices and performing experiments utilizing different musical works of contrasting character may further enhance this research.

Acknowledgements. The authors would like to thank all of the performer subjects for their cooperating with our research, and Ken Ueno (Corporate Research & Development Center, Toshiba Corporation) for helping with the measurement of respiration. Finally the authors greatly thank Assistant Professor Christopher Thomas Penrose (Keio University) and Naoyo Tamagawa (Keio Research Institute at SFC) for their help with proof reading.

References

- [Baelen96] Van Baelen,E. and De Raedt,L.:Analysis and Prediction of Piano Performances using Inductive Logic Programming, *Proceedings of the 6th International Conference on Inductive Logic Programming*, pp.55-71, 1996.
- [Dovey95] Dovey,M.J.:*Analysis of Rachmaninoff's Piano Performances using Inductive Logic Programming*, Oxford University Computing Lab, 1995. An extended abstract was published in *Proceedings of the 8th European Conference on Machine Learning*, Lecture Notes in Artificial Intelligence, Springer Verlag, 1995.
- [Furukawa00] Furukawa,K., Ueno,K., and Bain,M.:Motor Skill as dynamic Constraint Satisfaction, *Proceedings of the 17th International Workshop on Machine Intelligence*, pp.25-28, 2000.
- [Furukawa01] Furukawa,K., Ozaki,T., and Ueno,K.:Inductive Logic Programming (in Japanese) , Kyoritsu shuppan, 2001.
- [Igarashi01] Igarashi,S., Ozaki,T., Ueno,K., and Furukawa,K.:Analysis of respiration during musical performances by ILP, the 11th International Conference on Inductive Logic Programming, Work-in-Progress Reports, pp.65-76, 2001.
- [Quinlan95] Quinlan,J.R., Cameron-Jones,R.M.:Induction of Logic Programs: FOIL and Related Systems, *New Generation Computing* Vol.13, pp.287-312, 1995.
- [Ueno01] Ueno,K., Furukawa,K. Bain,M.:Motor Skill as dynamic Constraint Satisfaction, *Electric Transaction of Artificial Intelligence (ETAI)*, Linkoping University Electronic Press, 2001.
- [Widmer95] Widmer,G.:Modelling the rational basis of musical expression, *Computer Music Journal*, Vol.19, No.2, pp.76-96, 1995.
- [Widmer96] Widmer,G.:Learning Expressive Performance: The Structure-Level Approach, *Journal of New Music Research*, Vol.25, No.2, pp.179-205, 1996.

Mimetic Development of Intonation

Eduardo Reck Miranda

Sony Computer Science Lab Paris
miranda@csl.sony.fr

Abstract. This paper presents a simulation where a society of autonomous agents evolves a common repertoire of intonations from scratch by interacting with one another. We demonstrate by means of an example the role of social bonding for the evolution of intonation in a virtual society of simple agents furnished with a vocal synthesiser, a hearing apparatus and a simple brain capable of associating *auditory information* to *motor control* (in this case in terms of vocal synthesis control). We begin the paper with a succinct commentary on the motivation for this research, our objectives and the methodology for its realisation. Then we state the objective of the particular simulation introduced in this paper, followed by an explanation of its design and functioning, and an assessment of its results. The paper concludes with a short discussion on the importance of this research for music and its contribution to the advancement of Artificial Intelligence.

1 Introduction

We are interested in studying the origins and evolution of music and language by means of computer and/or robotic simulations, whereby musical or linguistic forms may originate and evolve. This paper presents a simulation where a society of autonomous agents evolves common repertoires of intonations. These repertoires could be considered either in the context of music or language evolution, depending on whether one views them as the basis for the formation of *musical melodies* or *speech prosodies*. In this paper we will consider them in the context of music.

The quest for the origins of music is not new; philosophers of all times have addressed this problem. As an example, we cite the book "Music and the Origins of Language", by Downing Thomas (1995) as an excellent review of the theories proposed by the philosophers of the Enlightenment. And more recently, "The Origins of Music", edited by Nils Wallin, Bjorn Merker and Steven Brown (2000), collates a series of chapters written by top contemporary musicologists. With the exception of one chapter (Todd, 2000), however, none of these thinkers sought theoretical validation through computer models and simulation. Although we are well aware that musicology does not need such support to make sense, we do believe that models and simulation can be useful to develop and demonstrate specific theories of music.

We are interested in developing a theoretical framework for studying the origins and evolution of music based upon the notion that music is an adaptive complex

dynamic system. In simple terms, adaptive complex dynamic systems emerge from the overall behaviour of interacting autonomous elements in a non-hierarchical manner; that is, there is no central control driving these interactions. In our case, these individual elements are software agents geared to interact co-operatively with one another, under specific environmental conditions. We are particularly interested in establishing the fundamental properties and mechanisms that these autonomous elements, the interactions and the environment, must possess in order to create music. The rationale here is that if we furnish the agents with proper cognitive and physical models, combined with appropriate interaction dynamics and adequate environmental conditions, then the society should be able to evolve realistic musical cultures. The questions that we are interested in addressing in this research are: What are the appropriate agent interactions and what motivates them? What are the proper cognitive and physical abilities of these agents? What environmental constraints are necessary to foster music evolution?

This paper introduces a model whereby a small society of interactive agents furnished with appropriate motor, auditory and cognitive skills can evolve a shared repertoire of intonations from scratch, after a period of spontaneous creation, adjustment and memory reinforcement. By way of related research we cite the work of Todd and Werner (1999) and Dahlstedt and Nordhal (2001). The model described below shares many features with the language games proposed by researchers in the field of robotics and linguistics at Sony CSL and the Free University of Brussels (Steels, 1997; de Boer, 2000).

2 The Agents

The motivation of the agents is to form a repertoire of intonations in their memories and foster social bonding. In order to be sociable, an agent must form a repertoire of intonations that is similar to the repertoire of its peers. Sociability is therefore assessed in terms of the similarity of the agents' repertoires. In addition to the ability to produce and hear sounds, the agents have a basic instinct: to *imitate* what they hear. Before we clarify what we mean by imitation, let us get acquainted with the anatomy of the agents and their perceptual representation of intonation.

The agents are equipped with a voice synthesiser, a hearing apparatus and a simple brain featuring an associative memory and an 'innate' enacting script (the enacting script will be introduced in section 4). The voice synthesiser is essentially implemented as an articulator model of the human vocal mechanism (Miranda, 2002). In short, the agents need to compute two parameters in order to produce a sound: $\phi(n)$ and $t(n)$, where $\phi(n)$ controls the pitch correlate of the sound and $t(n)$ gives its triggering time; n corresponds to the ordering of these parameters in a control sequence. As for the hearing apparatus, it employs short-term autocorrelation-based analysis to extract the pitch contour of a spoken signal. The algorithm features a parameter m that regulates the degree of attention of the hearing apparatus, by controlling the resolution of the short-term autocorrelation analysis (Miranda, 2001).

This resolution defines the sensitivity of the auditory perception of the agents. The agent's memory stores its sound repertoire and other data such as probabilities, thresholds and reinforcement parameters. In the present simulation, the memory holds values for the following parameters (their meaning will be clarified as we go along): P_a = creative willingness, P_b = forgetfulness disposition, H_r = bad history threshold, H^+ = success history threshold, E_r = erase threshold, R^+ = reinforcement threshold, J_r = jump threshold, D = motor deviation coefficient, L_{min} = minimum length, L_{max} = maximum length. Length refers to the number of target points (or notes) in an intonation. The agents process and store intonation in terms of synthesis and analysis parameters. They have a dual representation of intonation in their memories: a *motor map* and a *perceptual map*. The motor map represents intonation in terms of a function of synthesis parameters (Fig. 4) and the perceptual map represents intonations in terms of the CARMEC representation scheme.

3 CARMEC: Common Abstract Representation of Melodic Contour

An intonation pattern is represented in CARMEC as a graph whose vertices stand for *initial* (or *relative*) *pitch points* and *pitch movements*, and the edges represent a directional path. Whilst the first vertex must have one outbound edge, the last one must have only one incoming edge. All vertices in between must have one incoming and one outbound edge each. Vertices can be of two types, *initial pitch points* (referred to as *p-ini*) and *pitch movements* (referred to as *p-mov*) as follows: *p-ini* = {SM, SL, SH, SR} and *p-mov* = {JDB, JD, SDB, SD, R, SU, SUU, JU, JUU}, where: SM = start the intonation in the middle register; SL = start the intonation in the lower register; SH = start the intonation in the higher register; SR = start the intonation in a relative register; and JDB = jump down to the bottom; JD = simply jump down; SDB = step down to the bottom; SD = simply step down; R = remain at the same level; SU = simply step up; SUT = step up to the top; JU = simply jump up; JUT = jump up to the top. Pitch movements here are relative to each point in the intonation pattern and not only to the initial pitch.

An intonation will invariably start with a *p-ini*, followed by one or more *p-movs*. It is assumed that an intonation can start at three different voice registers: low (SL), middle (SM) and high (SH). Then, from this initial point the next pitch might jump or step up or down, and so forth (Fig. 1).

It is important to note that labels or absolute pitch values are not relevant here because CARMEC is intended to represent abstract melodic contours rather than a sequence of notes drawn from a specific tuning system. Fig. 2 shows an instantiation of the abstract representation in Fig. 1 using the standard Western classic music notation system. Note that there are other instantiations of this intonation pattern in Western classic music notation because CARMEC is more general than the classic notation.

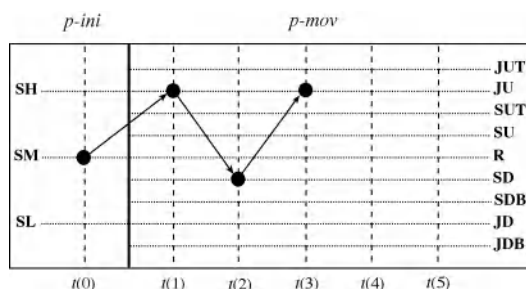


Fig. 1. The representation of an intonation pattern.



Fig. 2. An instantiation of the intonation in Fig. 1 using the Western classic music notation system.

4 The Basic Instinct: Imitation

We mentioned earlier that the agents have an instinct to *imitate* what they hear. Why imitation? The mimetic instinct is one of our most fundamental instincts. Life is animated with mutual interest between individuals who interact with one another to express reciprocity of intentions, co-ordination of skilful tasks, and so on. For example, Colwyn Trevarthen and co-workers observed that imitation of actions, roles and narrative display is typical of infants and toddlers long before they master language. Newborns are keen imitators from the first contacts they have with other people; e.g., (Trevarthen *et al.*, 1999) to cite but one of various references on imitation in babies. We seem to use imitations interactively and selectively to reciprocally motivate one another right from the start of our lives. Moreover, there is strong evidence that mimesis, or the ability to imitate the actions of other people, animals and nature, is a defining feature of the first evolutionary steps to human intelligence (Nadel and Butterworth, 1999). Our hypothesis is that mimetic interaction is one of the keys to bootstrap music in our society of agents.

Imitation is therefore defined as the task of hearing an intonation and activating the motor system to reproduce it. When we say that the agents should evolve a shared repertoire of intonations, we mean that the perceptual representation in the memory of the agents of the society should be identical, or at least close to each other, but the motor may be different. Fig. 3 shows an example where two different motor maps yield an identical perceptual representation of a certain intonation.

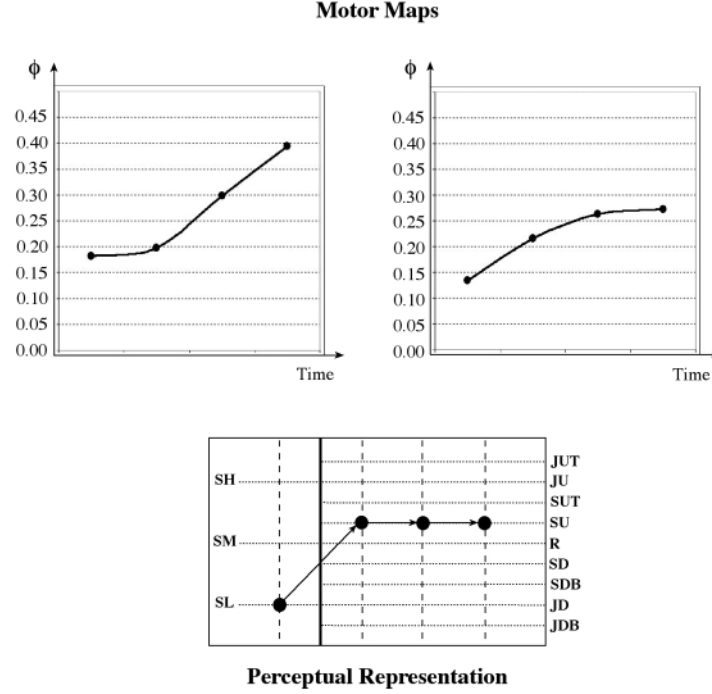


Fig. 3. An example where two different motor maps (top) yield the same perceptual representation (bottom). In this case, the degree of attention of the hearing apparatus was set to a medium resolution. ϕ is a function of controllers for the pitch of the synthesized sound; in this case it involves parameters for controlling lung pressure and the vocal folds.

5 The Enacting Script

The enacting script provides the agent with knowledge of how to behave during the interactions: the agent must know what to do when another agent produces an intonation, how to assess an imitation, when to remain quiet, and so forth. The enacting script does not evolve in the present model; all agents are alike in this respect. Also, all agents have identical synthesis and listening apparatuses. Please refer to the Appendix for the main algorithm of the enacting script. At each round, each of the agents in a pair from the society plays one of two different roles: the *agent-player* and the *agent-imitator*. The agent-player starts the interaction by producing an intonation p_a , randomly chosen from its repertoire. If its repertoire is empty, then it produces a random intonation. The agent-imitator then analyses the intonation p_a , searches for a similar intonation in its repertoire, i_n , and produces it. The agent-player in turn analyses the intonation i_n and compares it with all other intonations in its own repertoire. If its repertoire holds no other intonation p_n that is more perceptibly similar

to i_n than p_d is, then the agent-player replays p_d as a reassuring feedback for the agent-imitator; in this case the imitation would be acceptable. Conversely, if the agent-player finds another intonation p_n that is more perceptibly similar to i_n than p_d is, then the imitation is unsatisfactory and in this case the agent-player would halt the interaction without emitting the reassuring feedback; the agent-imitator realizes that no feedback means imitation failure. In all these cases, the agents always add a small deviation to the motor realization of the intonations they know; the amount of deviation is determined by the coefficient D (mentioned in section 2). If the agent-imitator hears reassuring feedback, then it will reinforce the existence of i_n in its repertoire and will change its perceptual parameters slightly in an attempt to make the melody even more similar to p_d (i.e., only if they are not already identical). In practice, the reinforcement is implemented as a counter that registers how many times an intonation has successfully been used.

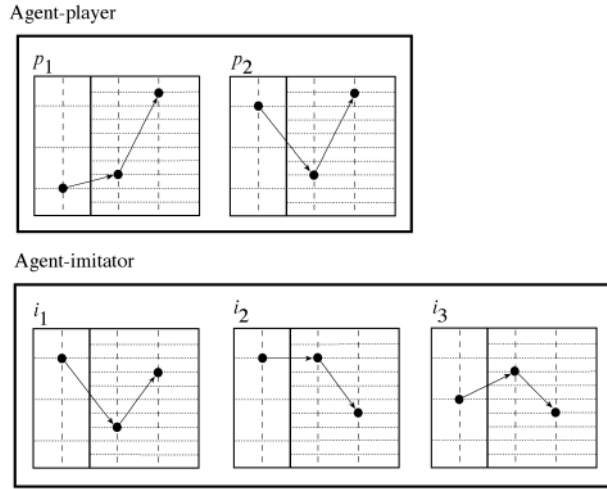


Fig. 4. A case where the agent-player has two intonations in its repertoire whereas the agent-imitator has three. If the agent-player produces p_1 , the agent-imitator will produce i_1 as an imitation because it is the most similar intonation to p_1 in its repertoire. Unfortunately the agent-player discovers that, as far as its repertoire is concerned, i_1 is more similar to p_2 than to p_1 . In this case the imitation is unsuccessful.

Conversely, if the agent-imitator does not receive the feedback then it will infer that something went wrong with its imitation. In this case, the agent has to choose between two potential courses of action. If it finds out that i_n is a weak intonation (i.e., bad past success rate) in its memory, because it has not received enough reinforcement in the past, then it will try to modify its motor representation of i_n slightly, in an attempt to further approximate it to p_n . It is hoped that this approximation will give the intonation a better chance of success if it is used again in another round. This motor

approximation is done by slightly increasing and/or decreasing all motor parameters; the algorithm keeps the set of changes that produced the best perceptual match. But if i_n is a strong intonation (i.e., good past success rate), then the agent will leave i_n untouched (because it has been successfully used in previous imitations and a few other agents in the society also probably know it), create a new intonation that is similar to p_d and include it in its repertoire. In case of failure, no reinforcement is applied to i_n . Before terminating the round, both agents perform final updates. Firstly they scan their repertoire and merge those intonations that are considered to be perceptibly identical to each other. Also, at the end of each round, both agents have a certain probability P_b of undertaking a spring-cleaning to get rid of weak intonations; those intonations that have not been sufficiently reinforced are forgotten. Finally, at the end of each round, the agent-imitator has a certain probability P_a of adding a new, randomly created intonation to its repertoire (Fig. 4).

Bear in mind that an important presupposition in this model is that the action of singing intonations involves the activation of certain vocal motor mechanisms in specific ways. The recognition of intonations here thus requires knowledge of the activation of the right motor sequence in order to reproduce the intonation in question. It does not matter, however, if one agent uses different values to its peers in order to produce identical intonations.

6 A Sample Case Study

In order to evaluate and discuss the results of our simulations, let us consider a simple case study whereby a society of 5 agents performed 5000 interactions, with the following settings: P_a = probability of 0.5%; P_b = probability of 20%; H_t = 0.1% of total amount of interactions of the round; H^+ = 90% of interactions so far; E_t = 1% of total amount of interactions of the round; R^+ = 95% of interactions so far; J_t = 0.25 in a scale from 0.0 to 1.0; D = $\pm 1\%$; L_{min} = 3 and L_{max} = 3. That is, there is a 0.5% probability that an agent will insert a new random intonation in its repertoire after an interaction and a 20% probability that this agent will scan its repertoire in order to delete intonations that did not score well so far: if an intonation has been used for more than E_t times and has scored less than 90% of the interactions so far (H^+), then it is deleted. In the case of an unsuccessful imitation, the agent-imitator checks whether the failed intonation has been already used at least H_t times and scored well for at least 95% of the interactions so far (R^+); if not, then the agent will attempt to correct the pattern. In a scale from 0.0 to 1.0, a pitch interval that is higher than 0.25 is considered as a *jump*, otherwise it is a *step*. Finally, the motor deviation is set to $\pm 1\%$ of its current value and the length of the evolved melodies is fixed to 3 pitches.

The graph in Fig. 5 shows the evolution of the average repertoire of the society, with snapshots taken after every 100 interactions. The agents quickly increase their repertoire to an average of between six and eight intonations per agent. At about 4000 interactions, more intonations appear, but at a lower rate. Identical behaviour has been observed in many such simulations with varied settings. The general tendency is to quickly settle into a repertoire of a certain size, which occasionally increases at lower

rates. The pressure to increase the repertoire is mostly due to the creativity willingness parameter combined with the rate of new inclusions due to imitation failures. In this case the repertoire settled to 8 intonations between 1600 and 4000 interactions.

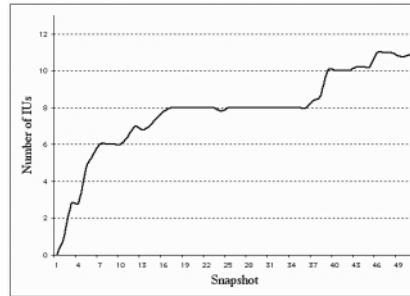


Fig. 5. The evolution of the average size of the repertoire of the whole society.

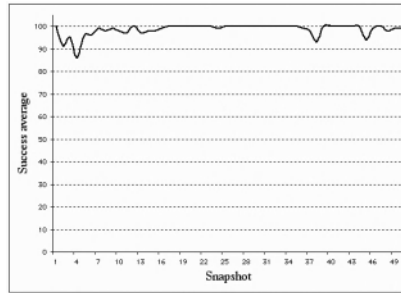


Fig. 6. The imitation success rate over time.

The graph in Fig. 6 plots the imitation success rate of the society, measured at every 100 interactions. Notice that the success rate drops within the first 1000 interactions, which coincides with the steep rising curve of the graph in Figure 6. This is the period in which the agents are negotiating how their repertoires should look in order to foster communication; this period is characterized by inclusions of intonations due to imitation failure and by motor adjustments due to imitation successes. At approximately 1800 interactions, the imitation rate goes back up to 100%. Then, occasional periods of lower success rate occur due to the appearance of new random intonations or eventual motor-perceptual inconsistencies that might be caused by pattern approximations. Although the repertoire tends to increase with time, the success rate is maintained consistently high. This is good evidence that the society does manage to foster social bonding. How about the other goal of the agents? Did they evolve a shared repertoire of intonations?

The answer is yes. Figs. 7(a) and 7(b) portray the perceptual memory of two agents randomly selected from the society, after 5000 interactions. The repertoire of all five agents plotted on top of each other is shown in Fig. 7(c). Fig. 7(c) demonstrates that at least two agents (the ones whose memories are plotted in Figures 7(a) and 7(b)) share

identical intonations with the whole society. Figs. 8(a) and 8(b) plot the motor maps of two agents, also randomly selected from the society. Although both maps give an overall impression of similarity, the actual curves are slightly different. A better assessment can be made if both maps are plotted on the top of each other, as shown in Fig. 8(c). This figure is a concrete example of how two different motor maps can yield identical perceptual representations; the two agents in question, randomly selected, have identical perceptual representations.

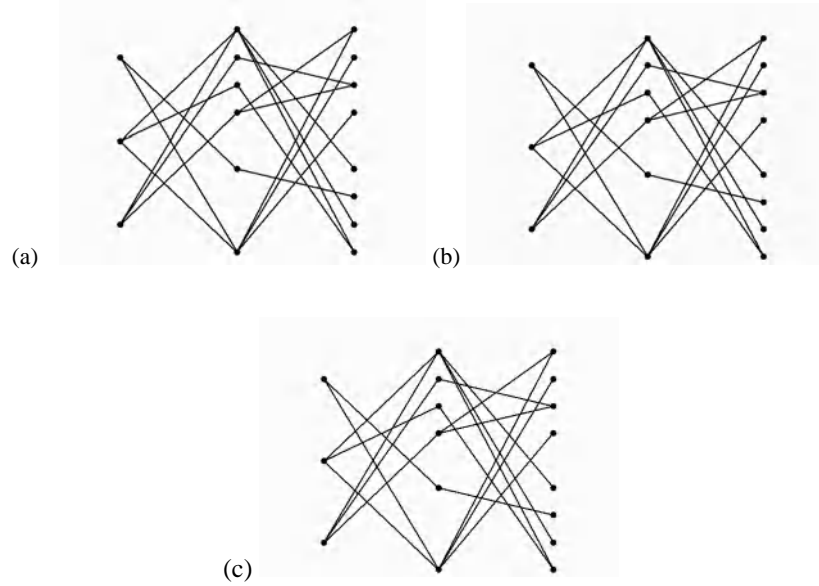


Fig. 7. The perceptual memory of the agents. Note that 7(c) is equal to 7(a) and 7(b), suggesting that the repertoire of the 5 agents is identical. For the sake of clarity, the background metrics and labels of the graph are not shown; refer to Figure 1 if necessary.

7 Discussion and Conclusion

This paper demonstrates that it is possible to evolve a shared repertoire of musical intonations in a society of simple mimetic software agents. What makes this model interesting is the fact that there is no global memory holding the repertoire of the society. The agents are autonomous individuals, each of them has its own memory and, most importantly, no other agent can see what is stored in there. The only way to access the sound repertoire of a society is by considering the memories of all agents put together; e.g., Fig. 7c. The repertoire of intonations emerges from the interactions of the agents and there is no global procedure supervising or regulating them; the actions of each agent are solely based upon their own local rules.

What is the contribution of this research to music? AI models have been extremely useful for designing systems to aid musical creativity or for solving Music Technology problems in a number of domains (Balaban *et al.*, 1992; Miranda, 2000). With very few exceptions, however, AI has not been used to aid musicological research, especially in emerging areas such as Evolutionary Musicology, where AI models can be of great help to address questions about the origins and evolution of music. Research in these areas still is incipient and much work needs to be done in order to attain the standards of scholarly research. Models such as the one described in this paper have the potential to strengthen such research, but it is up to musicologists and scientists alike to embrace the challenge.

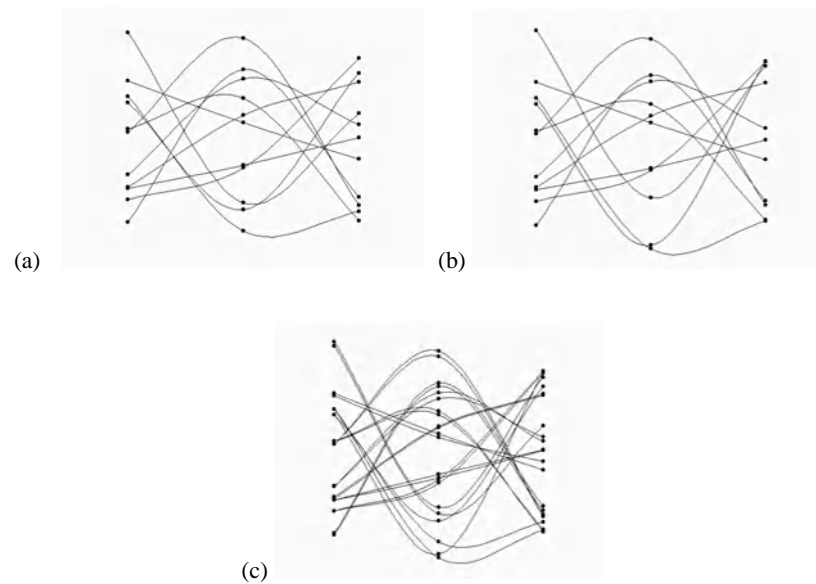


Fig. 8. The motor memories of the agents. Note that 8(a) and 8(b) are slightly different, suggesting that the agents developed different motor controls for identical intonations. For the sake of clarity the figure omits background metrics and labelling; only curves of the motor functions are displayed.

As for the contribution of this work to the advancement of AI, this paper suggests a less orthodox, but nevertheless efficient, approach to machine learning, and proposes an alternative solution to handling the problem of sensory-motor mapping.

Nowadays it is very hard to question the fact that machines can learn. The great majority of current machine learning systems, however, function as a kind of smart sponge that passively absorbs data from the world. Then, an algorithm either classifies these data into different categories or captures a general behaviour, with a view to reproducing or recognising such data and/or behaviour when exposed to them (Mitchell, 1997). This paper introduces a different scenario, where knowledge

emerges as the result of the actions of various machines in a social context. Knowledge here results from an emergent culture that is entirely established by interacting autonomous software agents. In simpler terms, instead of using a standard machine-learning algorithm to learn specific styles of music or compositional processes from given examples, we propose a system where artificial musicians actively evolve their own musical culture by interacting with one other. Moreover, there is no teacher telling these musicians what they have to learn.

With respect to the sensory-motor problem, imagine the following scenario: a pianist plays a melody on the piano and a violinist replays this melody on a violin. This trivial task poses a very difficult AI question: How can the violinist predict her movements on the violin in order to replay the melody that she heard on the piano? In other words, how does our brain associate sensory information with motor control? Suppose that the violinist is a robot who must learn how to accomplish this apparently simple task. The standard way of doing this would be to program the robot to learn the behaviour to be achieved, no matter whether or not the underlying mechanism reflects the way our brain works. This simplifies the problem considerably, but still there is the fundamental problem that there is no one-to-one mapping between perception and production in music. We propose an adaptive mechanism to address this problem. For the sake of simplicity, in this paper we force a one-to-one mapping, but it is clear that the agents learn by themselves how to correlate perception parameters with production ones. It would not matter if there were more motor controls than perceptual parameters, or vice-versa.

References

- Balaban, M., Ebcioglu, K. and Laske, O. (Eds.) (1992). *Understanding Music with AI: Perspectives on Music Cognition*. Cambridge, MA: The MIT Press.
- Boer, B. de (2000). Emergence of vowel systems through self-organisation. *AI Communications*, 13, 27-39.
- Dahlstedt, P. and Nordhal, M. G. (2001). Living Melodies: Coevolution of Sonic Communication. *Leonardo*, 34:3, 243-248.
- Miranda, E. R. (Ed.) (2000). *Readings in Music and Artificial Intelligence*. Amsterdam, The Netherlands: Harwood Academic Publishers.
- Miranda, E. R. (2001). Synthesising Prosody with Variable Resolution. *AES Convention Paper 5332*. New York, NY: Audio Engineering Society, Inc.
- Miranda, E. R. (2002). *Computer Sound Design: Synthesis Techniques and Programming*. Oxford, UK: Focal Press. (2nd edition)
- Mitchell, T. M., *Machine Learning*. New York, NY: McGraw Hill.
- Nadel, J. and Butterworth, G. (Eds.) (1999). *Imitation in Infancy*. Cambridge, UK: Cambridge University Press.
- Steels, L. (1997). The Synthetic Modelling of Language Origins. *Evolution of Communication Journal*, 1:1, 1-34.
- Thomas, D. A. (1995). *Music and the origins of language*. Cambridge, UK: Cambridge University Press.
- Todd, P. M. (2000). Simulating the evolution of musical behavior. In N. Wallin, B. Merker and S. Brown (Eds.), *The origins of music*. Cambridge, MA: MIT Press.

- Todd, P. M. and Werner, G. M. (1999). Frankensteinian Methods for Evolutionary Music Composition. In N. Griffith and P. M. Todd (Eds.), *Musical networks: Parallel distributed perception and performance* (pp. 313-339). Cambridge, MA: The MIT Press/Bradford Books.
- Trevarthen, C., Kokkinaki, T. and Fiamenghi Jr., G., A. (1999). What infants' imitations communicate: with mothers, with fathers and with peers. In J. Nadel and G. Butterworth (Eds.), *Imitation in Infancy* (pp. 127-174). Cambridge, UK: Cambridge University Press.
- Wallin, N. J., Merker, B. and Brown, S. (Eds.) (2000). *The Origins of Music*. Cambridge, MA: The MIT Press.

Appendix: The Main Algorithm of the Enacting Script

agent-player - AP	agent-imitator - AI
{ IF <i>repertoire(AP)</i> not empty pick motor control for p_d ; produce p_d ; ELSE generate random motor control for p_d ; add p_d to <i>repertoire(AP)</i> ; produce p_d ; }	{ analyse p_d } { build perceptual representation; } { IF <i>rep(AI)</i> not empty i_n = most perceptually similar to p_d ; ELSE generate random motor control for i_n ; add i_n to <i>repertoire(AI)</i> ; produce i_n ; }
{ analyse i_n ; } { build perceptual representation; } { p_n = most perceptually similar to i_n ; } { IF $p_n = p_d$ send <u>positive</u> feedback to AI; reinforce p_d in <i>repertoire(AP)</i> ; ELSE send <u>negative</u> feedback to AI; }	{ IF feedback = <u>positive</u> approximate i_n to p_d perceptually; generate appropriate motor control; reinforce i_n in <i>repertoire(AI)</i> ; } { IF feedback = negative IF i_n scores good H_r ; execute <i>add_new_similar(snd)</i> ; ELSE Modify motor representation of i_n towards p_d ; }
{ execute <i>final_updates(AP)</i> ; }	{ execute <i>final_updates(AI)</i> ; }

The *add_new_similar()* function works as follows: the agent produces a number of random intonations and then it picks the one that is perceptually most similar p_d to include in the repertoire.

Interacting with a Musical Learning System: The Continuator

François Pachet

SONY CSL-Paris,
6, rue Amyot, 75005 Paris, France
<http://www.csl.sony.fr/~pachet>

Abstract. The Continuator system is an attempt to bridge the gap between two classes of traditionally incompatible musical systems: 1) interactive musical systems, limited in their ability to generate stylistically consistent material, and 2) music composition systems, which are fundamentally not interactive. The purpose of Continuator is to extend the technical ability of musicians with stylistically consistent, automatically learnt musical material. This requires the ability for the system to build operational representations of musical styles in real time, and to adapt quickly to external musical information. The Continuator is based on a Markov model of musical styles augmented to account for efficient real time learning of musical styles and to arbitrary external bias. The paper describes the main technical issues at stake concerning the integration of an agnostic learning scheme in an interactive instrument, and reports on real-world experiments performed with various musicians.

1 Introduction

“I don’t play piano, I play pianist” the French Pop singer Claude Nougaro used to say, pointing to his long time accompanist Maurice Vander. This joke summarizes well the goal of the Continuator project. We seek to design musical instruments that provide exciting interactions through the ability to learn automatically arbitrary musical styles, and adapt these styles to the playing modes of the musician. The undertaking can be seen as a way to turn musical instruments from passive objects into active, autonomous systems, with which one can interact using high-level controls, much in the same way Claude Nougaro, through the blink of an eye, can control, or influence his pianist Maurice Vander.

Musical performance has been the object of numerous studies using all the software technologies at hand. In our context, we can divide these approaches into two categories: *interactive systems* and intelligent *music composition* systems. Interactive music systems propose ways of transforming in real time musical input into musical output. Musical interactive systems have been popular both in the experimental field [16] [19] as well as in commercial applications, from one-touch chords of arranger systems to the recent and popular Korg Karma synthesizer [9]. While a lot of work has been devoted to efficient controllers and interfaces for musical systems [4], [11], these systems all share a common drawback: they do not manage time, there is no memory of the past, and consequently the music generated is

strongly correlated with musical input, but poorly or not at all with a consistent and realistic musical style.

On the other hand, music composition systems precisely aim at representing stylistic information, to generate music in various styles, from the Illiac suite [8] to the automatic compositions of [6]. More recently, constraint techniques have been used to produce stylistically consistent 4-part Baroque music (see [13] for a survey). In the domain of popular music, prototypes such as [2], [3], [15] have demonstrated the technical feasibility of convincingly simulating jazz styles by computer. In opposition to interactive music systems, the main drawback of these approaches is that they do not allow real musical interaction: they propose fully-fledged automata that may produce realistic music, but cannot be used as actual instruments. Moreover, these approaches require explicit, symbolic information to be fed to the system, such as human input for supervised learning, underlying harmonic structure, tempo, song structure, etc.

The system we present here is an attempt to combine both worlds: to design real-time interactive musical instruments that are able to produce stylistically consistent music. More precisely, we propose a system in which musical styles are learned automatically, in an agnostic manner, and therefore which do not require any symbolic information (style, harmonic grid, tempo). The system is seamlessly integrated in the playing mode of the musician, as opposed to traditional fully automatic or question/answer systems, and adapts quickly and without human intervention to unexpected changes in rhythm, harmony or style. Finally, the design of the system allows the sharing of stylistic patterns in real time and constitutes in this sense a novel form of collaborative musical instrument.

The remaining of the paper is structured as follows. First we describe the heart of the system, based on a Markov model of musical styles, augmented with a hierarchy of learning functions to adapt to imprecision in musical inputs. Then we discuss the issues related to turning the learning facility into an actual musical instrument. Finally we report on experiments with the system in various real world musical contexts.

2 Inside the Continuator

In the standard mode, the system receives musical Midi input from one musician. The output of the system is itself sent to a Midi synthesizer and then to a sound reproduction system. The system acts basically as a sequence continuator: the note stream of the musician is continuously segmented into musical phrases. Each phrase is sent asynchronously to a phrase analyzer, which builds up a model of recurring patterns. In reaction to the played phrase, the system immediately generates a continuation, according to the database of patterns already learnt.

2.1 A Hierarchical Markov Model of Musical Sequences

Researchers in AI and information theory have long addressed the technical issue of learning automatically and in an agnostic manner a musical style. Shannon introduced in his seminal 1948 paper the concept of information based on probability of occurrence of messages. This notion was quickly used to model musical styles, and these experiments showed that it was possible to create pieces of music that would

sound like given styles, by simply computing and exploiting probabilities of note transitions. More precisely, given a corpus of music material (typically music scores, or MIDI files), the basic idea is to represent in some way the local patterns found in the learnt corpus, by computing transition probabilities between successive notes. New music sequences are then generated using these probabilities, and these sequences will contain, by definition, the patterns identified in the learnt corpus.

One of the most spectacular applications of Markov chains to music is probably by David Cope [6], although his musical productions are not entirely produced automatically. A good survey of state-of-the-art of Markov based techniques for music can be found in [18], and a recent development in [1] and [10].

This work shows clearly two things: 1) Markov chain models and their extensions are powerful enough to represent efficiently musical patterns, but 2) their generative power is limited due to the absence of long-term information. In another words, these models can fool the listener on a short scale, but not for complete pieces.

Using Markov models for interaction purposes, and not for composing complete, fully-fledged musical pieces, allow us to benefit from 1) while avoiding the drawback of 2). The responsibility for organizing the piece, deciding its high-level structure, is left to the musician. The system only "fills in the gaps", and the power of Markov chains can be exploited fully to this aim.

The Continuator system is yet another species in the world of musical Markov systems, although with novel features. In our context, we want to learn and imitate musical styles in a faithful and efficient manner, and make the resulting mechanism usable as an actual music instrument. This raised a number of technical issues, whose solutions were integrated in the Continuator. These issues are addressed in the following section.

2.2 Hierarchical Markov Models

The learning module we propose systematically learns all phrases played by the musician, and builds progressively a database of patterns detected in the input sequences. We designed an indexing scheme which represents all the subsequences found in the corpus, in such a way that the computation of continuations is complete and as efficient as possible.

This technique consists in building a prefix tree by a simple, linear analysis of each input sequence. Each time a sequence is input to the system, it is stored in memory, and all subsequences encountered are systematically added to the tree. A construction of a similar complete indexing scheme for musical phrases is described in [10]. Our scheme is designed for optimizing efficiency. For reasons of space, we describe here only the most generic functions.

For instance, let us suppose the following input sequences:

Sequence #1: {A B C D}

and later:

Sequence 2#: {A B B C}

The system will build a tree containing, for all possible subsequences of each of these two sequences, the list of all continuations encountered in this learning corpus, and weighted by their number of occurrences. The scheme is called variable-order Markov chain after [16] because it contains the continuations for all subsequences of any length (up to a given maximum, typically 10). In our example, a subsequence

such as {B} has the following possible continuations: C (from sequence #1), and B (from sequence #2).

A subsequence such as {A B} has continuations: C (from sequence #1) and B (from sequence #2).

A subsequence such as {B B C} has only as possible continuation D from sequence #1. Note that in this last case, there is no continuation for the whole subsequence {B B C}, so we get the continuation for the longest possible subsequence, here, {B C}. When several continuations are similar, they are all repeated. For instance, the continuations of {A} are {B, B} (from sequences #1 and #2 respectively).

In our context, an important characteristic of the data structure we propose is that the sequence learned is not the input sequence itself. Indeed, Midi sequences have many parameters, not all of which are necessarily interesting to learn.

For instance, a note has attributes such as pitch, velocity, duration, start time, and possibly other information provided by continuous controllers (pitch bend, modulation, etc.). A chord has attributes such as the pitch list, possibly its root key, etc. The sequence learned is therefore not the input sequence itself, but a sequence obtained by applying a *reduction function* to the original input sequence. The simplest function is the pitch. A more refined function is the combination of pitch and duration. [5] and [18] proposed different reduction functions (called viewpoints) for representing music in the context of musical analysis. Our experiments with real time interactive music led us to develop and implement such a library of reduction functions, including the ones mentioned in these works, as well as functions specially designed to take into account more realistic Jazz and classical styles. One of them is the *PitchRegion*, which is a simplification of pitch: instead of considering explicit pitches, we reduce pitches in regions, in practice by considering only *pitch / region_size*.

2.3 Generation

The generation phase consists then, given an initial input sequence played by the musician, in computing successively continuations, step by step using a tiling process. First a note item is computed for the input sequence. Then the input sequence augmented by this item is considered, and the next item is computed, etc. At each step, a continuation for the subsequence of the maximum length is found, which results in optimum consistency with regards to the learnt corpus.

An important improvement on classical Markov-based generation mechanisms is behavior of our algorithm when it encounters a phrase for which no continuation is found.

Suppose a model trained to learn the arpeggio in Figure 1:



Fig. 1. An arpeggio learnt by the Continuator.

Suppose that the reduction function is as precise as possible, say pitch, velocity and duration. Suppose now that the input sequence to continue is the following (Figure 2):



Fig. 2. An input sequence which does not match exactly with a subsequence in learnt corpus.

It is clear that any Markov model will consider that there is no continuation for this sequence, because there is no continuation for the last note of the input sequence (here, E flat). The models proposed so far would then draw a new note at random, and actually start a new sequence.

However, it is also clear intuitively, that a better solution, in such a case, is to shift the viewpoint. In our context, this corresponds to using a less refined reduction function. Let us consider for instance pitch regions of three notes instead of pitches.

The learnt sequence of Figure 2 is then reduced to:

{PR1 PR1 PR2 PR3 PR5}

The input sequence is itself reduced to

{PR1 PR1 PR2}

In this new model, there is a continuation for the input sequence {PR1 PR1 PR2}, which is PR3.

Because our model keeps track of the original input sequences (and not only their reductions), we can generate the note corresponding to PR3 in the learnt corpus, in our case G. Once the continuation has been found, the process is started again with the new sequence, using the more refined reduction function.

More precisely, we introduce a *hierarchy* of reduction functions, to be used in cases of failure. This hierarchy can be defined by the user. A typical hierarchy is:

- 1 – pitch * duration * velocity
- 2 – small pitch region * velocity
- 3 – small pitch regions
- 4 – large pitch regions

where the numbering indicates the order in which the functions are to be considered in cases of failure in the matching process.

The approach we propose allows to take into account inexact inputs, with a minimum cost. The complexity for retrieving the continuations for a given input sequence is indeed very small as it involves only walking through trees, without any search.

3 From an Automaton to a Musical Instrument

The learning module described in the preceding section is able to learn and generate music sequences that sound like the sequences in the learnt corpus. As such, it provides a powerful musical automaton able to imitate faithfully styles, but not an interactive musical instrument. This section describes the main design concepts that can be used to turn this style generator into an interactive musical instrument. This is achieved through several related constructs: 1) a feature remapping scheme by which generated phrases match as much as possible the input phrase, 2) a step-by step

generation of the music sequences achieved through a real time implementation of the generator, and 3) an extension of the Markovian generation process with a *fitness function* which takes into account characteristics of the input phrase.

3.1 Feature Remapping

The phrases generated by our Markov model take into account input phrases only superficially. As described above, the input phrase is considered as a potential match for new phrases, starting from its end. In the context of interactive systems, many important features of the input phrases are not considered. The reaction of the system may therefore sound too automatic. These features are notably the beat (or tempo), the dynamics (velocity for Midi streams), the phase (if there is a prominent beat), etc.

The generation scheme includes a feature analysis and remapping process, whose role is to ensure that the generated continuation is not only syntactically consistent with the input phrase, but also “sounds right”, according to global features of the phrases.

The scheme is outlined in Fig. 3. When an input phrase is detected a series of feature analysis algorithms are applied, mainly: Tempo extraction, phase induction and dynamics estimation. The tempo extraction uses an autocorrelation technique adapted to our Midi representation for efficiency. The phase induction attempts to find the most prominent beats using the tempo extracted. Dynamics is simply modeled as a mean of the velocity in the input phrase. These features are then remapped in the output phrase: time stretching for tempo, translation for phase, and multiplication of dynamics.

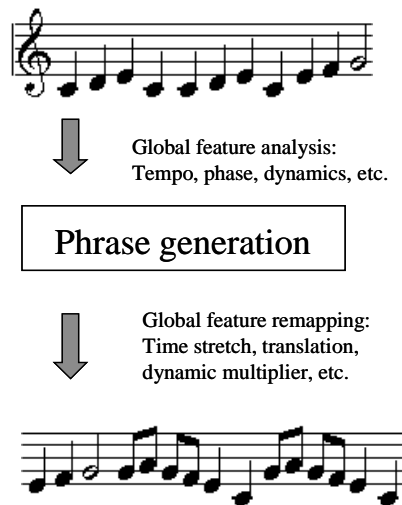


Fig. 3. Feature analysis and remapping

Current experiments showed that beat and dynamics are the most important in real life situations, but other features can be added to the list, such as dynamic evolution, to further ensure continuity between the input and output phrase.

3.2 Real Time Generation

The real time generation is a strictly technical issue but is an important aspect of the system since it is precisely what allows the system to take into account external information quickly, and ensure that the music generated follows accurately the input, and remains controllable by the user.

3.2.1 How Fast Is Fast?

To give an estimation of our real time constraints, we have to know how fast a musician can play. We have considered an example by John McLaughlin, considered as one of the fastest guitarist in the world, in an example performed for a demo of a pitch to Midi converter (<http://www.musicindustries.com/axon/archives/john.htm>). An analysis of the fastest parts of the sample yields a mean duration of 66 milliseconds per note. Of course, this figure is not definitive, but can be taken as an estimate for a reasonable maximum speed. Our system should have a response time short enough so that it is impossible to perceive a break in the note streams, from the end of the player's phrase, to the beginning of the system's continuation: a good estimation of the maximum delay between two fast notes is about 50 milliseconds.

3.2.2 Thread Architecture

The real time aspect of the system is handled as follows. Incoming notes are detected by the system using the interruption polling process of MidiShare [12]: each time a note event is detected, it is added to a list of current note events. Of course, it is impossible to trigger the continuation process only when a note event is received. To detect phrase endings, we introduce a phrase detection thread which periodically wakes up and computes the time elapsed between the current time and the time of the last note played. This time delta is then compared with a *phraseThreshold*, which represents the maximum time delay within notes of a given phrase. If the time delta is less than the *phraseThreshold*, the process sleeps for *SleepTime* milliseconds. If not, a new phrase is detected and the continuation system is triggered, which will compute and schedule a continuation. The phrase detection process is represented in Figure 3.

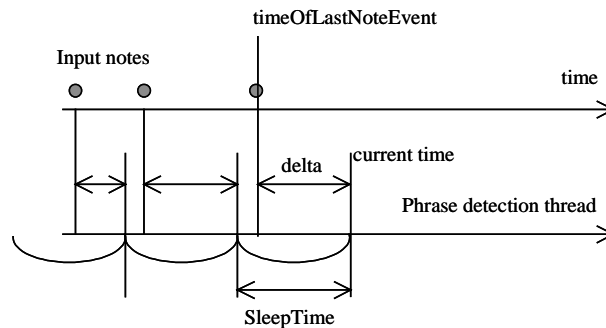


Fig. 4. The input phrase detection process.

In other words, each time the phrase detection thread wakes up at time t , it computes the current time delay δ :

```
delta := currentTime - timeOfLastNoteEvent
```

It then compares this delay with the phrase threshold, decides or not to detect a phrase ending, and schedules itself to wake up at $t + \text{SleepTime}$:

```
If (delta >= phraseThreshold) then detectPhrase();
Sleep (SleepTime)
```

The real time constraint we have to implement is therefore that the continuation sequence produced and played by the system should be played within a maximum of 50 milliseconds after the last note event. The delay between the occurrence of the last note of a phrase and the detection of the end of the phrase is bounded by *SleepTime*.

In practice, we use a value of 20 milliseconds for *SleepTime*, and a *phraseThreshold* of 20 milliseconds. The amount of time spent to compute a continuation and schedule it is on average 20 milliseconds, so the total amount of time spent to play a continuation is in the worse case of 40 milliseconds, with an average value of 30 milliseconds, which fits in the scope of our real time constraint.

3.2.3 Step-by-Step Generation Process

The second important aspect of the real time architecture is that the generation of musical sequences is performed step-by step, in such a way that any external information can be used to influence the generation (see next section). The generation is performed by a specific thread (generation thread), which generates the sequence by chunks. The size of the chunks is parameterized, but can be as small as 1 note event. Once the chunk is generated, the thread sleeps and wakes up for handling the next chunk in time. When phase information is available (see feature remapping section above) the generated phrase is translated to be played on the beat.

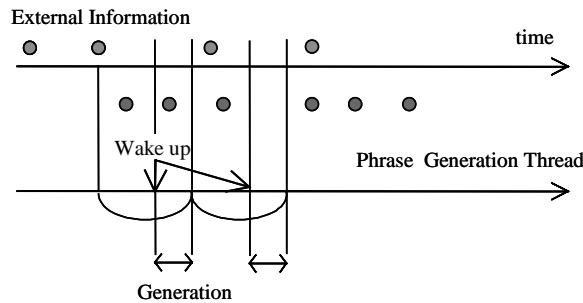


Fig. 5. The step-by-step Generation Process allows external information to be taken into account continuously.

3.3 Biasing Markov Generation

The main idea to turn our automaton into an interactive system is to influence the Markovian generation by characteristics of the input. As we saw above, the very idea of Markov-based generation is to produce sequences in such a way that the probabilities of each item of the sequence are the probabilities of occurrences of the items in the learnt corpus.

In the context of musical interaction, this property is not always the right one, because many things can happen during the generation process. For instance, in tonal

music, the harmony can change: in a Jazz trio for instance, the pianist will play chords which are not always the same, throughout the generation process. Because we target a real world performance context, these chords are not predictable, and cannot be learnt by the system prior to the performance. The system should be able somehow to take this external information into account during the generation, and twist its generated sequence in the corresponding direction.

The idea is to introduce a constraint facility in the generation phase. External information may be sent as additional input to the system. This information can be typically the last 8 notes (pitches) played by the pianist for instance, if we want the system to follow harmony. It can also be the velocity information of the whole band, if we want the system to follow the amplitude, or any information that can be used to influence the generation process. This external input is used to influence the generation process as follows: when a set of possible continuation nodes is computed (see section on generation), instead of choosing a node according to its Markovian probability, we weight the nodes according to how they match the external input. For instance, we can decide to prefer nodes whose pitch is in the set of external pitches, to favor branches of the tree having common notes with the piano accompaniment.

In this case, the harmonic information is provided implicitly, in real time, by one of the musicians (possibly the user himself), without having to explicitly enter the harmonic grid or any symbolic information in the system.

More precisely, we consider a function $Fitness(x, Context)$ with values in $[0, 1]$ which represents how well item x fits with the current context. For instance, a Fitness function can represent how harmonically close is the continuation with respect to external information. If we suppose that *piano* contains the last 8 notes played by the pianist (and input to the system), $Fitness$ can be defined as:

$$Fitness(p, piano) = \frac{\text{nb notes common to } p \text{ and piano}}{\text{nb notes in piano}}$$

This fitness scheme is of course independent of the Markovian probability defined above. We therefore introduce a new weighting scheme which allows to parameterize the importance of the external input, via a parameter S (between 0 and 1):

$$Prob(x) = S * Markov_Prob(x) + (1 - S) * Fitness(x, Context)$$

By setting S to extreme values we can get two extreme behaviors:

- $S = 1$, we get a musical automaton insensitive to the musical context,
- $S = 0$, we get a reactive system which generates the closest musical elements to the external input it finds in the database.

Of course, interesting values are intermediary: when the system generates musical material which is both stylistically consistent, and sensitive to the input. Experiments in these various modes are described below in the Experiment Section.

3.4 Control and High-Level Structure

Playing “interesting” phrases is an important ingredient of musical improvisation, but it is not the only one. High-level structure is as important to produce a full-fledged piece: it is not always desirable to have the system continue systematically all phrases played. Typically, the musician can start a piece by playing, e.g. a musical theme, and

then let the system play progressively longer and longer continuations until the end, when the musician plays back the theme, without the system continuation.

To allow the user to switch between these different modes in an intimate and non-intrusive way, we have identified a set of parameters that are easy to trigger in real time, without the help of a graphical interface. The most important parameter is the S parameter defined above, which controls the “attachment” of the system to the external input. The other parameters allow the musician to switch on or off basic functions such as the learning process or the continuation process.

By default, the system stops playing when the user starts a phrase, to avoid superposition of improvisations. With minimum training, this mode can be used to produce a unified stream of notes, thereby producing an impression of seamlessness between the sequence actually played by the musician and the one generated by the system.

Additionally, a set of parameters can be adjusted from the screen, such as the number of notes to be generated by the system (as a multiplicative factor of the number of notes in the input sequence), and the tempo of the generated sequence (as a multiplicative factor of the tempo of the incoming sequence). One important control parameter allows the musician to force the system to remain in some regions, deemed interesting or particularly well suited to the moment of the piece. This parameter is usually associated with a Midi control change such as the breath control. By pushing the control up, the system will remain in the region, and instantaneously become a sort of arpeggiator, creating a periodic rhythm from the local Markovian region selected. Another trigger of the same control restores the system back to the usual Markovian generation process, which results in forcing the system to explore another musical region.

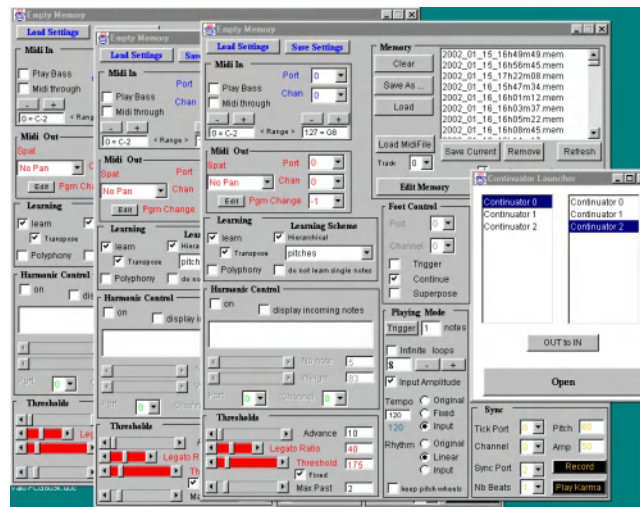


Fig. 6. Multiple copies of the Continuator in action

The system described above contains many parameters, but is in some sense autonomous. There are musical situations in which it is interesting to use several,

independent versions of the system, each with its own inputs and outputs. We have designed a scheme which is able to launch different continuators at the same time, possibly synchronizing them (see Fig. 6).

4 Experiments

We have conducted many experiments with the system, in various modes and configurations to validate our claims. We report results and lessons learned in the following sections.

4.1 Indistinguishability

It is difficult to describe music by words, and rate its quality, especially jazz improvisation. However, we can easily rate how the system differs from the human input. We have conducted tests to check whether listeners could tell when the system is playing or not. In most of the cases, if not all, the music produced is indistinguishable from the user's input. This is typically true for quick and fast Jazz solos. An audio example available at our web site gives an example of a Jazz tune ("On Green Dolphin Street", by Kaper & Washington), where the musician (here, the author) plays a guitar solo which is continued by the system, interrupts several times the system to launch another phrase, and finally concludes the improvisation. The reader/listener can assess the difficulty in distinguishing these different phases as the whole improvisation is seamless. Other examples can be found at our web site, in which the system generates long and often impressive jazzy phrases in the styles of guitarists such as Pat Martino, John McLaughlin, or Alan Holdsworth.

4.2 Attachment

The attachment mechanism we have introduced is particularly spectacular when used in conjunction with a fixed metrical structure. In this mode, the system can play an accompaniment in a given tempo which tries to satisfy two conflicting constraints: 1) stylistic consistency and 2) consistency with the external input. Audio Examples will be presented in which the system plays a chord sequence (from previously learnt material), and tries in real time to "follow" harmonically the input by a real musician (the author again). The chords generated by the system fit naturally and quickly to harmonic changes. Occasional unexpected harmonic progressions are also generated, but which all fit the two constraints of stylistic consistency and fitness with external input.

Many experiments in the various styles of the Karma music workstation were also recorded and will be made available at our web site. In these experiments, we have connected the Continuator to the Korg Karma workstation, both in input and output. The Continuator is used as an additional layer to the Karma effect engine. The Continuator is able to generate infinite variations from simple recordings of music, in virtually all the styles proposed by the Karma (over 700).

4.3 Subjective Impressions: The Aha Effect

Besides the evaluation of the musical quality of the music produced by the system, we have noticed a strong subjective impression on the musician playing. We have conducted a series of experiments and concerts with famous Jazz musicians, and the reactions of musicians playing with the system were always extremely positive. The most striking effect, noticed systematically on all musicians experimenting with the system, can be best described as a *Aha* effect, triggered by the sudden realization that the system is starting to play exactly in the same style as oneself, or suddenly pops up patterns played a few minutes earlier.

The accompanying Video shows a series of such effects on different musicians, styles and instruments (Bernard Lubat, Alan Silva, Claude Barthélémy), with sudden and characteristic bursts of laughter or astonishment. Some of them are illustrated in Figures 5 and 6.

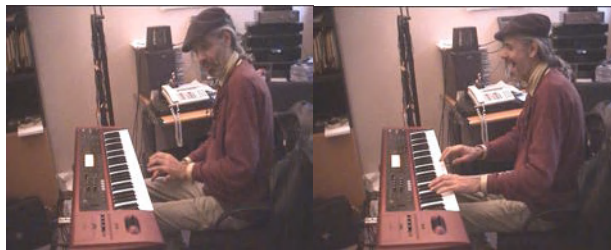


Fig. 5. Jazz musician Alan Silva playing with Continuator.



Fig. 6. Bernard Lubat playing with the Continuator.

4.4 New Musical Collaborative Modes

An interesting consequence of the design of the system is that it leads to several new playing modes with other musicians. Traditionally, improvised music has consisted in quite limited types of interaction, mostly based around question/answer systems [2] [3]. With the Continuator, new musical modes can be envisaged:

- *Single autarchy*. One musician plays with the system after having fed the system with a database of improvisations by a famous musician, as Midi files. We have experimented in particular with a database of midi choruses from Pat Martino, provided by [7], and a database of Bernard Lubat's piano style. An extension of

this mode consists in using several versions of the same system, with the same inputs, but generating simultaneously different outputs. Used in the linear rhythmic mode, this configuration results in a multiple voice arpeggiator which that produces variations continuously.

- *Multiple autarchy*: each musician has her own version of the system, with its own database. This provides a traditional setting in which each musician plays with his/her own style. Additionally, we experimented concerts in which one musician (György Kurtág) had several copies of the system linked to different midi keyboards. The result for the listener is a dramatic increase in musical density.
- *Master/Slave*: one musician uses the system in its basic form, another (e.g. pianist) provides the external data to influence the generation. This is typically useful for extending a player's solo ability while following the harmonic context provided by another musician.
- *Cumulative*: all musicians share the same pattern database. This setting was experimented with during a Jazz festival (Uzeste, France), where two musicians played with the same (Bernard Lubat) database,
- *Sharing*: each musician plays with the pattern database of the other (e.g. piano with guitar, etc.). This creates exciting new possibilities as a musician can experience playing with unusual patterns.

5 Conclusion

We have described a music generation system which is able to produce music learnt in an agnostic manner, while remaining intimately controllable. This is made possible by introducing several improvements to the basic Markovian generation, and by implementing the generation as a real time, step-by-step process. The resulting system is able to produce musical continuations of any user – including beginners - according to previously learnt, arbitrary styles.

The experiments and concerts performed with professional artists show not only that the music generated is of very high quality (as good as the music learnt by the system), but, more importantly, that such a learning facility can be turned into an actual music instrument, easily and seamlessly integrated in the playing mode of the musician. Current work focuses on the design of an audio version to expand the possibility of musical input (voice in particular). This version will use the same kernel described here, augmented with audio descriptors extracted in real time. These descriptors are made possible by ongoing work on musical metadata (in particular in the Cuidado project, see[14]). The resulting system, besides extending the possibility to audio, will also provide a link between the domain of musical performance and musical listening.

References

1. Assayag, G. Dubnov, S. Delerue, O. Guessing the Composer's Mind: Applying Universal Prediction to Musical Style, Proc. ICMC 99, Beijing, China, I.C.M.A., San-Francisco, 1999.
2. Baggi, D. L. NeurSwing: An Intelligent Workbench for the Investigation of Swing in Jazz, in Readings in Computer Generated Music, IEEE Computer Society Press, 1992.

3. Biles, John A. Interactive GenJam: Integrating Real-Time Performance with a Genetic Algorithm, Proc. ICMC 98, Ann Arbor, Michigan, 1998.
4. Jan Borchers, Designing Interactive Musical Systems: a Pattern Approach, HCI International '99. 8th International Conference on Human-Computer Interaction, Munich, Germany, from 22-27 August, 1999.
5. Conklin, D. and Witten, Ian H. Multiple Viewpoint Systems for Music Prediction, JNMR, 24:1, 51-73, 1995.
6. Cope, David. Experiments in Musical Intelligence. Madison, WI: A-R Editions, 1996.
7. Heuser, Jorg, Pat Martino – His contributions and influence to the history of modern Jazz guitar. Ph.D thesis, University of Mainz (Ge), 1994.
8. Hiller, L. and Isaacson, A., *Experimental Music*, New York: McGraw-Hill, 1959.
9. Karma music workstation, Basic guide. Korg Inc. Available at: http://www.korg.com/downloads/pdf/KARMA_BG.pdf, 2001.
10. Lartillot O., Dubnov S., Assayag G., Bejerano G., Automatic modeling of musical style Proc. ICMC 2001, La Habana, Cuba.
11. New Interfaces for Musical Expression (NIME'01), <http://www.csl.sony.co.jp/person/poup/research/chi2000wshp/>, 2000.
12. Orlarey, Y. Lequay, H. MidiShare: a Real Time multi-tasks software module for Midi applications Proceedings of the International Computer Music Conference, Computer Music Association, San Francisco, pp. 234-237, 1989.
13. Pachet, F. Roy, P. "Automatic Harmonization: a Survey", Constraints Journal, Kluwer, 6:1, 2001.
14. Pachet, F. "Content-Based Management for Electronic Music Distribution", Communications of the ACM, to appear, 2002.
15. Ramalho G., Ganascia J.-G. Simulating Creativity in Jazz Performance. Proceedings of the National Conference in Artificial Intelligence, pp. 108-113, AAAI-94, Seattle, AAAI Press, 1994.
16. Robert Rowe, Machine Musicianship, MIT Press, 2001.
17. Ron, D. Singer, Y and Tishby, N., "The power of amnesia: learning probabilistic automata with variable memory length", Machine Learning 25(2-3):117-149, 1996.
18. J. L. Triviño-Rodríguez; R. Morales-Bueno, Using Multiattribute Prediction Suffix Graphs to Predict and Generate Music, Computer Music Journal 25 (3) pp. 62-79, 2001.
19. William F. Walker A Computer Participant in Musical Improvisation, Proc. of CHI 1997. Atlanta, ACM Press, 1997.

Recognition of Isolated Musical Patterns Using Hidden Markov Models

Aggelos Pikrakis¹, Sergios Theodoridis¹, and Dimitris Kamarotos²

¹ Division of Communications and Signal Processing
Dept. of Informatics and Telecommunications
University of Athens, Greece
{pikrakis, stheodor}@di.uoa.gr
<http://www.di.uoa.gr/dsp>

² IPSA Institute of the Aristotle University of Thessaloniki
dimik@otenet.gr

Abstract. This paper presents an efficient method for recognizing isolated musical patterns in a monophonic environment, using Discrete Observation Hidden Markov Models. Each musical pattern is converted into a sequence of music intervals by means of a fundamental frequency tracking algorithm followed by a quantizer. The resulting sequence of music intervals is presented to the input of a set of Discrete Observation Hidden Markov models, each of which has been trained to recognize a specific type of musical patterns. Our methodology has been tested in the context of Greek Traditional Music, which exhibits certain characteristics that make the classification task harder, when compared with Western musical tradition. A recognition rate higher than 95% was achieved. To our knowledge, it is the first time that the problem of isolated musical pattern recognition has been treated using Hidden Markov Models.

1 Introduction

In many musical traditions, certain musical patterns have been shaped and categorised through practice and experience over the years. An important task of interest to the musicologist and ethnomusicologist is the automated search of such pre-defined sound patterns within a large number of sound files stored *in raw audio format*. This paper offers a first step towards the development of digital signal processing tools to achieve this task and proposes a scheme for the recognition of isolated musical patterns in a monophonic environment. Previous work based on raw data has employed Dynamic Time Warping techniques [8]. The term *isolated* refers to the fact the patterns have been isolated from their context by means of a *manual segmentation process*. Automated segmentation of the audio file to musical patterns is outside the scope of this paper.

As a case study, the proposed recognition scheme was applied on musical patterns originating from *Greek traditional clarinet*, a single non-polyphonic instrument recorded under laboratory conditions with an ambient noise of less than 5dB. From a large number of types of musical patterns, encountered in

practice in Greek traditional music, we have selected twelve typical cases because of their common use in practice and their respective time elasticity. Time elasticity along with certain other characteristics exhibited by Greek Traditional music makes the classification task harder when compared with Western music (see Section 4).

Our approach consists of two stages. At a first stage, a feature generation algorithm converts the unknown musical pattern into a sequence of music intervals (multiples of one quarter-tone) by means of a fundamental frequency tracking algorithm followed by a quantizer. The second stage employs a set of Discrete Observation Hidden Markov Models (DOHMM's) [5], [10], [13], in order to determine the unknown musical pattern. Each DOHMM has been trained to match a specific type of the musical patterns. The DOHMM that generates the highest a posteriori probability is considered to be the one that matches best the unknown pattern.

Section 2 presents the aforementioned feature generation procedure and Section 3 deals with the architecture of the DOHMM recognizer. Section 4 presents details of the application of the method to patterns from the Greek Traditional music. Conclusions and future work are presented in Section 5.

2 Feature Generation

2.1 Fundamental Frequency Tracking

The goal of the feature generation stage is to convert the unknown musical pattern into a sequence of music intervals (frequency jumps) that are multiples of one quarter-tone. For the signals encountered in Greek traditional music, frequency jumps equal to one quarter-tone are quite a common phenomenon.

At a first step, our method extracts a sequence of fundamental frequencies from the unknown musical pattern. This can be achieved by any robust fundamental frequency tracking algorithm. Having experimented with several algorithms, we adopted Brown's narrowed autocorrelation method [2] because it gave the best results for the signals that we studied, with respect to accuracy and frequency doubling. Certain implementation details for Brown's method are presented in Section 4. As an alternative we propose Tolonen's multipitch analysis model [4]. The table below lists the algorithms that were tested. Figure 2

Fundamental frequency tracking methods	
Frequency domain	Time domain
Schroeder's histogram [11]	Brown's narrowed autocorrelation method [2]
Schroeder's harmonic product spectrum [11]	Cooper and Kia's method [3]
Piszscalski's method [9]	Tolonen's multi-pitch analysis model [4]
Brown's pattern recognition method based on a constant-Q transform [1]	

demonstrates the fundamental frequency tracking results for one of the patterns whose frequency contour plot is depicted in Figure 1.

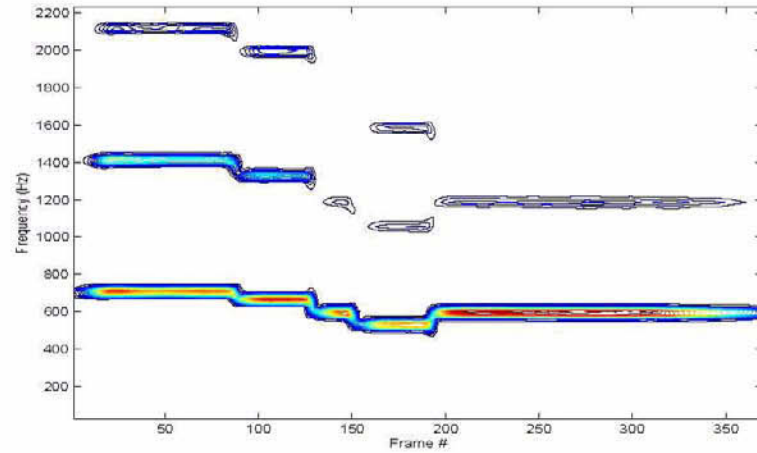


Fig. 1. Contour plot of the spectrogram of a musical pattern of Type II

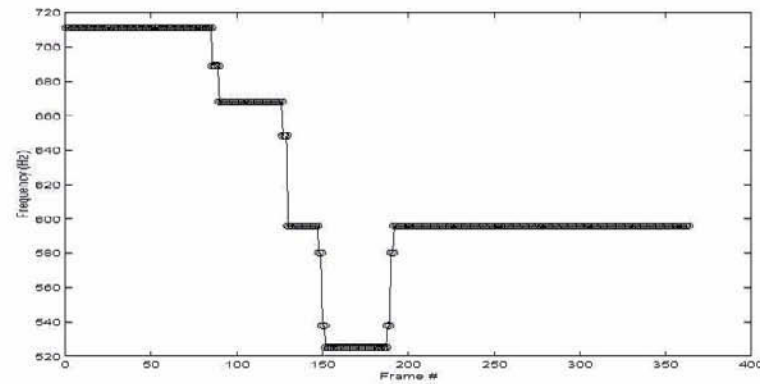


Fig. 2. Fundamental frequency tracking results for the pattern of Figure 1 using the narrowed autocorrelation method

During this processing, errors are likely to occur in certain frames. Some of the errors can be eliminated at a post-processing step, by setting the pitch period of a frame to be equal to the average of the pitch periods of its two neighboring frames. However, most of these errors appear in more than one consecutive frames and cannot be removed using just simple heuristic post-processing rules. Such is the case when a transition between notes takes place, as in Figure 2. We deal with this situation in Section 3.1 where the design methodology for the DOHHM's is explained.

2.2 Quantization of the Extracted Frequencies

This step aims at converting the generated sequence of fundamental frequencies $\{\mathbf{f} : f_i, i = 1 \dots M\}$, to a sequence of frequency jumps (music intervals). Each f_i is first mapped to a positive number, say k , equal to the distance (measured in quarter-tone units) of f_i from f_s , i.e., $k = \text{round}(24 \log_2 \frac{f_i}{f_s})$, where f_s is the lowest fundamental frequency that the clarinet can produce (for the signals that we studied $f_s = 146.8 \text{ Hz}$) and $\text{round}(\cdot)$ denotes the roundoff operation. As a result, sequence \mathbf{f} is mapped to sequence $\mathbf{L} = \{l_i, i = 1 \dots M\}$, where l_i lies in the range 0 to 80. This mapping process imitates some aspects of the human auditory system, which is known to analyse an input pattern using a logarithmic frequency axis.

It is now straightforward to compute D , the sequence of music intervals (frequency jumps), from the symbol sequence \mathbf{L} . This is achieved by calculating the difference of symbol sequence \mathbf{L} , i.e., $\mathbf{D} = \{d_{i-1} = l_i - l_{i-1}, i = 2 \dots M\}$.

Representing the unknown musical pattern as a sequence of music intervals deals with the fact that instances of the same musical type may have different starting frequencies, i.e., may appear at different frequency bands. It is also important to notice that each note in a musical pattern is likely to span more than one consecutive frames, therefore, most of the time l_i is equal to l_{i-1} and as a result $d_i = 0$ for most of the frames (i's). The d_i 's fall in the range of $-G$ to G , where G corresponds to a maximum allowed frequency jump ($G = 60$ quarter-tones, i.e., 15 tones for the signals that we studied).

3 Discrete Observation HMM's

In the sequel, sequence $\mathbf{D} = \{d_i, i = 1 \dots M - 1\}$ is provided as input to a maximum likelihood classifier based on a set of 12 DOHMM's. Each DOHMM has been trained to model a specific musical pattern and is represented by means of a set of variables $\lambda = (S, V, A, B, \pi)$, where S is the set of states of the model, $V = \{-G, \dots, G\}$ is the finite alphabet for the features d_i , A is the set of the state transition probabilities, B is the set of the observation symbol probabilities for the set S , and π is the set of the initial state probabilities. For each DOHMM, the probability of occurrence of the sequence of feature observations \mathbf{D} , resulting from the unknown pattern, is computed. The DOHMM corresponding to the largest probability is considered to be the model which matches best the specific observation sequence \mathbf{D} . For the calculation of the above probabilities, the scaled version of the forward-backward (Baum-Welch) approach was adopted ([10]).

The rest of this section describes the topology of the DOHMM's and the training method that was employed in order to determine the parameters A , B and π .

3.1 Topology of the DOHMM's

In order to explain the design methodology adopted for developing the DOHMM's that model the patterns under study, certain observations regard-

ing the structure of the feature sequence \mathbf{D} have to be taken into account. As it is explained, the modeling philosophy accounts for all possible deviations a musical instance may exhibit with respect to a typical reference pattern.

It has already been pointed out in Section 2.2 that a number of consecutive values of d_i is usually zero. This number, of course, exhibits a random variation from case to case. In the ideal case, different instances of the *same* musical type should correspond to symbol sequences that differ *only* in the number of these zeros, due to the phenomenon of time elasticity. However in practice, the following deviations from this ideal situation are often encountered:

- a) Certain intervals are one quarter-tone higher or lower with respect to the reference pattern. This is most usually due to errors during the feature generation stage, but can also be due to variations among instrument players.
- b) The sequence of zero-valued d_i 's is often intervened by negative or positive jumps, equal to one quarter-tone, usually appearing in pairs. Such phenomena are due to errors in the feature extraction stage.
- c) In some cases, certain jumps are missing or are “broken” into two successive jumps whose sum is equal to the original. Such is the case, for example, in Figure 2. The frequency jump just before frame #100 is split into two successive jumps, i.e., $710 \rightarrow 690$ and $690 \rightarrow 670$. This is due to errors in the feature generation stage whenever a transition between notes takes place. This phenomenon can also be due variations among instrument players.

It must be pointed out that the human ear is immune to such deviations and is likely to achieve a high recognition rate when classifying such musical patterns.

Our design methodology will be presented via an example, using musical type I (corresponding to the contour plot of Figure 3,) which will demonstrate our philosophy in developing the architecture of the DOHMM's.

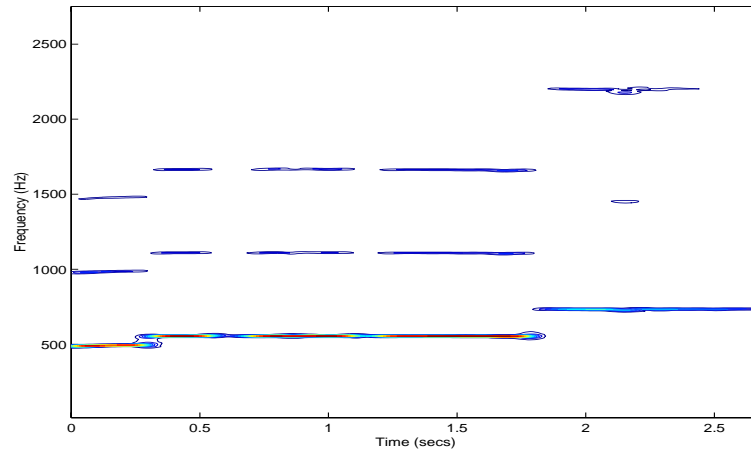


Fig. 3. Contour plot of the spectrogram of a musical pattern of Type I

1. In the first step, one has to determine, ignoring deviations, the number of music intervals that *typically* appear in the resulting symbol sequences of each specific musical type. Patterns of type I, typically consist of three notes, i.e., two frequency jumps, the first being equal to $10QT = 2\frac{1}{2}T$ and the second equal to $8QT = 2T$ (QT=quarter tone, T=tone). In the sequel, one state is assigned per each non-zero jump. For notational purposes, these states are referred to as S -states and are indexed as $S_1 \dots S_k$. For example, for type I patterns, $k = 2$, yielding two states, S_1 and S_2 . For the state transitions, the following assumptions have been adopted, based on extensive experimentation. If the model is at a state S_i , it may either remain in this state or proceed to the next state S_{i+1} . For some musical types, in order to deal with the fact that certain music intervals may be absent from the pattern to be recognised, we may allow, for some states, the transition from S_i to S_{i+2} to take place.
2. In the next step, we consider the implications imposed by the deviations from the ideal case, as explained in b) above. Thus, a new set of states, P , is introduced, which are inserted in the beginning and in the end of the DOHMM, as well as between each pair of successive S -states. Following the notation introduced for the S -states, these are the P -states and are indexed similarly. Although these are also states, they are treated via a different notation, in order to point out that they model deviations from the ideal case. For the DOHMM of type I, three such states are inserted, P_1 in the beginning, P_3 in the end and P_2 between S_1 and S_2 . Thus, this DOHMM consists, so far, of 5 states, indexed as $\{P_1, S_1, P_2, S_2, P_3\}$ (Figure 4). If the DOHMM is at a P state, it is allowed to either remain in this state or jump to the closest S -state on its right. In addition, if the model is at a state S_i , it may jump to the P -state separating S_i from S_{i+1} . In our example, S_1 may lead to S_1 , S_2 or P_2 .

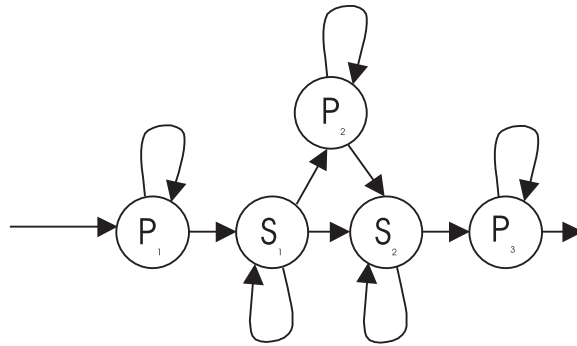


Fig. 4. Intermediate stage in the design of the DOHMM of type I

3. A case that has to be treated separately is when a frequency jump, normally assigned to some state S_k is “broken” into a sequence of two successive jumps. For example, for type I patterns, it has been observed that the first frequency jump (equal to 10QTs) is often “broken” to a jump of 2QTs and a jump of 8QTs. This demands to include in the DOHMM a completely new branch that, in a way, serves as a substitute to state S_k . This branch consists of two states S_{k1} and S_{k2} , plus two P states, P_{k1} between S_{k1} and S_{k2} and P_{k2} following S_{k2} (Figure 5.) In order to bind this

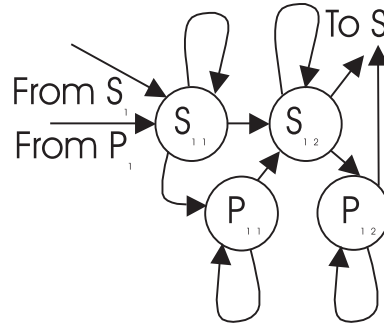


Fig. 5. Branch to add for the first S -state of the DOHMM of musical Type I

new branch with the rest of the DOHMM, let us denote with P_{k-1} the P state preceding state S_k . We demand that the state transition probability $A(P_{k-1}, S_{k1}) > 0$, $A(P_{k2}, S_{k+1}) > 0$ and $A(S_{k2}, S_{k+1}) > 0$. In our example, states $\{S_{11}, S_{12}, P_{11}, P_{12}\}$ form the new branch. The final version of the DOHMM of type I can be seen in Figure 6 and is the result of the superposition of the models in figures 4 and 5. It can be observed that our methodology does not permit transitions from one state to its predecessors, thus leading to the so called left-to-right or Bakis models.

4. The next step is to determine which symbols are emitted from each state. From the respective definition, the P states are allowed to emit the symbols 0, -1 and 1. Each S -state can emit a) the symbol 0, or b) the frequency jump d_k that has been assigned to the state, or c) frequency jumps that differ one quarter-tone from d_k . This can accommodate the phenomena grouped under case a) in the beginning of Section 3.1. In addition, if a state S_i can be reached from state S_{i-2} , then S_i has to emit the sum of frequency jumps assigned to S_i and S_{i-1} .
5. Finally, the initial states of the model have to be determined, i.e., the probability of accessing the various nodes at the first transition. The set of initial states must include P_1 and S_1 . If a parallel branch exists for S_1 , then S_{11} is also inserted in this set.

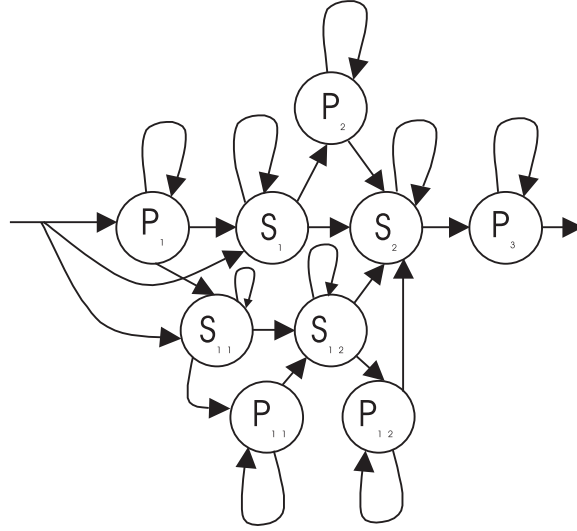


Fig. 6. Final version of the DOHMM for musical type I

The architecture of each DOHMM is the result of a careful study of the structure of the respective musical patterns (as revealed by the contour plot of the spectrograms), and extensive experimentation, guided by the methodology presented above, regarding the parameters involved. Due to paper length restrictions, the spectrograms, alongside with the respective DOHMM's for the twelve types of musical patterns under study, can be accessed at <http://www.di.uoa.gr/pikrakis/patterns>.

3.2 Training of the DOHMM's

Each DOHMM was individually trained to recognize a specific type of musical pattern. For each type of musical pattern, a set of N symbol sequences, $\mathbf{O} = [\mathbf{D}^{(1)}, \mathbf{D}^{(2)}, \dots, \mathbf{D}^{(N)}]$, was chosen for the training phase ($N = 40$). The goal of the training method was to adjust the model parameters to maximize the probability $P(\mathbf{O} | \lambda) = \prod_{k=1}^N P(\mathbf{D}^{(k)} | \lambda)$. In other words, a multiple observation training procedure was adopted [10]. This is achieved by means of an iterative procedure. During each iteration, each observation sequence $\mathbf{D}^{(k)}$ is presented to the input of the DOHMM and for each $\mathbf{D}^{(k)}$, frequencies of occurrence of various events are calculated. At the end of every iteration, the individual frequencies of occurrence are added together and the cumulative results are used to adjust the model parameters (A , B , π), using the scaled version of the Baum-Welch reestimation formulae for multiple training sequences [10].

The choice of initial estimates for the DOHMM parameters is crucial for the convergence in the training phase. Although random values may be assigned to the model parameters following a uniform distribution, it turns out that a higher

maximum of the likelihood function can be reached if the topology rules defined in section 3.1 are exploited. Specifically:

1. Initialization of A : let F be the set of allowable transitions from a state k . Then, $A(k, i) = 1/|F|$, $\forall i \in F$, where $|F|$ is the size of the set F . All other transitions are assigned a small constant of the order of 10^{-5} . This is repeated for all states.
2. Initialization of B : let R be the set of symbols that can be emitted by state k . Then $B(i, k) = \frac{1}{|R|}$, $\forall i \in R$, where $|R|$ is the size of set R . All other symbol probabilities are assigned a small constant in the order of 10^{-5} . This is again repeated for all states.
3. Initialization of π : If S_i is an initial state, then $\pi(S_i) = 1/|Z|$ where Z is the size of the set of allowable initial states of the DOHMM.

Another issue that has to be carefully treated is the fact that, depending on the training set, parameters that have been assigned non-zero values upon initializing the DOHMM, may turn out to have a zero value at the end of the training phase. This results in a zero probability for certain musical patterns during the recognition phase. To circumvent this situation, all zero-valued parameters, at the end of the training phase, are assigned a very small constant of the order of 10^{-5} .

Convergence of the parameters to their final values was achieved after 15 – 20 iteration steps at the most.

4 Application of the Method in the Context of Greek Traditional Music

Our effort was focused on the monophonic melodic lines of one Greek popular instrument, the Greek traditional clarinet, which is an instrument of the clarinet family that closely resembles the western-type clarinet.

A set of 1200 musical patterns, equally spread over many different initial tones, were generated by four professional Greek Traditional Clarinet players in a monophonic environment, under laboratory conditions, involving all the aforementioned twelve types of musical patterns. A sampling rate of 22050Hz was used. The criteria for the choice of musical types were their common use in practice and the differences on time length that occur in their use. This last musical characteristic –the elasticity of the musical pattern, retaining its musical function, while stretching its length (both locally and as a whole) up to five times in some cases– is a major problem when studying such musical patterns.

The musical system of Greek traditional music and the techniques used by the instrument players give to the sound material that we studied a very different structure, when compared with this of a typical western equal-tempered tradition [6]. Major differences are:

- the intervalic system (system of musical scales) that contains many musical intervals that are smaller than the well-tempered, and
- the use of larger, formalised transitory patterns as a main element of the

musical structure and not as an ornamental one, as it is in the case of western musical tradition.

For the feature generation stage, fundamental frequency tracking was achieved by means of Brown's narrowed autocorrelation method. A window length equal to 1024 samples was used (shifted $5ms$ each time) and narrowing was achieved by means of four shifted versions of the autocorrelation function. The use of more than four shifted versions did not affect the performance of the frequency tracker.

For the quantization step we used an alphabet of 121 discrete symbols, with each symbol being equal to a frequency jump in the range of $-60 \dots +60$ quarter-tones, i.e. $G = 60$ (Section 2.2).

All twelve DOHMM's were designed following the design methodology introduced in Section 3.1. Approximately 40 patterns per musical type were used as the training set for the respective DOHMM. The rest of the patterns served as the data set. Our tests were carried out using Matlab. The total recognition rate was above 95% irrespective of the fundamental frequency tracking method employed. This is because the architecture of the DOHMM's deals successfully with the deviations described in Section 3.1. The confusion matrix of Table 1 demonstrates the distribution of recognition failures among the musical types considered.

Table 1. Confusion matrix of recognition failures. The number at cell (i, j) is equal to the percentage of test patterns that have been wrongly classified as type j patterns instead of type i patterns. empty cells correspond to a zero error rate.

Type	1	2	3	4	5	6	7	8	9	10	11	12
1			0.28%	0.42%								
2						0.14%						
3	0.28%			0.42%								
4	0.42%		0.42%									
5								0.28%				
6		0.28%										
7												0.28%
8					0.28%							
9												0.28%
10	0.14%		0.14%									
11					0.28%							
12							0.28%					

5 Conclusions and Future Research

In this paper an efficient scheme for the recognition of isolated musical patterns was presented. The methodology was applied with success in the context of Greek Traditional Music. Future research will focus on applying the new recognition

scheme in the context of Classic Western Music and with other instruments besides clarinet. Also a focus of our research activity is to employ continuous observation Hidden Markov Models, used in continuous speech recognition systems. This will also permit to experiment with the extraction of multi-dimensional features from the musical patterns to be recognised. Currently, the DOHMM methodology is also applied for multimedia retrieval based on MIDI transcription.

References

1. J.C. Brown, "Musical fundamental frequency tracking using a pattern recognition method", *Journal of the Acoustical Society of America*, Vol. 92, No 3, 1992
2. J.C. Brown and B. Zhang, "Musical frequency tracking using the methods of conventional and narrowed autocorrelation", *Journal of the Acoustical Society of America*, Vol. 89, No 5, 1991
3. D. Cooper, K.C. Ng, "A Monophonic Pitch-Tracking Algorithm based on Waveform Periodicity Determinations using Landmark Points", *Computer Music Journal*, Vol. 20, No 3, Fall 1996
4. T. Tolonen and M. Karjalainen, "A Computationally Efficient Multipitch Analysis Model", *IEEE Transactions on Speech and Audio Processing*, Vol. 8, No 6, November 2000
5. J. Deller, J. Proakis, J. Hansen, *Discrete-Time Processing of Speech Signals*, McMillan, 1993
6. S. Karas, *Theoritikon - Methodos*, on Greek Traditional Music, ed. Athens, 1982 (in Greek) A Survey", *Proceedings of the IEEE*, Vol. 65, No 8.
7. P.E. Papamichalis, *Practical Approaches to Speech Coding*, Prentice-Hall, 1987
8. A. Pikrakis, S. Theodoridis, D. Kamarotos, "Recognition of Isolated Musical Patterns using Context Dependent Dynamic Time Warping", to appear in *IEEE Transactions on Speech and Audio Processing*.
9. M. Piszczalski and B. Galler, "Predicting musical pitch from component frequency ratios", *Journal of the Acoustical Society of America*, Vol. 66, No 3, 1979
10. L.R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", *Proceedings of the IEEE*, Vol. 77, No. 2, 1989
11. M.R. Schroeder "Period Histogram and Product Spectrum: New Methods for Fundamental-Frequency Measurement", *Journal of the Acoustical Society of America*, Vol. 43, No 4, 1968
12. D.R. Stammen, B. Pennycook, "Real-time Recognition of Melodic Fragments using the Dynamic Time Warping Algorithm", *Proceedings of the International Computer Music Conference (ICMC)*, 1993
13. S. Theodoridis, K. Koutroumbas, *Pattern Recognition*, Academic Press, 1998

A Model for the Perception of Tonal Melodies

Dirk-Jan Povel

Nijmegen Institute for Cognition and Information (NICI)
P.O. Box 9104
6500 HE Nijmegen, The Netherlands
Povel@NICI.kun.nl

Abstract. This paper presents a concise description of OPTM, a computational model for the *On-line Processing of Tonal Melodies*. It describes the processes a listener executes when transforming a tone sequence into a tonal melody. The model is based on the assumption that a tone sequence is represented in terms of a chord progression underlying the sequence. Chord tones are directly coded in the harmonic frame, while non-chord tones can sometimes be linked to a subsequent chord tone. The model implements three primary mechanisms: *key finding*, *chord recognition*, and *anchoring*. The operation of the model and its associated output are displayed while the tone sequence is entered incrementally. The output consists of the evolving harmonic framework, the assimilation of non-chord tones, the arising expectations, the tones that do not fit, and an overall indication of the ‘goodness’ of the melodic percept.

1 Introduction

A tonal melody is a sequence of tones that is perceived as well-formed or acceptable within the repertoire of tonal music. Thus, a melody is the result of a perceptual process. It is this process that we have described formally in the OPTM model.

1.1 Background

The OPTM model is founded upon theoretical and experimental research in the perception of tonal music. This research has provided the insight that a listener when listening to tonal music activates a mental framework consisting of a metrical schema and a harmonic schema that serve as the context in which the input is coded [e.g. 1, 2, 3, 4, 5, 6]. This conception implies that the perceptual process has a bottom-up aspect in which the incoming tones are registered and pre-processed, and a top-down aspect in which the listener activates a pertinent metrical and harmonic framework that guides further coding of the input.

The global form and structure of the model was inspired by [7, 8, 9, 10] among others, while its specific architecture is based on experimental work in [11, 12]. The latter research has shown that a tone sequence is perceived as a tonal melody *only* if the listener succeeds in discovering the underlying harmony and if occurring non-chord tones can be linked to closely following chord tones (see section 2.1 for an

example). That research has also shown that the judgment of melodic quality is hardly influenced by characteristics of the pitch-height configuration such as number of contour changes, pitch reversal tendency, mean interval dissonance, mean interval size, or the variation of interval sizes.

These combined findings suggest that the activation of a harmonic framework is an indispensable step in the processing of tone sequences, and, more importantly, that a melody is represented in terms of its harmonic implications, i. e., as a pattern of stabilities, attractions between tones and chords, and expectancies for future events, rather than in terms of properties of the pitch-height configuration. This is a significant result because it indicates that listeners represent a melody in terms of its ‘musical meaning’ rather than in terms of its surface detail, just as a listener will usually only be able to paraphrase the meaning of a spoken sentence but not the actual wording. Independent evidence supporting this conception has been reported in [13].

As the main goal is to describe the perceptual processes that take place while the input sequence is made available incrementally, the OPTM model is designed as a series of processes unfolding in time¹. For this purpose the model is provided with a flexible interface that displays the incremental build up of a mental representation while the sequence unfolds.

The OPTM model is still under construction. At present there are limitations regarding the size and type of sequences that can be processed, and some of the algorithms will need further elaboration and fine-tuning. The overall structure of the program, however, is fairly well established.

1.2 Domain

The domain of the model consists of monophonic (single) tone sequences comprising tones of equal duration and segmented into groups of three tones. The tones in a sequence are specified by the following parameters: *Onset* (moment in time at which the tone is turned on), *Note* (pitch height as a MIDI number); *Duration* (duration of the note in ms); *Velon* (the ‘loudness’ of the note on a scale of 1 – 127); and *Beat* (a Boolean, if *Beat* = true the note is the first of a group or segment). Due to graphical restrictions of the interface the maximum number of tones in a sequence is presently limited to 13.

2 Global Description of the Model

The model accepts as input a sequence of tones upon which a number of processes are executed that simulate the formation of a musical (tonal) representation by a human listener. These processes attempt to discover the *key* of the piece, to establish a *harmonic framework*, and to *link* tones to this framework.

At each moment these processes render a hierarchical structural analysis that describes which notes are integrated in the analysis and which expectations arise about future tones and chords. For some sequences the model will find an appropriate harmonic progression that allows the accommodation of all tones, whereas for other

¹ In this respect the model differs from the Melisma model of Temperley [9] that only arrives at an interpretation after the entire sequence has been analyzed.

sequences the model fails to find a suitable harmonic progression or may not be able to account for all notes. By comparing the output of the model with perceptual judgments of human listeners the veridicality of the model will be examined.

OPTM version 1.4 consists of three primary processes: *KeyFinder*, *ChordFinder*, and *Anchoring*. These processes are described in section 3. A description of the additional processes as well as other details of the system can be found at <http://www.socsci.kun.nl/~povel>.

The model is programmed in REALbasic, an object oriented language for the Macintosh computer (from which applications for the Windows platform can be built). Both the elements (tones, chords, keys, etc.) and the processes are implemented as objects.

2.1 An Example

Figure 1 presents an example of how the results of the model are displayed in the main interface of the system. The interface has been designed with the aim to efficiently enter and select input sequences, and to optimally visualize the operation of the model: the incremental build up of the underlying structure and the coding of the input sequence.

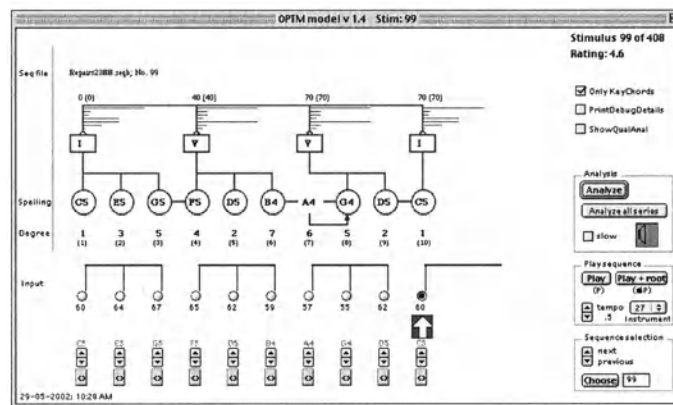


Fig. 1. Illustration of the processing of a sample sequence $\{C_5 E_5 G_5 F_5 D_5 B_4 A_4 G_4 D_5 C_5\}$ on the main interface of OPTM version 1.4. See text for explanation

The interface shows the input sequence in a pseudo-score notation. Underneath each input-note three buttons are placed that serve to alter the sequence and to play a fragment; on the lower right there is a column of buttons for setting 1) analysis parameters, 2) play parameters, and 3) stimulus selection. The actual output of the model, resulting from the above mentioned analyzers *KeyFinder*, *ChordRecognizer* and *Anchoring*, is depicted in the large area above the input (the stepwise construction of the output can obviously not be shown here). It shows the chords that the model yields for the four segments of the sequence (I, V, V, I) and the linking of the chord tones to those chords. Segment 3 (A4, G4, D5) contains a non-chord tone (A4) that is not linked to the chord, but to the subsequent chord tone G4.

The seven thin horizontal lines drawn just above each chord rectangle indicate the expectations for the next chord (represented in terms of transition probabilities from the current chord to each of the chords on the 7 degrees) which are part of the top-down input to the *ChordRecognizer* (see sections 3.1.1 and 3.1.2)

3 Description of the Main Analyzers and Knowledge Bases

3.1 KeyFinder

The function of the *KeyFinder* is to establish the key which supplies the overall musical context and its associated knowledge bases.

The KeyFinder takes as input one or more notes from the first segment, expressed as pitch classes (PC's). A key is represented as consisting of 12 notes, seven diatonic notes and five chromatic notes, each having a certain degree of stability (a measure of the relative prominence of the note in the key). These stabilities are based on the tonal hierarchy profiles collected in experiments reported in [14]. The stabilities of the 12 tones in the major key are: 6.35, 2.23, 3.48, 2.33, 4.38, 4.09, 2.52, 5.19, 2.39, 3.66, 2.29, 2.88; and of those in the (harmonic) minor key: 6.33, 2.68, 3.52, 5.38, 2.60, 3.53, 2.54, 4.75, 3.98, 2.69, 3.34, 3.17. Next, the activation of each of the 24 keys (12 major and 12 minor keys) is determined as the sum of the stabilities of the input notes (represented as PC's) in those keys. In calculating the key the occurrence of a PC is only counted once. The key with the highest activation is selected as the key of the piece.

The algorithm is a simplified version of the algorithm developed by Krumhansl & Schmuckler [3, p. 77–110]. Our version does not take into account the duration of the notes, nor the order of the notes in the segment, both identified factors in key finding. However, the key finder in the present model is not meant to be a robust algorithm as it is only used to get the analysis started. For simple tonal melodies in which the key is clearly established by the first few tones it works well enough. It should be noted that since the key is only based on the notes in the first segment, the model cannot handle modulation.

3.1.1 The KeySpace

As soon as a key has been induced, a *KeySpace* is constructed that describes the diatonic tones (as PC's) and the chords that constitute the key. After, for instance, the key of C major has been recognized, a KeySpace will be configured as shown in Table 1.

Table 1. Key space for the Key of C major

Degree	I	II	III	IV	V	VI	VII
Diatonic tones	0	2	4	5	7	9	11
Major triad	+	+	+	+	+	-	-
Minor triad	-	+	+	-	-	+	-
Dominant 7th	-	+	-	-	+	-	-

The KeySpace represents part of the knowledge that the listener brings to bear in perceiving the input. It is top-down information that helps to determine the degree of a tone, and that supports the process of chord recognition. In a later phase, other chords may be added to KeySpace.

3.1.2 The Piston Table

The PistonTable is another aspect of the knowledge that the listener uses in processing the input. The PistonTable represents the transition probabilities of chords in tonal music as specified in the ‘Table of usual root progressions’ by Piston [15]. Given the root of a chord, it shows the likelihood that it will be followed by one or more other chord(s). See Table 2.

Table 2. The table of usual root progressions from Piston & DeVoto [15]

Chord	Is followed by	Sometimes by	Less often by
I	IV or V	VI	II or III
II	V	IV or VI	I or III
III	VI	IV	I, II or V
IV	V	I or II	III or VI
V	I	VI or IV	III or II
VI	II or V	III or IV	I
VII	III	I	-

A Table of Root-transition-probabilities is derived from the PistonTable. Like KeySpace, the PistonTable plays an important role in the process of chord recognition. It is a top-down constraint that affects the activation of chord candidates in the *ChordRecognizer*. It specifies the chance that some chord is followed by some other chord. As such, the expectations that are derived from this table are based on a first-order approximation in which only the preceding chord is taken into account. This approach is clearly rather limited as a typical listener probably creates expectations based on longer chord progressions (e.g. cadences). Therefore, in future developments, we intend to base the harmonic top-down constraints on more sophisticated theoretical notions like those proposed in [8, 16, 17].

3.2 ChordRecognizer

3.2.1 Introduction

ChordRecognizer is the most important analyzer in the model; it is also by far the most complex one. Its purpose is to establish the harmonic interpretation of a segment, thereby providing the frame of reference for that segment. The main problems encountered in chord identification are:

1. to determine the window in which one is going to search for a chord. Depending on the partitioning of a sequence different harmonic analyses will tend to be generated, as shown in Figure 2.



Fig. 2. Different harmonic analyses for a melodic fragment depending on the applied partitioning

2. the fact that in a key a variety of chords (each consisting of 3 or 4 tones, if we reckon the most common chords) are formed using only a very limited (7 or 8) number of PC's. This results into considerable overlap between the constituents of different chords leading to serious identification problems. See Figure 2.
3. the overlap problem is often aggravated by the phenomenon of underspecification: not all elements of a chord are played but a listener still identifies a chord.
4. not all tones within the identification window belong to one chord; i.e., non-chord tones occur that are usually anchored to chord tones (passing tones, neighbor tones, appoggiatura's).

To solve the chord identification problem as a bottom-up problem without using the segmentation of the sequence induced by meter is virtually doomed to fail (see however [18]). Only if the chord recognition process is guided by the segmentation of the sequence (effected by the meter), and if the expectations for specific chord progressions suggested by the key and the chords already given are taken into account, it will appear feasible to arrive at a veridical harmonic interpretation. Moreover we believe that the human listener also works along these lines.

For these reasons, the present chord identification algorithm uses a) the successive segments as windows within which chords are being identified, and b) the expectations for the current harmony derived from the Piston Table associated with the current key. The successive steps in *ChordFinder* are discussed below.

3.2.2 Steps in the Chord Recognition Process

3.2.2.1 Step 1: Bottom-Up Processing: Activation of Chord Templates

First the notes in the segment are converted into Pitch classes and mapped on four types of chord templates: major triad, minor triad, dominant seventh chord, and diminished seventh chord. These templates can be represented on a chromatic circle comprising all 12 pitch classes as shown in Figure 3.

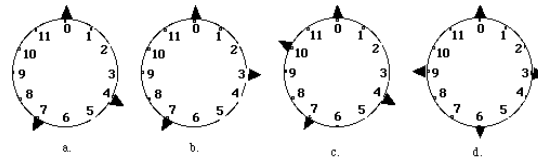


Fig. 3. Templates for a Major triad (a), a Minor triad (b), a Dominant seventh chord (c), and a Diminished seventh chord (d).

The bottom-up activation of each of the chord types is calculated by establishing to what degree the input activates each of the different instances (e.g. C major, C# major, D major etc.) of the four chord types. In terms of the above circles, this can be represented as rotating the input, also displayed on a chromatic circle, around a chord template and determining at each step the sum of the weights of the elements in the template that correspond to the notes in the input. See Figure 4.

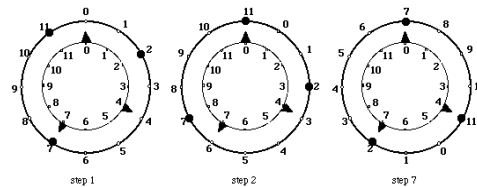


Fig. 4. Three stages in the process of establishing the activation of a major triad by rotating the input (PC's 2 7 11) around the major triad template: 1) activation of C major; 2) activation of C# major; 3) activation of G major. G major (PC 7) yields the highest activation, because all tones in the input correspond to elements in the template.

The code that calculates the activation is:

```

for j = 0 to 11           // rotate in 12 steps
  for i = 1 to Length    // number of elements in the input
    Activation(j)
      =Activation(j)+Template(((PCsInFragment(i)-j)+12)Mod12)
  next
next

```

First the net activation is computed. This activation is caused by the unique PC's in a segment, i.e. irrespective of the number of times a PC occurs in a segment. Total bottom-up activation is defined as a function of the net activation, the number of repeated chord tones, and the number of non-chord tones according to formula (1).

$$\text{BottomUpActivation}(i) = \text{NetActivation}(i) + \text{NrofRepeatedChordTones}(i) * .5 \quad (1) \\ + \text{NrofNonChordTones}(i) * .1$$

It should be noted that a dominant-seventh chord is only activated if the seventh is present, and that a diminished seventh chord is only activated if both the root and the seventh are present. In a later stage, other chords (e.g. minor seventh, half diminished seven, etc.) may be added, making the selection process inevitably more difficult.

3.2.2.2 Step 2: Incorporating Top-Down Information

Next, top-down information is added to the bottom-up information of all chord instances of all types. Specifically, the total activation is calculated by multiplying the bottom-up activation with the expectation for the chord derived from the Piston Table. In this way the Total Activation for all chords is obtained.

3.2.2.3 Step 3: Determining the Maximally Activated Chord

In this step the chord(s) with the maximal activation, are searched for by finding the chord(s) with the highest activation. These chord candidates are put in the list MaxChords. If this list contains more than one chord, which occurs rather frequently due to the small number of tones in a segment, further steps are taken to arrive at a unique identification. If it only contains one chord, this is the chord underlying the current segment.

3.2.2.4 Step 4: Towards the Identification of a Single Chord

If MaxChords contains more than one chord, it is determined whether there are non-chord tones within that chord interpretation and whether these can be anchored (see below) to a succeeding tone in the segment. If there is a non-chord tone that can *not* be anchored, that chord will be removed from MaxChords. In virtually all cases MaxChords will then contain only one chord candidate: the *Recognized Chord* representing the harmonic interpretation of the current segment.

3.3 Anchoring

The perceptual mechanism of *Anchoring* was first described in [7] that also specified the circumstances needed for a non-chord tone to be anchored to a subsequent chord tone. Two variants of the analyzer *Anchoring* have been implemented. The first, *Anchoring1*, takes a non-chord tone as input and determines if one of the following tones in the segment is a chord tone at an interval of less than 2 semitones. If this is the case the non-chord tone is anchored to that chord tone. Figure 5 shows two examples of this type of anchoring: in the first example the F# is anchored to its neighbor G, while in the second example the A is anchored to the G (named ‘delayed’ anchoring [7]).

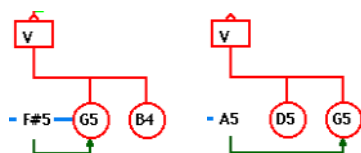


Fig. 5. Two examples of Anchoring1. See text for explanation.

The other variant, *Anchoring2*, determines whether an unanchored nonchord tone in the previous segment can be linked to a current chord tone. If this is the case the two tones will be linked. Figure 7 shows two examples.

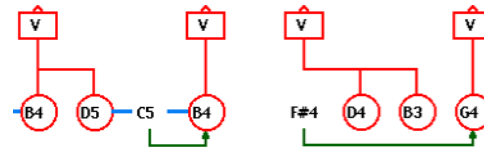


Fig. 6. Two examples of Anchoring2. See text for explanation.

A note can only be anchored once, so that the A in the segment $\{A_4 B_4 G_4\}$, (underlying harmony G) will anchor to the B and not to the subsequent G.

4 Output: The Percept

The output generated by the model is expressed in the class *Percept*. This class is viewed as representing aspects of the mental representation that has been formed during the processing of the input. These include perceptually relevant characteristics associated with the representation that could show up in listener reports, and that will be used to test the model. The specific contents of Percept are:

1. The *structural description* displayed in the main window, showing
 - a. the harmonic progression assigned to the melody
 - b. The chord tones and non-chord tones
 - c. Which non-chord tones are/are not anchored
 - d. The expectations arising at different points in the sequence
2. A number of properties derived from the structural description, namely:
 - a. The progression strength (the sum of the expectations of the successive chord transitions, indicating how 'regular' or expected the harmonic progression is)
 - b. The number of anchored non-chord tones
 - c. The number of non-anchored non-chord tones
 - d. Progression is hampered (a Boolean which is 1 if ChordRecognizer has failed at some point)
 - e. Sequence ends on beat (a Boolean)
 - f. Sequence ends on I or VI (a Boolean)
 - g. Finally, a quantitative overall score *Goodness* is computed as a weighted sum of the features a - f above.

4.1 A Few Examples

Figure 7 shows the output of OPTM 1.4 for three selected sequences. Sequence 1 is a well-formed sequence with one anchored non-chord tone in the penultimate segment. Sequence 2 is an ill-formed sequence (at least the second half) for which the model indeed does not succeed in creating an appropriate analysis: there are two unconnected tones, the assigned chord progression is uncommon, the last tone is not an element of I or VI, and the sequence does not end on a down beat. Sequence 3

shows that the model is unable to recognize the (rare) modulation from C-major to Eb-major, as a result of which it generates a highly inappropriate parsing.

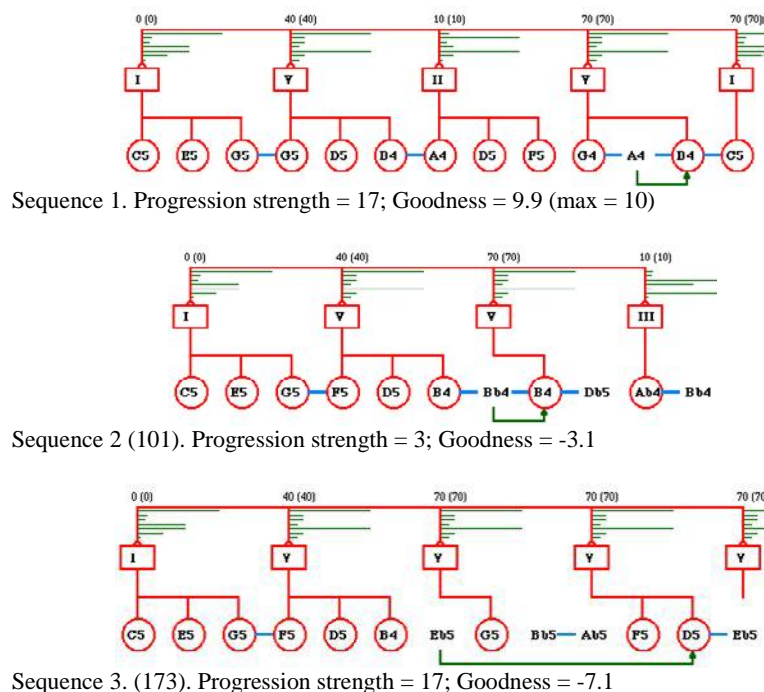


Fig. 7. The output of OPTM 1.4 for three sample sequences. See text for explanation.

5 Outlook

The present form of the OPTM model was obtained by adjusting the parameter values and the interaction between the modules until the performance of the model approximated that of a human listener. Although the model at present appears to work reasonably well, it still needs to be tested formally.

The various types of output that the model generates, listed in section 4, yield ample possibilities for testing: predictions from the model can be compared to the responses of human listeners. Currently we are in the process of testing the model using 1) existing tunes and melodies selected from <http://www.themefinder.org/>, and 2) a set of over 400 sequences, collected in a study in which experienced musicians completed unfinished melodic fragments, that greatly differ in the degree of well-formedness. Proceeding in this way, we expect to gain a reliable evaluation of the model as well as suggestions for future improvements.

Acknowledgment. I thank Erik Jansen and Olivier Povel for textual suggestions that have considerably improved the readability of this paper.

References

1. Bharucha, J. J.: Music Cognition and Perceptual Facilitation: A connectionist Framework. *Music Perception* 5 (1987) 1-30
2. Cuddy, L. L., Cohen, A., Mewhort, D. J.: Perception of structure in short melodic sequences. *Journal of Experimental Psychology: Human Perception and Performance* 7 (1981) 869-883
3. Krumhansl, C.L.: *Cognitive Foundations of Musical Pitch*. Oxford University Press, New York (1990)
4. Lerdahl, F., Jackendoff, R. *A Generative Theory of Tonal Music*. MIT Press, Cambridge MA (1983)
5. Longuet-Higgins, H. C., Steedman, M. J.: On interpreting Bach. In B. Meltzer, D. Michie (Eds.) *Machine Intelligence*. Edinburgh University Press, Edinburgh (1971)
6. Povel, D. J.: Internal Representation of Simple Temporal Patterns. *Journal of Experimental Psychology: Human Perception and Performance* 7 (1981) 3-18
7. Bharucha, J. J.: Anchoring Effects in Music: The Resolution of Dissonance. *Cognitive Psychology* 16 (1984) 485-518
8. Lerdahl, F.: Tonal Pitch Space. *Music Perception*, 5 (1988) 315-350
9. Temperley, D.: *The Cognition of Basic Musical Structures*. MIT Press, Cambridge MA (2001)
10. Zuckerkandl, V. *Sound and Symbol*. Princeton University Press, Princeton (1956)
11. Povel D. J., Jansen, E.: Perceptual Mechanisms in Music Perception. *Music Perception* 19 (2001) 169-199
12. Povel, D. J., Jansen, E.: Harmonic Factors in the Perception of Tonal Melodies. *Music Perception* (2002) Fall issue
13. Sloboda, J. A., Parker, D. H. H.: Immediate Recall of Melodies. In P. Howell, I. Cross, R. West (Eds.) *Musical Structure and Cognition*. Academic Press, London (1985)
14. Krumhansl, C. L., Kessler, E. J.: Tracing the Dynamic Changes in Perceived Tonal Organization in a Spatial Representation of Musical Keys. *Psychological Review* 89 (1982) 334-368
15. Piston, W., Devoto, M.: *Harmony*. Victor Gollancz, London (1989 originally published 1941)
16. Sutcliffe, T.: Syntactic Structures in Music. <http://www.harmony.org.uk/>
17. Steedman, M. J.: A Generative Grammar for Jazz Chord Sequences. *Music Perception* 2 (1984) 52-77
18. Pardo, B., Birmingham, W. P.: *The Chordal Analysis of Tonal Music*. The University of Michigan, Department of Electrical Engineering and Computer Science Technical Report CSE-TR-439-01 (2001)

Control Language for Harmonisation Process

Somnuk Phon-Amnuaisuk

Faculty of Information Technology, Multimedia University, Malaysia.
somnuk.amnuaisuk@mmu.edu.my

Abstract. An approach to the automatic harmonisation of chorales is described. This involves using musical rules about this style of music, and controlling the use of these rules when searching for a harmonisation. To gain more control over the search process, control knowledge is explicitly represented, allowing a hierarchical decomposition of the harmonisation process. When the control knowledge is made explicit, the control structures can be modified with ease and flexibility. When the control is hierarchically structured, the effects of its application are clear. Our explicitly structured control model provides flexibility, and automatically generates natural harmonisations. In this paper, we present our control language and some harmonisation outputs to show the flexibility of the language.

1 Introduction

Harmonising Bach chorales is a hard AI problem, comparable to understanding natural language. In this class of problem, solutions must be both correct (syntactically well-formed) and appropriate to the contexts (semantically sound). There seem to be no fixed universal algorithms for this class of problem. A search control mechanism with heuristics seems to be a more effective approach than rigid algorithms since search can be guided and controlled during the search process.

In general, the symbolic AI approach allows reasoning to be examined via the manipulation of formal representations. The control of this reasoning can then itself be represented. This approach allows the systems to be designed with useful control features. Our explicit control of heuristic search (for the harmonisation process) is based on this intuition. Rules such as those giving the permissible cadence forms for this style we call *object rules*, in contrast with control knowledge which describes how the object rules should be used.

In this paper, we investigate the use of explicit control in chorale harmonisation in the style of J. S. Bach. The aim here is to devise a control language which allows us to conveniently and flexibly carry out the task of harmonisation. We can imagine a spectrum of languages which at one end is the natural language that we use to communicate with others, and at the other end is a low level language only suitable for communication with computing machines. Thus our aim is to devise an intermediate language which is readable and flexible, yet can effectively exploit the domain knowledge of harmonisation rules.

The paper proceeds as follows: in section 2 we describe related work on this topic; in section 3 we present our approach to the problem, and our version of explicit control; section 4 gives examples that illustrate the flexibility that follows from use of explicit control; we finally draw conclusions.

2 Related Work

There have been many attempts at modelling the behaviour of chorale harmonisation before; the CHORAL system ([1],[2]) and the HARMONET system ([4]) are two such examples. However, neither of these systems uses explicit control; as a result they are both hard to adapt or extend.

Domain knowledge in the CHORAL system is implemented using a backtracking specification language (BSL). The program is implemented in three main sections: the chord skeleton process, the fill-in process (which uses four different views or representations: fill-in view, melodic string view, merged melodic string view and time slice view) and the Schenkerian analysis. The harmonisation process for each melody note is a cycle of processes from the chord skeleton process, the fill-in process and the Schenkerian analysis. The harmonisation is carried out from left to right. Control knowledge is not made explicit but embedded in the program structure. To alter harmonisation behaviour, changes in the program are required. The changes must be made in the program at the object-level which may require modifications to the whole program. This form of control is inflexible and hard to modify.

The *HARMONET* system is a hybrid between a neural network and a rule-based system. The representation at the neural network level encodes the following information:

- Melodic context (i.e. soprano input)
- Function of chords: harmonic description of chords and their inversions.
- Relative position to the start of a phrase.
- Stress information in a bar

The harmonisation process in *HARMONET* involves interaction between the neural network part and the rule-based part. First, a chord and a bass prediction are given by the neural network part, then the rule-based part of the system fills in the inner voices and adds quaver note decorations. The harmonisation process starts from the left and moves to the right.

The neural network part behaves like a black box. Little can be done to control the process inside neural network. Hence, there is not much we can do if we do not like the solutions offered by the neural network. There is no backtracking mechanism in the neural network model. Connection weights between neurons are fixed after the net is trained. The neural network must be re-trained if a different behaviour is desired.

3 Methods

There are two relevant dimensions of musical knowledge: *musical structure* and *musical process*. Musical structure is concerned with the abstract structural prop-

erties of music. Musical process provides the dynamic dimension in music (see Smoliar [8]). Sloboda [6] suggests that the musical process carries an implicit part of knowledge which is not transparent from the structure of finished products. Part writing technique is an example of musical process. It reveals powerful heuristics for writing a harmonised chorale, and this knowledge is not normally apparent in the finished chorales. In our work, score representation and object rules fall in the musical structure category; the harmonisation process, described by our control language, falls in the musical process category.

3.1 Domain Knowledge Representation

Knowledge representation in our system falls into two main components: score representation and control language.

- The score represents musical information in two main components: *music materials* and their *interpretations*. The musical materials section captures the domain in the concept of notes and their associated properties. The interpretation section captures an interpreter’s beliefs and understanding of the domain in concepts of musical structures and compositional processes. These concepts are declarative relationships formed between different parts of the score (of whatever size). Different interpretations of the same material yield different viewpoints of the knowledge content.
- Our control language consists of two main parts: *atomic control definitions* and *compound control structures*. The atomic control definitions are classified under three main classes (i) atomic rules which take a data stream as their input and give a data stream as their output, (ii) atomic tests which take a data stream input and give a Boolean output, and (iii) atomic measure which take a data stream input and output numerical measurement. Compound control structures are built up hierarchically from atomic definitions using a set of *control primitives* that we supply. The control primitives allow expressions of control flows such as sequencing; branching (with biased probabilities or preferences at branching points); and repetitions of control structure. The control primitives also allow the constraining and prioritising of the input data stream (see examples in section 4.2).

Score representation. The score has two parts: *music material* and *music interpretations*. The musical material part is based on two basic structures: *lines* and *note events*. The line is a list of consecutive note events. The note event is constructed from the pitch notation based on the conventional western notations and time (see Wiggins *et al.*, [9], Harris *et al.*, [3], Smaill *et al.*, [7]). The pitch notation describes a pitch as it should be read from the stave. The time describes the notated musical time. The musical material part of the score is required to at least capture all the information available from the conventional western music notations. It is then possible to use this information in further processing to obtain realised sound events, or to reason about other properties of the musical materials.

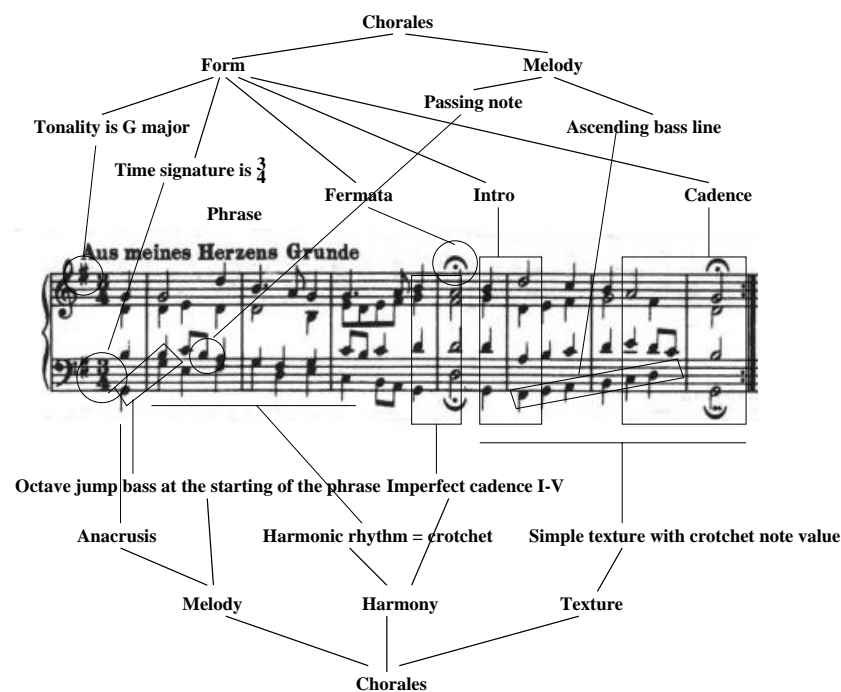


Fig. 1. Representation of musical information

The interpretation part captures knowledge of the chorale being harmonised. Knowledge in this part expands as the harmonisation progresses. Initially, the music material of the score contains only the input soprano line; and the interpretations of the score consist of information such as phrase structure, time signature, key signature, chorale title, etc. (see Figure 1). We summarise our score representation below:

```

Score          : score(music Material, interpretations)
Music Material : line(name, list of note events)
Interpretations : object(name, list of attributes)
Note event     : event(pitch, time, feature slots)
Pitch          : {1 2 3 4 5 6 7} × {nat # b bb x} × integer (e.g. [1,nat,3])
Time           : integer × integer
Feature slot   : list of attributes
Attribute      : type × name × value (e.g. form:id:chorale)

```

At the beginning, the interpretations contain minimum information. During the harmonisation process, more interpretations may be added and modified until the satisfactory solution is reached. Snapshots of interpretation attributes (type:name:value) at the beginning of the harmonisation process are given below.

- `form:id:chorale` means the object is identified as a chorale object.
- `form:ksig:'G major'` means key signature is in the key of G major.
- `form:tsig:34` means time signature is in $\frac{3}{4}$ time.
- `form:first_strong_beat:anacrusis(8)` means anacrusis duration has the value of a crotchet note.
- `form:phrase_length:[0,96]` means the phrase has an interval span between the time of 0 to 96.
- `harmony:harmonic_rhythm:crotchet` means anacrusis has the duration of a crotchet value.

Control definitions: Atomic rules, atomic tests and atomic measures manipulate the knowledge content in our score representation. We prefer these operations to operate at a suitable grain size of the knowledge content (e.g. at a desired conceptual level). Lower level operations are encapsulated with lower level control and they are hidden below the atomic rules, atomic tests and atomic measures. In other words, we abstract away unnecessary detail and complexity below the grain size we are interested in. These atomic control definitions are combined to form control definitions (which are compound atomic control definitions, see examples in sections 4.1, 4.2). Examples of atomic control definitions in our system are given in Figure 2.

Compound control definitions: We define a set of *control primitives* which allows us to express simple control behaviours by combining atomic rules, atomic tests and atomic measures together. Thus, the composition process may be described by this configuration. The compositional process reflects the thoughts relevant to musical tasks. Naturally, hierarchical structure exists in the process, for example, harmonic implication is usually realised before actual voicing. In trying to model the compositional process, it is inevitable that the system must support the expression of hierarchical structure of tasks and flow of tasks. Our control language offers control primitives such as **then**, **if-then-do**, **repeat**. The control language also deals with declarative reading in the control features such as the **branch-with**, **n_score-size**, **sort_score** primitives. The control primitives implemented in our system are displayed in the appendix.

Space precludes a full description of the semantics of these control primitives, but here is an illustration of how they work. We suppose the input is a score and a control expression; behaviour is defined in terms of a (possibly infinite) stream of solutions, each of which is a score derived from the input score using the specified control.

Thus, the behaviour of the control C_1 **then** C_2 on an input score S is to return the stream

$$S_{11}, S_{12}, S_{13}, \dots, S_{21}, S_{22}, S_{23}, \dots$$

where the control C_1 returns S_1, S_2, \dots , and C_2 returns $S_{11}, S_{12}, S_{13}, \dots$ on input $S_1, S_{21}, S_{22}, S_{23}, \dots$ on input C_2 , and so on.

For another example, the control **filter** C **with** T returns a sub-stream of the result of using C , namely the stream

$$S_{n_1}, S_{n_2}, S_{n_3} \dots$$

Atomic control definitions	Options
atomic rule definitions selectPhrase(Optional) outlineChord(Optional) outlineBass(Optional) outlineAlto(Optional) outlineTenor(Optional) fill(Optional)	outlineAlto, outlineTenor, outlineBass intro, body, cadence, transition intro, body, cadence, transition transition transition outlinePitch, neighborNote, passingNote, resolveStepDown, suspension, holdSuspension, etc.
atomic test definitions constrain(phraseHarmonicPlanOutline) constrain(tritoneSpan,Optional) constrain(doublingLeadingNote,Optional) constrain(cadenceLeadingNote,Optional) constrain(rhythmicPattern,Optional)	n/a alto, tenor, bass alto,tenor,bass alto,tenor,bass global
atomic measure definitions preferredBassSuspension spacingInnerVoices linearProgression(Optional) preferredRhythmicPattern(Optional) stepProgression(Optional)	n/a n/a alto, tenor, bass bass alto, tenor, bass

Fig. 2. Examples of atomic control definitions

where S_1, S_2, S_3, \dots is the stream produced by C , and $S_{n_1}, S_{n_2}, S_{n_3} \dots$ are the scores for which the test T succeeds.

The choice of control primitives was motivated by the particular musical task at hand; they are more widely applicable, however. Musical knowledge comes in where this language is used to specify particular control definitions for a musical task, as we see in the next section.

4 Results

A control structure written using our control language is a program at the meta-level (the syntax of our control language is illustrated in the appendix). Search control is explicit to users at this level. Users can modify and change control behaviours with ease and flexibility (in contrast to search control at the object-level). In this section, we illustrate flexibility features of our control language.

4.1 Flexibility in Structuring Harmonisation Process

Our control language allows users to construct a variety of control structures according to different harmonisation strategies. For example, the top level har-

monisation process may be accomplished by completing all the voices in each melody position before moving to the next note. On the other hand, the harmonisation process may be accomplished by determining the harmonic progression of the whole phrase (or even the whole piece) before deciding which bass notes to put in, and then fill in the rest of the voices. These are just two plausible strategies. The harmonisation process in the CHORAL system is hard coded with the first strategy. Our control language allows us to realise both strategies easily.

The control structure below is similar to the second strategy mentioned earlier¹. This control structure is used to produce the harmonisation output of the fourth phrase of chorale ‘*Aus meines Herzens Grunde*’² (see the harmonisation result in Figures 3).

```
Control fillinChoralStyle is
  repeat {
    rule:selectPhrase(fillinProcess)
  then
    ( filter outline_phrase_harmonic_plan
      with ( test:constrain( globalHarmonicPlanOutline )
        and test:constrain( phraseHarmonicPlanOutline )))
  then outlineBass
  then rule:display(harmonisationResult)
  then outlineInnerVoices
  then rule:display(harmonisationResult)
  then rule:initializePhrase(fillinProcess)
  then fillinView
  then rule:closePhrase(fillinProcess)
  then rule:display(harmonisationResult)
  }
then rule:export2MIDI.
```

Another example of the same kind is illustrated using the first two phrases of Bach’s chorale *Ach Gott, wie manches Herzeleid*, No. 217. The first output is from control definition #2 and the second output is from control definition r217³. The control definition r217 is the same as control definition #2, except for the fill-in section which has now been constructed to fill in detailed decorations..

4.2 Flexibility in Modifying Control Structures

Our control language allows users to modify the earlier control structure to adjust a near solution towards a desired solution. This is a very useful feature. In this section, we give various examples to illustrate the idea.

¹ This excerpt is taken from the control definition #1, which is available online at <http://www.dai.ed.ac.uk/~somnukpa/controlDef/node1.html> .

² All chorale melodies used in this paper are based on the collection in Riemenschneider [5].

³ Control definitions #2 and r217 are available online as above.

Phrase 4 'Aus meines Herzens Grunde'

$G:I$ I IV IV V V ii vi $e:V$ V i

I I IV_b^7 IV I_b V V^7 vi $e:V$ V i

I I IV_b^7 IV I_b V V^7 vi $e:V$ V i

Fig. 3. Example of a control structure in use

Filter out unwanted solutions using test. The first example is an instance of the harmonisation of phrase 8 and 9 of Bach's chorale: *Ich dank' dir, Gott für Wohltat*, No. 223. Suppose we want to move the tenor part higher up (see example marked 'before' in figure 6. We may wish to control this by constraining the output score with a test *property(lowerBound(tenor), [5, nat, 3])*. This is done by simply modifying the original control block using **filter-with** primitive (i.e. **filter** <Original_control> **with** <test_definition>).

This means that the solution with the tenor part below the G below middle C is filtered out (see figure 6, 'after'). The same property may be diluted as a soft constraint with a measure (instead of a test). In such a case, solutions with the tenor line below the bass clef E3 will not be absolutely rejected. The output will be ranked and the least flawed one would be selected. This allows more flexibility in applying heuristics which are not just black or white, but grey.



Fig. 4. Control Definition #2



Fig. 5. Control Definition r217

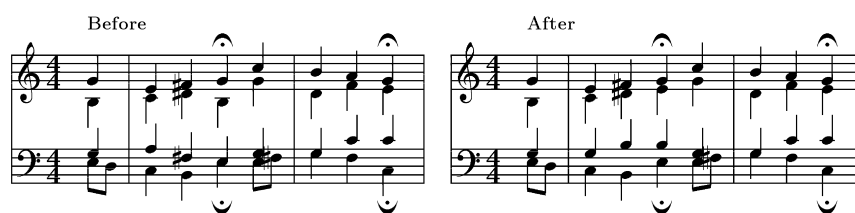


Fig. 6. Modifying a solution with constraint

Reprioritise input data stream with `n_score` and `sort`. The following two harmonisation examples (from the Bach chorale *Aus meines Herzens Grunde*)

are harmonised with two different control structures (control definitions #1 and r001)⁴. In this discussion, let's focus on the first phrase and the fifth phrase of both outputs. We include sections of control structure from both control definitions below. Basically they share the same skeleton; however, the control structure in definition r001 uses **n_score** and **sort** to reprioritise the input data stream. A stream of input scores are re-ordered according to the sorting criteria (defined with measure definitions). The partial solutions with wider spans between the tenor and the bass voice are placed in front of the stream. As a result, the span between the bass and the tenor voice is increased, as can be seen by comparing the results in figures 7 and 8.

% excerpt from control definition #1 (use for all phrases)

```
Control fillinChoralStyle is
  repeat {
    rule:selectPhrase(fillinProcess)
    then ( filter outline_phrase_harmonic_plan
          with ( test:constrain( globalHarmonicPlanOutline )
                and test:constrain( phraseHarmonicPlanOutline )))
    then outlineBass
    then outlineInnerVoices
    then fillDecorations
  }
```

% excerpt from control definition r001 (use for phrase 1 & 5)

```
Control fillinChoralStyle is
  rule:selectPhrase(fillinProcess,phrase-1)
  then ( filter outlinePhraseHarmonicPlan
        with ( test:constrain( globalHarmonicPlanOutline )
              and test:constrain( phraseHarmonicPlanOutline ) ) )
  then outlineVoices
  then fillDecorations.
```

```
Control outlineVoices is
  n_score outlineBass size 3
  then sort_score measure:measure( property(linearOutlineBass) )
  then n_score outlineInnerVoices size 20
  then sort_score ( measure:measure( property(spacingInnerVoices) )
    and measure:measure( property(linearOutlineAlto) )
    and measure:measure( property(linearOutlineTenor) ) )).
```

5 Conclusion

In programs where the control knowledge is not explicitly represented, control information is mixed with the procedural flow of the program. The main drawback

⁴ Control definitions #1 and r001 are available online as above.



Fig. 7. Control definition #1



Fig. 8. Control definition r001

of this approach is that the mixed control does not allow easy understanding of the control regime and does not easily allow control reconfiguration or modification when a new control structure is required. In our approach, the control (i.e. exploitation of control definitions at the meta-level) and the domain are explicitly represented. Reconfigurations and modifications of control parameters are easy as a result of expressive primitives and modular structure of the control definitions. This allows greater flexibility in expressing the control strategy over the search space.

This research has extensively investigated the modelling of a harmonisation process using our explicitly structured control model. It is apparent from our experiments that apart from a sufficient amount of domain knowledge, the system must also possess the right control in order to deliver a good solution. This factor makes our approach more attractive than other unstructured search approaches such as Genetic Algorithms.

We have illustrated in the examples that control structures can be changed with ease and with flexibility. Different harmonisation outputs can be obtained from different control structures, from the same object rules. Our control language allows control definitions to be plugged into and out of the control structure without having to worry about input and output parameters for each control definition. The variety of our primitives seem to be rich enough to support different control strategies.

Since human composers do not merely put their musical ideas together randomly, each work has its own character which holds the music together. We believe that to successfully model human musical expertise in machines, a direct access to control these details is essential. We suggest that our problem solving approach offers many attractive features in constructing a machine model of complex musical expertise.

References

1. Kemal Ebcioglu. Report on the choral project: An expert system for harmonizing four-part chorales. Technical report, IBM, Thomas J. Watson Research Center, March 1987.
2. Kemal Ebcioglu. An expert system for harmonizing four-part chorales. In M. Balaban, K. Ebcioglu, and O. Laske, editors, *Understanding Music with AI: Perspectives on music cognition*, chapter 12, pages 294–333. The AAAI Press/The MIT Press, 1992.
3. M. Harris, G. Wiggins, and A. Smaill. Representing music symbolically. In Antonio Camurri and Corrado Canepa, editors, *Proceedings of the IX Colloquio di Informatica Musicale*, 1991. Also Research Paper 562, Department of Artificial Intelligence, University of Edinburgh.
4. H. Hild, J. Feulner, and W. Menzel. HARMONET: A neural net for harmonizing chorales in the style of J.S. Bach. In R. P. Lippmann, J. E. Moody, and D. S. Touretzky, editors, *Advances in Neural Information Processing 4*, pages 267–274. Morgan Kaufman, 1991.
5. Albert Riemenschneider. *371 Harmonized Chorales and 69 Chorale Melodies with Figured Bass*. G. Schirmer, Inc, 1941.

6. John Sloboda. *The Musical Mind: The Cognitive Psychology of Music*. Oxford University Press, 1985.
7. A. Smaill, G. Wiggins, and E. Miranda. Music representation – between the musician and the computer. In M. Smith, G. Wiggins, and A. Smaill, editors, *Music education: an Artificial Intelligence Perspective*, London, 1993. Springer-Verlag. Also Research Paper 668, Department of Artificial Intelligence, University of Edinburgh.
8. Stephen W. Smoliar. Process structuring and music theory. In S. M. Schwanauer and D. A. Levitt, editors, *Machine Models of Music*, chapter 9, pages 188–212. The MIT Press, 1993.
9. G. Wiggins, M. Harris, and A. Smaill. Representing music for analysis and composition. In *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI'89)*, 1989. Also Research Paper 504, Department of Artificial Intelligence, University of Edinburgh.

Appendix: Syntax of the Control Language

```

<control_spec>      ::= definition(<name>,<control_definition>)
<control_definition> ::= <atomic rule>
                    | <control_definition> then <control_definition>
                    | <control_definition> or <control_definition>
                    | if <test_definition> then_do <control_definition>
                    | if <test_definition> then_do <control_definition>
                      else_do <control_definition>
                    | first_score <control_definition>
                    | branch [{ control(<control_definition>,<Pr>) }+]
                    | filter <control_definition> with <test_definition>
                    | n_score <control_definition> size <number>
                    | sort_score <measure_definitions>
                    | repeat <control_definition>
                    | backtrackRepeat <control_definition> with <test_definition>
                    | interactive <control_definition>
<test_definition>   ::= <atomic test>
                    | <test_definition> and <test_definition>
                    | <test_definition> or <test_definition>
                    | not <test_definition>
<measure_definition> ::= <atomic measure>
                    | <measure_definition> and <measure_definition>
<name>              ::= Prolog atoms
<number>            ::= integer
<Pr>                ::= real value between 0.0 and 1.0

```

Evaluating Melodic Segmentation

Christian Spevak, Belinda Thom, and Karin Höthker*

Universität Karlsruhe, Institut für Logik, Komplexität und Deduktionssysteme,
Am Fasanengarten 5, 76128 Karlsruhe, Germany
{spevak,hoethker}@ira.uka.de, bthom@cs.cmu.edu
<http://i11www.ira.uka.de/~musik>

Abstract. We discuss why evaluating melodic segmentation is difficult when solutions are ambiguous and explore the nature of this ambiguity using a corpus of melodies hand-segmented by musicians. For each melody, the musicians identified different, musically coherent solutions, suggesting that ambiguity should be modeled when assessing how “well” an algorithm segments the same melody. We propose a probabilistic framework for modeling ambiguity that integrates both the segment boundaries and the lengths that the musicians preferred. The framework gives rise to several potential extensions of existing segmentation algorithms.

1 Introduction

Segmenting or parsing a melody amounts to 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 that do not belong 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* that contains a series of non-overlapping, contiguous fragments. An algorithm that automatically produces a “musically reasonable” segment stream would provide an invaluable tool for melodic modeling. For example, it could perform an important preprocessing step, dividing up a melody into “locally salient chunks” before considering higher-level melodic features.

Although we are unaware of any past research that directly supports this claim, melodic segmentation is most certainly an ambiguous affair. Ambiguity arises because it is typically not possible to determine one “correct” segmentation for a melody. Rather, the process is influenced by a rich and varied set of contexts, where local structure (gestalt principles), higher-level structure (e.g. recurring motives, harmony, melodic parallelism), style-dependent norms and the breaking of these norms all have an impact on what is perceived as “salient.” A two-fold goal drives this research. First, we want to explore qualitatively how this

* Author ordering was determined by stochastic simulation.

ambiguity manifests itself in different musical situations. Second, we strive to provide a framework for quantifying ambiguity.

We began our research by conducting a study in order to observe the solutions we would get when different musicians were asked to segment a fixed melodic corpus. This data collection and its qualitative evaluation are described in Sects. 2 and 4, respectively. In Sect. 3, we present three existing segmentation algorithms, which set the stage for considering the systematic evaluation of an algorithm’s performance. In Sect. 5, we introduce three increasingly complex approaches for quantitatively assessing a solution in the context of an ambiguous set of reference segmentations. These approaches address the problem of identifying how “appropriate” a segmentation is rather than just identifying whether a segmentation is “correct.” Since musically plausible segmentations tend to exhibit higher-level structure, such as a certain range of lengths, we incorporate it explicitly, obtaining more complex, yet more plausible, evaluation measures.

2 Manual Segmentation

We began exploring the task of melodic segmentation by collecting data from a number of musicians. We compiled a small corpus of ten melodic excerpts in different musical styles. Some excerpts were selected because of their ambiguous musical structure (e.g. excerpts from a Bach fugue and a Haydn string quartet), others were selected because of their noticeable lack of structure (an excerpt from Wagner’s *Parsival*). Four folk songs from the *Essen* collection (Schaffrath 1995) were included in order to explore how ambiguous a “simple” folk tune might be. Finally, the corpus included an improvisation over a Bluegrass standard, a Pop song, and a Bebop jazz solo. For each excerpt, we provided a minimal score representing the monophonic melody (including meter) and a “dead-pan” MIDI file. Additional information, such as accompaniment, harmony, lyrics, and origin, was withheld as segmentation algorithms typically do not use such information.

Twenty-two musicians with various levels of expertise – including professionals, musicologists, music teachers, and amateurs – were asked to identify “salient melodic chunks” for each excerpt. Subjects were instructed to identify boundaries at two different levels of granularity, which we called the *phrase level* and the *sub-phrase level*. Sub-phrases 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.

3 Algorithmic Segmentation

We now present three existing segmentation algorithms and consider how they have been evaluated. A key aspect of each algorithm is its output representation, for it has an impact on how ambiguity might be considered when evaluating an algorithm’s performance. A segmentation algorithm’s most basic input is a melodic line, encoded by a *note list* comprised of note elements. Variable n refers to the number of notes contained within the list. Note elements are temporally ordered, each defined by a discrete pitch, an onset, and an offset time. Individual notes are referred to by index i , which identifies the location a note occupies in the sequence. Different segmentation algorithms might extract different features from this list. For instance, a list of inter-onset intervals (IOIs) measures the duration between successive pitches, a list of offset-to-onset intervals (OOIs) indicates the rests between successive note pairs, and a list of absolute pitch intervals (APIs) contains the absolute difference in semitones between adjacent pitches (each interval’s sign is ignored). A segmentation algorithm’s most basic output is *segmentation vector* \mathbf{s} , where $\mathbf{s} = (s_1, \dots, s_i, \dots, s_n) \in \{0, 1\}^n$. Only when a segment starts at position i does s_i equal 1; otherwise it is 0. An algorithm might also output a *boundary strength* vector \mathbf{w} , where $\mathbf{w} = (w_1, \dots, w_i, \dots, w_n) \in [0, 1]^n$. w_i quantifies how “strong” an algorithm perceives the boundary at position i to be, and the vector is normalized over its entire length. Finally, one could imagine that an algorithm might output a *fitness value* that rates its solution’s plausibility.

3.1 Algorithms

Grouper. This algorithm is part of the Melisma Music Analyzer developed by Temperley (2001) and Sleator.¹ Grouper was designed to extract melodic phrase structure from a monophonic piece of music. Its input is both a note list and a *beat list*, which specifies the metrical structure of the melody. Internally, only duration (IOIs and OOIs) and metric information (beat list) are considered; pitch is ignored. For a given input, output \mathbf{s} is deterministic. Dynamic programming (Bellman 1957) is used to carry out an efficient search for an “optimal” segmentation over the set of 2^n possible solutions. Optimality is defined in terms of preference rules adapted from Lerdahl and Jackendoff (1983). Roughly speaking, these rules prefer a solution whose boundaries are both coincident with the underlying metrical structure and with events whose IOIs and OOIs are larger. Preference is also given to solutions whose individual segment lengths approximate a prespecified number of notes. These rules are controlled by high-level parameters that specify the penalty to be assigned to each segment within \mathbf{s} . The optimal solution is the one whose cumulative penalty is smallest.

Local Boundary Detection Model (LBDM). This model, developed by Cambouropoulos (2001), is motivated by the gestalt principles of similarity and

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

proximity. As such, it quantifies the degree of discontinuity locally along each adjacent pair of notes. While LBDM is not a complete model of grouping in itself,² it is still compelling to consider this algorithm in isolation, especially since it might be used to segment raw MIDI data in real time. Another advantage is the model’s parsimony, which contributes to making it straightforward to implement.³ LBDM deterministically transforms a note list into a segmentation vector \mathbf{s} and boundary strength vector \mathbf{w} . Internally, discontinuity is quantified using IOI, OOI, and API lists. At each potential boundary location, a strength is calculated that is proportional to the rate at which the neighboring intervals change and the magnitude of the current interval. For each list, a separate strength vector is calculated. \mathbf{w} is the weighted sum of the three individual strength profiles, where high-level parameters control the contribution of each profile. Segmentation vector \mathbf{s} is obtained by analyzing the shape of \mathbf{w} and inserting a boundary at every “significant” local maximum. To prevent the algorithm from subdividing a melody too much, maxima are only considered at those locations where \mathbf{w} exceeds a high-level threshold parameter (cf. Fig. 1).

Data-Oriented Parsing (DOP). Bod (2001) argued for a memory-based approach to segmentation and implemented such a system using a probabilistic grammar technique. His most sophisticated and successful grammar performs data oriented parsing (DOP), which learns probabilistic trees from a large corpus of presegmented musical 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. Internally, parsing only considers pitch (in relation to the tonic) and note length. After training, the trees can parse an input note list, producing an output \mathbf{s} . To implement DOP would involve significantly more work than LBDM, and since we did not have access to a completely ready-to-use version of this software, we do not present examples of DOP’s output. We introduce this model, rather, because of its probabilistic nature, a property that provides an attractive mechanism for handling ambiguity.

3.2 Algorithm Parameter Estimation

In the examples described in Sect. 4, we present segmentations by LBDM and Grouper along with the musicians’ data. Each segmentation was constructed using a high-level set of parameters that performed well on a large subset of the Essen collection. For details, see Thom et al. (2002).

3.3 Evaluation

Computational melodic segmentation is still a relatively new field of research and many unexplored issues remain. This is especially true concerning an algorithm’s

² For example, musical parallelism (Cambouropoulos 1998) is not considered.

³ Our software is available at i11www.ira.uka.de/~{}musik/segmentation.

evaluation. Cambouropoulos (2001) and Temperley (2001) have both validated their models by demonstrating how well they behave on hand-selected musical examples.⁴ To demonstrate an algorithm’s “musical plausibility,” an analysis using specific melodic examples is *crucial*. However, a more objective, systematic measure is also needed because it allows important computational questions to be investigated. For example, such a measure was used to explore how sensitive LBDM and Grouper are with respect to high-level parameter settings and different musical styles (Thom et al. 2002). Even more compelling, when machine learning is used, as with DOP, a systematic measure becomes indispensable.

One difficulty in assessing an algorithm’s performance is that the classification task – to segment or not to segment at each note – is highly skewed. For instance, in the 52-note melody in Fig. 1, between 3 and 12 boundaries seem plausible. Thus, an algorithm that predicted no boundaries would appear to have a very low error rate. In the literature, a melodic segmentation algorithm’s performance has been systematically measured using *F score* (Bod 2001, Thom et al. 2002):

$$F(\mathbf{s}, \mathbf{s}^*) = \frac{1}{1 + \frac{FN+FP}{2TP}} \in [0, 1], \quad (1)$$

where \mathbf{s} and \mathbf{s}^* are two segmentations of the same note list. Whereas TP, the number of true positives, records how many boundaries are identical in both segmentations, FP and FN, the number of false positives and false negatives, record how many boundaries are inserted in only one of the two solutions. F score is an appropriate evaluation measure because it excludes true negatives, and keeps them from misleadingly dominating the assessment. On the other hand, it makes the questionable assumption that errors at different boundary locations are independent.

4 Results

In this section, we present results from the data collection described in Sect. 2 and the segmentation algorithms referred to in Sect. 3.2. When we began analyzing the data, three subjects’ segmentations were eliminated because they were incomplete. Among the remaining segmentations, a few seemed to be inconsistent or “strange.” However, we decided not to interfere with this remaining data by eliminating potential outliers. For example, when one experienced musician obtained a substantially different segmentation for a folk song – either because a strong upbeat was disregarded or went unnoticed – we did not remove it because the task is so subjective and ambiguous. Additionally, robust outlier detection generally requires more than twenty data points, a practical issue, for the data collection and entry is quite time-consuming.

Despite these problems, the data clearly corroborates the hypothesis that ambiguity is an inherent property of the segmentation task and should not be ignored. Even for the “simple” folk song displayed in Fig. 1, nineteen musicians

⁴ Temperley has also validated his model using songs from the Essen collection.

produced nine different segmentations on the sub-phrase level. In a more complex excerpt (Fig. 3), the number of segmentations rose to eighteen (only two of the nineteen musicians gave the same answer). In none of the ten melodic excerpts was unanimous agreement obtained. One cause for this ambiguity concerns granularity – one musician’s notion of a phrase might more closely coincide with another’s notion of a sub-phrase – yet in terms of identifying “locally salient chunks,” both are musically reasonable. In other cases, musicians might agree on a phrase’s length, but disagree on location, and in this situation, ambiguous boundaries are often adjacent to one another. The examples that follow clarify these points.

4.1 Example 1: Folk Song

The first example (Fig. 1) is a 19th century folk song taken from the Essen collection (E0356). The phrase structure contained in the Essen notation divides the song into 4+4+2+2+2+3 bars. The first two Essen phrases comprise a plain repetition, and are confirmed by practically all of the musicians at the phrase level. On the sub-phrase level, there is structural ambiguity, and a choice between “parallel alternatives” presents itself, where one can either emphasize the downbeat (more common) or the upbeat eighth-note G (less common). A notable feature of this ambiguity is its distribution on adjacent locations. There is also considerable disagreement on the phrase structure of the last five bars. The spectrum ranges from no phrase boundary, to boundaries at bar 13 or bar 15, to boundaries at both locations. Different notions of granularity are prevailing, resulting in a variable number of phrases per song (cf. Fig. 2). Again, either choice – whether to insert a boundary at bar 13 or at bar 15 – is reasonable, the former alternative focusing on the implicit harmonic structure and the latter emphasizing the melodic build-up in bars 9–14.

These findings suggest that the explicit segmentations in the Essen collection should not be regarded as the only, or even the most musically plausible, possibilities. It should rather be assumed that, in principle, different solutions can be equally well-founded. Our assumption is supported by a quotation from Dahlig (2002) that describes the Essen segmentation process:

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

In other words, an algorithm’s segmentations should be assessed in the light of these remarks. For instance, in the second half of Example 1, Grouper and Essen agree, yet they differ in the first half. Alternatively, when compared with the musicians’ solutions, Grouper and LBDM both miss important boundaries at bars 5 and 9. The segmentation boundaries suggested by LBDM are located at large IOIs and pitch intervals, which do not provide reliable cues for musically plausible segmentations in this example. The musicians’ segmentations are

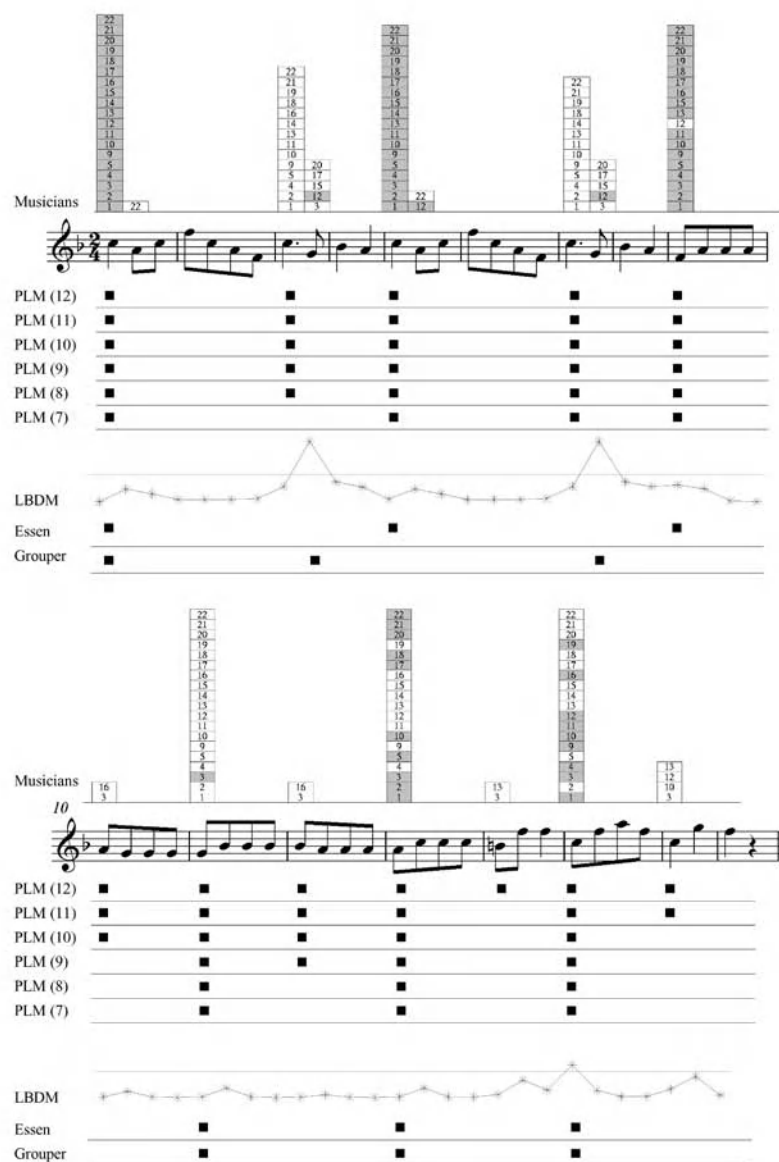


Fig. 1. Example 1: Folk song E0356, annotated with segmentation results from the musicians (upper histogram) and the algorithms (lower table). Each histogram count number identifies a particular musician. Grey squares identify the first note of both a phrase and sub-phrase; white squares correspond to sub-phrases only. The black squares refer to segmentations generated by the Position Length Model (Sect. 5.5). The last three rows show the boundary strength calculated by LBDM, the segmentation provided in the Essen collection, and Grouper's solution. The dashed line demonstrates how LBDM uses its threshold to derive s from w .

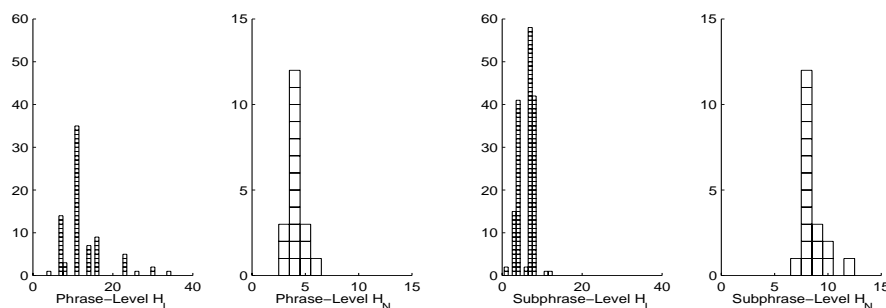


Fig. 2. Example 1: Distributions of the number of notes per segment (H_L) and the number of segments per song (H_N). For definitions, see Table 2.

clearly influenced by motivic parallelism, something neither algorithm considers. Metrical parallelism is also an influence, which only Grouper has access to.

4.2 Example 2: Fugue

The second example (Fig. 3), an excerpt from the beginning of Fugue XV of Bach’s Well-Tempered Clavier I, is melodically more complex. In contrast to the previous example, we have no indication of a theoretically “correct” segmentation, and Bach did not notate explicit phrase marks.⁵

A quick glance at the histograms in Fig. 3 conveys an impression of the disagreement between the subjects, particularly on the sub-phrase (i.e. the motivic) level, where boundary marks can often be found right next to each other. A closer look, however, reveals that adjacent phrase marks never stem from the same musicians, and some segmentations run parallel to one another for several bars (e.g. along the melodic sequence in bars 6–7). While Grouper explicitly models such a behavior (preferring boundaries at parallel metric positions), in this example it nevertheless produces incoherent results in bars 6–7. Because of the complex interplay between the weighted rules and the dynamic programming optimization over the entire melody, it is difficult to explain why the algorithm chose exactly these boundaries. The behavior of LBDM is easier to predict because it operates on a strictly local context. The huge peak in the boundary strength in bar 9 shows that a single long note can create an unreasonably large amplitude that makes threshold selection more difficult.⁶

⁵ Temperley (2001) points out: “Even structural information (phrasing slurs, bar lines, etc.) added by the composer is really only one ‘expert opinion,’ which one assumes generally corresponds to the way the piece is heard, but may not always do so.” (p. 365).

⁶ Cambouropoulos (2001) recommends using another high-level parameter to limit the boundary strength for large intervals.

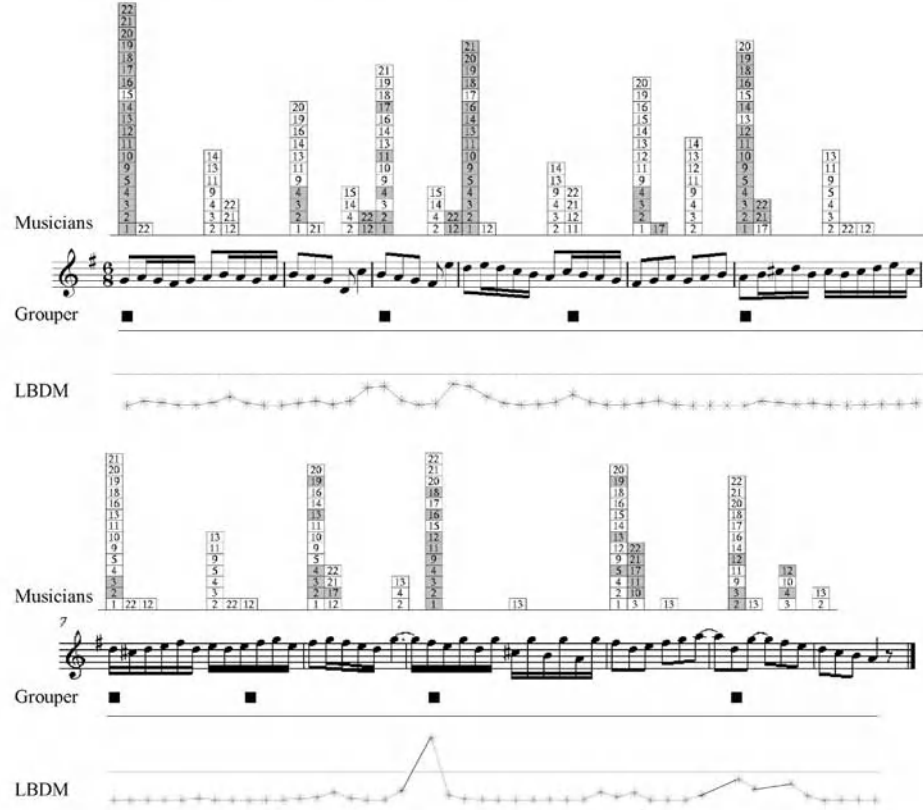


Fig. 3. Example 2: Excerpt from Fugue XV of Bach’s Well-Tempered Clavier I. For details, see Fig. 1.

5 Modeling Ambiguity

Although the musician data obtained for a note list is clearly valuable, by itself it provides no mechanism for automatically determining how some new segmentation for the same note list should be evaluated. For a systematic evaluation of this sort, one would like a fitness evaluator:

$$\text{fit}(\mathbf{s}, \mathcal{S}) \rightarrow [0, 1]$$

that maps the new \mathbf{s} into a larger value when it is “more plausible” given the musically motivated segmentations in *reference set* \mathcal{S} .

Several questions arise when attempting to evaluate a new and possibly *novel* segmentation given an ambiguous reference set. First, when should certain segmentations be perceived as more or less “similar” to one another? Second, when should an explicit parametric model be used and what parametric form is appropriate? Alternatively, should the mapping be constructed in a completely non-

parametric, data-driven way, for instance comparing segment s to each $s^* \in S$ and then averaging these values? When allowing for ambiguity, a note list can map into many segmentations, and the musical quality of these solutions might vary considerably. A probabilistic, parameterized model becomes attractive provided it assigns larger probabilities to those segmentations that are musically more plausible.

Data collection and entry are expensive, so this important issue must be considered. The more *complex* a model, i.e. the more parameters it contains, the more likely it is to *overfit*, which is to say that it predicts the behavior of S very closely yet fails to generalize well on unseen examples. Overfitting typically occurs when a model is fit using too little data (Bishop 1995), and often non-parametric methods are very sensitive to this because their complexity is not restricted by a parametric form. We are concerned about overfitting because the data in our reference set seems to be *incomplete*, meaning that it does not adequately represent all reasonable segmentations. A strong argument for this incompleteness comes from the histograms in Figs. 1-3. In each case, if a single segmentation had been withheld, bins could become empty. Consequently, unless we are careful when building a probabilistic model from such data, we might easily derive a model that could map a musically reasonable segmentation onto a fitness value of zero. We handle this issue by *smoothing*, adding an additional, small value ϵ to every bin and then renormalizing.

5.1 An Evaluation Example

In Table 1, we present a simple, musically motivated example for a 12-note segmentation task. In the 2nd column of rows (i)-(iii), a reference set of three segmentations is depicted, two of which are identical. This set would arise when two of three musicians disagreed about whether or not a repeated melodic line began on an upbeat or an adjacent downbeat. The top stave in Fig. 1 presents one example of such *positional parallelism* for adjacent locations. In rows (a)-(d) of Table 1, four novel segmentations are given. These were chosen because they were musically less reasonable given the reference set S . Example (d) provides a baseline worst case, demonstrating how a model assesses a ridiculous solution. Musically, the positional parallelism that characterizes the reference solutions justifies the inferiority of examples (a)-(c). Assuming that segments of length one are exceptional, examples (b) and (c) are also musically less plausible than (a). The remaining columns display the values assigned to each example under a particular fitness evaluation model. These models will now be described and considered in light of these examples. Since the fitness values for the different models grow at different rates between zero and one, they will only be directly compared per-column. Only the differences between these corresponding rankings are relevant across columns.

Table 1. An ambiguous evaluation example

	Segmentations	fit(\mathbf{s} PM)	fit(\mathbf{s} AFM)	fit(\mathbf{s} PLM)
$\mathcal{S} \left\{ \begin{array}{l} \text{(i)} \\ \text{(ii)} \\ \text{(iii)} \end{array} \right.$	100010001000	0.24	0.77	0.13
	100010001000	0.24	0.77	0.13
	100100010000	0.16	0.55	0.031
(a)	100100001000	0.19	0.66	0.037
(b)	100110000000	0.19	0.66	0.0099
(c)	100110011000	0.17	0.75	0.00071
(d)	111111111111	0.02	0.4	0.000075

5.2 Notation

For convenience, two transformations of segmentation vector \mathbf{s} are introduced. Each element in *segmentation index vector* $\mathbf{s}' = (s'_1, \dots, s'_k)$ identifies the first position of a segment, that is $s'_j = i$ if and only if $s_i = 1$. Variable k refers to the total number of segments in solution \mathbf{s} . The *segmentation length vector* $\mathbf{l} = (l_1, \dots, l_k)$ is calculated as the number of notes in each segment. For $j \in \{1, \dots, k-1\}$, we set $l_j = s'_{j+1} - s'_j$; for $j = k$, we set $l_k = n + 1 - s'_k$. Three histograms, which summarize various average features in the reference set, are defined in Table 2. The sums range over $\mathbf{s} \in \mathcal{S}$; k and \mathbf{l} depend on the current segment \mathbf{s} being summed; $\mathbb{1}$ is the indicator function. $H_S(i)$ refers to bin i in histogram H_S and indicates how often any musician began a segment at location i . $H_S(1)$ is not included because, according to our task definition, the first segment always starts at position one, which does not provide any relevant information. Thus, to construct H_S for Example 1, use the counts in bins 2 through 52 in Fig. 1. The notation of H_L and H_N are similar. $H_L(i)$ indicates how often any musician chose a segment of length i , and $H_N(i)$ denotes how many musicians used i segments in their segmentation.

In our probabilistic models, histograms are always smoothed. For instance, the position histogram in Table 1 yields

$$H_S = (0.318, 0.005, 0.005, 0.109, 0.214, 0.005, 0.005, 0.109, 0.214, 0.005, 0.005, 0.005).$$

when smoothed with $\epsilon = 0.05$.

Table 2. Histogram definitions

Histogram	Definition	Range	Normalization
H_S	$H_S(i) = \frac{1}{C_S} \sum_{\mathbf{s}} s_i$	$i \in \{2, \dots, n\}$	$C_S = \sum_{\mathbf{s}} (k-1)$
H_L	$H_L(i) = \frac{1}{C_L} \sum_{\mathbf{s}} \sum_{j=1}^k \mathbb{1}(l_j = i)$	$i \in \{1, \dots, n\}$	$C_L = \sum_{\mathbf{s}} k$
H_N	$H_N(i) = \frac{1}{C_N} \sum_{\mathbf{s}} \mathbb{1}(k = i)$	$i \in \{1, \dots, n\}$	$C_N = \mathcal{S} $

5.3 The Position Model

A very simple but naive approach for combining the reference set into a parametric model assumes that histogram H_S alone is sufficient in defining how likely a particular segmentation is. We refer to this approach as the *Position Model* (PM) because no higher-level structural information, such as segment length, is considered. A segment's probability or likelihood determines its fitness:

$$\text{fit}(\mathbf{s}|\text{PM}) = \Pr(\mathbf{s}|H_S)^{\frac{1}{k}} \propto \left(\prod_{i=2}^k H_S(s'_i) \right)^{\frac{1}{k}}. \quad (2)$$

This model assumes that \mathbf{s} was generated by a Multinomial distribution that prefers bin i with probability $H_S(i)$. The proportional symbol \propto arises because the probability distribution has not been normalized. Provided only solutions with a fixed number of segments k are being compared, normalization can be ignored (each scales by the same divisor). When solutions are composed of different k s, however, the power of $\frac{1}{k}$, which computes the *geometric mean*, is needed to remove the dependence of a solution's likelihood on k .⁷

The third column in Table 1 illustrates the behavior of PM and motivates why considering only the positions of \mathcal{S} provides a musically inadequate assessment. First, note that example (d) would have had a fitness of zero if no smoothing had been used ($\epsilon = 0.05$). Also note that the rankings of examples (a)-(c) are problematic, for they all rank higher than segment (iii). Examples (a) and (b) are undifferentiated even though the latter contains an implausible segment of length one. While this behavior sheds light on how PM generalizes – an ability that is especially important given our belief that \mathcal{S} is incomplete – the way it generalizes is musically inadequate. One problem with 2 is that it only considers where segments begin ($s_i = 1$), which is reminiscent of only considering the number of true positives. As a consequence, both the length of the individual segments and the number of segments within a solution are ignored. Regardless of these deficiencies we introduced this model because it is all-too-easy to mistakenly assume that H_S adequately quantifies a melody's ambiguity. For example, at first we assumed that, in terms of LBDM's performance, ambiguity could be explicitly modeled by considering H_S and \mathbf{w} together. The problems we encountered are similar to those described above.

5.4 Average F Score Model

The *Average F score Model* (AFM) improves on PM by considering additionally the false negatives and positives in its calculation:

$$\text{fit}(\mathbf{s}|\text{AFM}) = \text{AFM}(\mathbf{s}) = \frac{1}{|\mathcal{S}|} \sum_{\mathbf{s}^* \in \mathcal{S}} \text{F}(\mathbf{s}, \mathbf{s}^*). \quad (3)$$

⁷ The geometric mean rescales the probability mass contained in a k -sided volume into the length of one side of the corresponding k -sided cube, providing a probabilistic measure that is decoupled from length. Without this term, solutions with larger k are usually significantly smaller than solutions with smaller k .

Because AFM is obtained directly by averaging over all pairs of F scores between \mathbf{s} and each reference segmentation, it is non-parametric.

At first glance, one might expect this model to grasp positional parallelism because the locations of each reference solution’s zeros and ones are considered together. Upon deeper consideration, however, it can be shown that this parallelism is not adequately handled. For example, even though example (iii) results from (i) by sliding its boundaries one position to the left, one can prove that their musically unrelated combination, i.e. example (c), has a higher average F score than reference segmentation (iii) (Höthker et al. 2002). The primary problem with AFM is that an error occurring on one boundary is treated independently from all other boundary errors, i.e. a segment’s length is never explicitly considered. Another disadvantage is that AFM does not model a segment’s likelihood and thus cannot be used to generate new “typical” solutions stochastically. This inability is a real handicap because simulation provides valuable insight into a model’s behavior.

5.5 Position Length Model

In the *Position Length Model* (PLM), we combine the musicians’ preferences for certain local position boundaries, a solution’s segment lengths, and the total number of segments. The benefit of this approach is that it incorporates higher-order information, which allows us to model ambiguity simultaneously at different levels of abstraction. Likelihood is estimated as:

$$\text{fit}(\mathbf{s}|\text{PLM}) = \text{Pr}(\mathbf{s}, \mathbf{l})^{\frac{1}{k}} \quad (4)$$

$$\approx \text{Pr}(\mathbf{s}, \mathbf{l} | H_S, H_L, H_N)^{\frac{1}{k}} \quad (5)$$

$$\propto H_N(k) \left(H_L(l_k) \prod_{j=2}^k H_S(s'_j) H_L(l_{j-1}) \right)^{\frac{1}{k}} \quad (6)$$

The first equality indicates that the model is based on the joint distribution of both a solution’s position and its length. Again, the geometric mean decouples the dependence between fitness and the number of segments in \mathbf{s} . The first approximation assumes that this joint can be described adequately by the empirical distributions estimated for the reference set. The second approximation makes two important assumptions: that the likelihood of each pair (s_j, l_{j-1}) is independent of all others (hence the product over j); and that the likelihood of each pair is independent in length and position (hence each pair of $H_S(s'_j)$ and $H_L(l_{j-1})$ terms).

The behavior of PLM ($\epsilon = 0.05$) is shown in the fifth column of Table 1. As in the previous models, the musically most plausible segmentation (a) has the highest likelihood among those not contained in the reference set. This segmentation’s fitness is also higher than the likelihood of reference segmentation (iii), which demonstrates that positional parallelism is not fully captured by PLM. In contrast to the previous models, however, segmentations with unlikely lengths

(i.e. (b) and (c)) have significantly smaller fitness values than the segmentations with more plausible lengths.

In Fig. 1, PLM solutions with different numbers of segments ($k = 7$ through 12) are shown. These examples demonstrate how PLM can accommodate interpretations of different granularities. These solutions were calculated by determining which segmentation was most likely for a given k . Their boundaries line up with those chosen by the musicians. Interestingly, as k ranges from 7 to 12, the segments are arranged hierarchically, i.e. the solution for a larger k contains the solution for a smaller k within it. This behavior is fairly typical. Due to lack of space, another important aspect of this model is not displayed in Fig. 1. For a fixed k , PLM can also be sampled (rather than just reporting the most likely solution). When this is done, multiple solutions are obtained and therein the model’s preference for position-parallel solutions is evidenced.

6 Conclusions and Future Directions

The data we collected from the musicians suggests that ambiguity is an inherent part of the segmentation task and should not be ignored when evaluating an algorithm’s performance. We argue that the Position Length Model provides a more compelling fitness evaluation for handling musical ambiguity than some simpler models that do not consider the positions, lengths, and number of segments together. PLM maps an algorithm’s solution into a fitness value that defines how probabilistically similar it is to the musicians’ solutions. Such insight can provide guidance about what aspects of a segmentation algorithm’s behavior are most problematic.

Ambiguity also motivates a natural extension to the LBDM and Grouper algorithms: these algorithms should report fitness values for their segmentations so that these could be compared to those reported by the musician model. Although LBDM does output boundary strength vector \mathbf{w} , the issues that arose in the Position Model demonstrate why this information alone is insufficient for constructing an ideal fitness value. Grouper should also report how fit its solutions are, and since, during optimization, this algorithm explicitly considers segment length and position, perhaps the penalties it assigns to individual segments could be used to construct a fitness. In addition, these values might be used to construct a \mathbf{w} -like vector, which would provide insight into Grouper’s behavior in complex situations (cf. the Bach fugue). More generally, it is worth recasting these algorithms within a probabilistic framework because then one can generate “typical” solutions stochastically, providing a valuable tool for exploring the model’s underlying behavior systematically.

This realization brings us back to DOP, whose inherent probabilistic basis should result in a straightforward sampling procedure and likelihood calculation. Currently, however, DOP has only been trained on segmentations from the Essen collection, which presents a single (as opposed to multiple, ambiguous) solution for each folk song. Thus, DOP’s performance should be explored in more explicitly ambiguous settings. DOP also suffers from a major practical problem:

it has great need for hand-segmented data. Perhaps the most promising future direction for algorithmic segmentation is to combine Grouper's and LBDM's musically relevant features into a probabilistic, machine learning based method in order to reduce the amount of data needed to configure the model.

Acknowledgments. We are grateful to the musicians who took the time to segment our corpus, and to Rens Bod, Emiliós Cambouropoulos, Ewa Dahlig, Ralf Schoknecht, and three anonymous reviewers for their valuable input. This research was supported by the Klaus Tschira Foundation and the German-American Fulbright Commission. For a more complete acknowledgment and a listing of the musician data, see

<http://i11www.ira.uka.de/~{}musik/segmentation>.

References

- R. Bellman (1957). *Dynamic Programming*. Princeton University Press.
- C. M. Bishop (1995). *Neural Networks for Pattern Recognition*. Clarendon Press.
- R. Bod (2001). Memory-based models of melodic analysis: challenging the gestalt principles. *Journal of New Music Research*, 30(3):In Press.
- E. Cambouropoulos (1998). Musical Parallelism and Melodic Segmentation. In *Proceedings of the XII Colloquium on Musical Informatics*, Gorizia, Italy, pages 111-114.
- E. Cambouropoulos (2001). The *Local Boundary Detection Model (LBDM)* and its application in the study of expressive timing. In: *Proceedings of the International Computer Music Conference (ICMC)*, Havana, Cuba, pages 290-293.
- T. Crawford, C. S. Iliopoulos, and R. Raman (1998). String-matching techniques for musical similarity and melodic recognition. In: W. B. Hewlett and E. Selfridge-Field, editors, *Melodic Similarity: Concepts, Procedures, and Applications*, pages 73-100. MIT Press.
- E. Dahlig (2002). Personal communication.
- K. Höthker, B. Thom, and C. Spevak (2002). Melodic segmentation: evaluating the performance of algorithms and musicians. Technical Report 2002-3, Faculty of Computer Science, University of Karlsruhe.
- F. Lerdahl and R. Jackendoff (1983). *A Generative Theory of Tonal Music*. MIT Press.
- H. Schaffrath (1995). *The Essen Folksong Collection*. D. Huron, editor. Center for Computer Assisted Research in the Humanities, Stanford, California.
- D. Temperley (2001). *The Cognition of Basic Musical Structures*. MIT Press.
- B. Thom, C. Spevak, and K. Höthker (2002). Melodic segmentation: evaluating the performance of algorithms and musical experts. Submitted to the *International Computer Music Conference*.

Combining Grammar-Based and Memory-Based Models of Perception of Time Signature and Phase

Neta Spiro

Cognitive Science Center Amsterdam,
University of Amsterdam, Nieuwe Achtergracht 166,
1018 WV, Amsterdam, The Netherlands.
`nspiro@science.uva.nl`*

Abstract. The present study investigates the modeling of the perception of time signature and phase using grammar-based and memory-based approaches. It explores how far note-length can be used as the sole input to a model of perception of heard music. Two models are developed: one uses a rule-based grammar and the other uses a combination of a rule-based grammar and a memory-based approach. The relative success of the two models is compared and evaluated. This study explores one dialect, solo string music by Bach: The unaccompanied Suites for Violoncello, BWV 1007-1012 and unaccompanied Partitas and Sonatas for Violin BWV 1001-1006. It is shown that the results of two approaches improve over previous models such as those of Longuet-Higgins and Steedman (1971) and Temperley and Sleator (1999) and that the combination of the two approaches is most successful.

1 Introduction

The aims of this study are to explore, develop and combine different approaches to modeling the processes of perception of music and to develop two models of metrical perception using relative note-length as the only source of information.

This study focuses on time related elements which are fundamental to all types of music: time signature and phase.¹ The bar normally has a certain repeating number of beats and is sometimes identified by some kind of ‘accent’ at its start. This accent, though not felt at the beginning of every bar, can involve the relative length of a note with respect to its surrounding ones. The listener superimposes some sort of ‘accent’ on notes once he identifies the time signature. This accent is an important part of the process of music perception and will be incorporated in this study.

* This work was carried out at the Division of Informatics, University of Edinburgh, 2 Buccleuch Place, Edinburgh EH8 9LW

¹ These are reflected in notated music by the time signature and the position of the bar lines. It is possible to describe phase by identifying the length (in beats) of the anacrusis.


Given a representation of:



which will be represented as

1 1 1 1 4 2 2 4 2 2 4 4 4

the model should be able to return:



which will be represented as

[1 1 1 1][4 2 2 4 2 2][4 4 4

Fig. 1. Bourrée No. 1, Suite No. 4 for ‘cello, bars 1-3³

The first of the two models, a purely grammar-based one (the Internal Structure Program), is used as a comparison for the second which combines the grammar-based approach with a memory-based one (the Combined Program) (The approaches are discussed in Section 2 and the Models and Programs are given in Section 4). The results of these programs are compared with those of earlier grammar-based programs developed by Longuet-Higgins and Steedman [1] and Temperley and Sleator [2] and show that the Combined Program compares favorably with the earlier techniques (Sections 5 and 6).

2 Rule-Based Grammars and Memory-Based Approaches

Rule-based grammatical approaches to modeling music perception aim to represent possible processes that follow hierarchical rules that either succeed or fail. The resulting models are designed to capture essential aspects of the structures of the music being perceived.

Longuet-Higgins and Steedman [1] created two formally precise models of the cognitive process involved in the comprehension of classical melodies (Bach Fugue subjects), one using relative note-length and one note-pitch. In the present study only the former is considered and is partly described in Section 4.2.² Their model presents a clear approach, with a number of basic principles, for the use of relative note-length in the perception of music. Some more recent and complex approaches analyse MIDI rather than symbolic representations of the music, such as in [2]. This study concentrates on approaches and techniques of modeling music perception so it is appropriate to begin with a clear rule-based approach and less complex, symbolic representations.

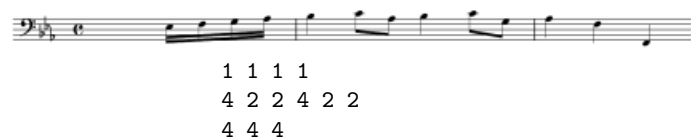
An example of the more developed grammar-based approach is Temperley and Sleator’s Melisma Music Analyzer program [2] which also uses elements of Longuet-Higgins’ and Steedman’s work. The Preference Rules that form the basis of their model, provide a high-level way of describing a cognitive process [2]. However, they are not all helpful in this study. For example, in the identification of meter, the Event Rule looks for the greatest possible number of notes that start on a beat. In solo string music, this is not always useful. Temperley and

² The algorithm is given in [1].

Sleator's program, like Longuet-Higgins and Steedman's, uses an incremental system. It also uses parallel processing and a search procedure to model the ambiguity that arises, and the choice that takes place during the perception of music. The current study adopts a simpler version of this approach. Temperley and Sleator's program is used as part of the evaluation of the programs developed in this study.³

Rule-based grammars are developed in this study on the basis of the program by Longuet-Higgins and Steedman and are used to determine time-signature and phase of new input.

Memory-based models, rather than relying on generalized rules, take evidence from previously seen information for the formation of a database and, for music, have mainly been used in the identification of phrases ([7], [8]). In some memory-based approaches, the patterns used for identification can be variants of those in the original music. In this study a pattern is a finite sequence of relative note-lengths representative of a bar (or part of a bar in the case of an anacrusis). In this study, the relative note-lengths are calculated with reference to the scores of the pieces, with 1 semiquaver being 1 unit. The three relative note-length patterns of the following phrase are represented as three patterns:



Such numerical patterns, identical to those found in the development data, were stored in the corpus and only exact matches were accepted when testing the novel input. Other studies often use probabilities of finding a sequence of events ([7], [9]) but there are problems of determining the significance of patterns ([10], [11]). This study shows that it is not always essential to use probabilistic approaches. Instead, each pattern in the development data is equally weighted.

The memory-based pattern matcher (henceforth MBPM) learns patterns of relative note length within the context of their time signature and uses this to infer the time signature and phase of new input.

3 The Corpus of Music

The unaccompanied Suites for Violoncello, BWV 1007-1012 [12] and unaccompanied Partitas and Sonatas for Violin BWV 1001-1006 [13] are examples of an advanced form of Solo music by Bach which are complex enough for a valid test of the models. Relative note length is used because when listening to music we abstract from the physical stimulus and we remember simple melodies in terms of patterns and relationships i.e. in terms of the pitch and duration ratios [14].

³ The program in [1] was implemented for this study and the program available in [3] was used. Therefore, this study concentrates on these two programs. Other models include [4], [5] and [6] whose programs were not implemented in this study. Therefore, they are not used in the evaluation of the programs developed here.

Among the movements of the pieces there are some that contain chords which, in the corpus, are represented by one note-length resulting in the creation of a monophonic corpus. Some movements are excluded from the corpus because they were too polyphonic or they contained only notes that are all of equal length (such as the Courante from Partita No. 1). Individual movements were entered into a database manually. There are 32 movements in total, of which 16 were used for the development of the program and the rest were used for testing.

4 The Models and the Programs

The first of the two programs developed in this study is grammar-based and gives as output the time signature of the movement, the internal structure of the bar and the length of the anacrusis (called the Internal Structure Program below).

The second program (called the Combined Program below) is based on the grammar-based and memory-based approaches discussed above (see parts 1 and 3 below). The rule-based grammar used in the Internal Structure Program may be seen as an extension of the one used in the Combined Program. The extensions are described immediately after the discussion of the rule-based grammar of the Combined Program below.

Both the rule-based grammar and the MBPM are able to give information on time signature and phase (see also [1]). However, the aim of this study is to explore the two different approaches to modeling music perception. At the moment there is no feedback between the pattern matcher and the rule-based grammar. The information about the time signature is concluded by the rule-based grammar and is then used as input to the pattern matcher which identifies phase. The Combined Program is made up of four consecutive parts.⁴

1. The simpler rule-based grammar
2. The segmenter
3. The pattern matcher
4. The beat segmenter

4.1 The Simpler Rule-Based Grammar

The rule-based grammar identifies time signature using relative note length as the only input. The algorithm for this part of the program is partly based on the one by Longuet-Higgins and Steedman [1]. Their program identifies metrical unit that can be smaller or larger than the time signature. Their theory assumes that ‘accent’, identified when one note is longer than at least one of the notes that follow it, is the main tool for the identification of metrical units during music perception. This is formalized through the notion of a Dactyl which is defined as the first three notes in a sequence of four, such that the second and third are equal in length and shorter than the first or fourth. The first three of these notes create a metrical accent at some level (Rule 2a below).

⁴ A full description of the model and the results is given in [15].

The grammar-based model in the present study consists of the following rules, process and decision making procedures. The definitions in Rule 2 are from the original program by Longuet-Higgins and Steedman. The rest are developments that were made in the current study.

The following abbreviations are used below:

TS = Current hypothesis for time signature,

N1 = length of note 1 of the piece, N2 = length of note 2 of the piece etc.,

n1 = length of note 1 of section being studied, n2 = length of note 2 of section being studied etc.,

A slur indicates the identification of a metrical unit giving the new hypothesis for time signature.

Relative lengths of notes are given in number of semiquavers, 1 semiquaver = 1.

The Rules

1. Once a time signature is adopted it cannot be replaced by a shorter one or one that cuts across it. The new hypothesis identified by each rule is only accepted if its only prime factors are 2 or 3. Two hypotheses for the time signature can be suggested at the start of the input.

- IF $N1 > N2$ and $N2 \neq N3$ and $N1 + N2$ has only 2 or 3 as prime numbers
THEN $TS = N1 + N2$.


For instance in⁵  $TS = 3 + 1 = 4$.

- IF $N1$ has only 2 & 3 as prime numbers, THEN $TS = N1$
ELSE IF $N1$ is followed by equal length notes and total length of the equal length notes plus $N1$ has 2 or 3 as prime numbers,
THEN TS is the total of adding all equal length notes plus $N1$.

For instance in⁶  $TS = 5 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 = 16$.

2. There are rules for the confirmation or refutation of the time signature or the increase of the hypothesis to a total equivalent to a higher level of the tree. The list of notes is examined from the note that has been reached for differing lengths of note-lists depending on the rule. a, b and c are arranged hierarchically.

- a. IF $n1 > n2$ and $n2 = n3$ and $n4 > n2$ (i.e. if there is a Dactyl)

THEN $TS = n1 + n2 + n3$. For instance in⁷ 
if the test begins from A flat as $n1$, the new $TS = 4 + 2 + 2 = 8$.

- b. IF $n1 > n2$ and $n2 < n3$

THEN $TS = n1 - n2$ if TS has only 2 and 3 as prime numbers.

In this case⁸ 

if the test begins from E as $n1$, the new $TS = 6 - 2 = 4$.

⁵ Beginning of Sarabande, Suite no. 2.

⁶ Beginning of Prelude, Suite no. 5.

⁷ Beginning of Gavotte I, Suite no. 5.

⁸ Sarabande, bars 23-4, Suite no. 2.

- c. If n_1 is twice the length of the current hypothesis of the time signature or more, then TS is the longest that is filled by that note.

These rules are called the Dactyl rules.

3. The Dactyl rules begin to develop the identification of accent using notes that are relatively long note in a sequence. Another such rule is:
IF n_1 is double or triple the length of n_2 and $n_3 \neq n_2$,
THEN $TS = n_1 + n_2$.



For instance in⁹

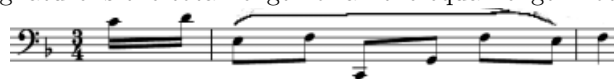
if the test starts from D as n_1 ,

the new $TS = 4 + 2 = 6$.

This rule is called the Waltz rule.

4. A common criticism of models that look only at note-length is that they cannot deal with pieces in which there are many equal-length notes. It is rare that a whole piece is made up of equal-length notes. Therefore, the following rules were added:

If there is a series of equal-length notes followed by a different length note, the new time signature is the total length of all the equal-length notes.



For instance in¹⁰

if the test starts on E as n_1 ,

the new $TS = 2 + 2 + 2 + 2 + 2 + 2 = 12$.

These rules are called the Equal rules.

The Process. In Longuet-Higgins and Steedman's model, the rules were tested in the order they were written and if a rule was successful this was taken as the result for the current set of notes and the program moved onto the next set of notes. In this study the process is different and is in three parts:

1. Two hypotheses are possible at the first test (see Rule 1 above).
2. For the first 32 notes of the movement (the decision stage) each of the rules above is tested and each new hypothesis is added to the list of possible time signatures.¹¹ Any time signature is only recorded once, minimizing the size of the tree.
3. After 32 notes (the confirmation stage) all the hypotheses are taken as the basis for further decisions. From each current hypothesis there can now be only two possible options. The Waltz Rules and the Dactyl Rules are kept separate and tested each time. The Equal rules are less reliable and are only applicable if the Dactyl rules fail.

In the confirmation stage, the note lists are also assessed in relation to the hypothesized start of the bar. This is on the basis that when we have heard a number of notes of a piece we generate a hypothetical time signature placing our own 'accent' on the first beat of each bar, and on this basis we then

⁹ Beginning of Gigue, Suite no. 2.

¹⁰ Sarabande, bars 10³-12¹, Suite no. 2.

¹¹ The reasons for the choice of 32 notes as the cut-off point are based on experience [15]. An automated system for the decision of the cut-off point would be preferable.

make further tests and decisions,. The program therefore keeps track of the part of the position in the bar and checks rules in relation to this.

The Decision. The method to identify the preferred hypothesis uses a numeric system of assigning scores to each rule. Each of the rules above was originally assigned a score of 1. The program was then run on the development data. Each time a Dactyl Rule or a Waltz Rule succeeded, its score was increased proportionally. The scoring assigned to the Equal rules was kept low to reflect the problematic nature of such rules. While the program is running, if a rule succeeds, its score is added to the total score of that hypothesis and the hypothesis with the highest score is taken as the time signature.

4.2 Internal Structure Program (Complex Rule-Based Grammar)

Longuet-Higgins and Steedman's program also identified the internal structure of the bar and the phase (seen through the length of the anacrusis). These are identified and presented by storing and printing each hypothesized time signature during the movement and constructing tree structure 'bottom-up'. This was also implemented in this study in the Internal Structure Program. A further test for the identified phase is also carried out in this study by proposing lengths for the anacrusis for each path, and checking the last bar of the piece. This is on the same assumption of accent discussed for the confirmation stage above. If we know when the beginning (and end) of each bar should be and what we hear does not coincide with our expectation we can re-asses our hypothesis of time signature and phase. Here this is implemented in one of a number of possible ways that follow the position in the bar and takes place in the following stages:

1. Propose an anacrusis length.¹²
2. When the end of the input is reached, take off the notes that total the length of the anacrusis.
3. Compare the resulting final bar length to the current hypothesis of the time signature.
4. If the last note of the piece ends a full bar then the score is increased, if it does not the score is decreased.

The resulting Internal Structure Program gives time signature, phase and internal structure using just the rule-based grammar.

In the Combined Program the aim of the rule-based section is to determine just the possible number of beats per bar. The following sections of the model take the results from the simpler rule-based grammar as input and were developed solely in this study.

4.3 The Segmenter

The segmenter takes the output of the rule-based program and divides the string of notes into bars in preparation for the pattern matcher. Several up-beat lengths are hypothesized so different structures result from the same time signature.

¹² Currently, the possibilities are limited to the five lengths of anacruses that occur in the corpus, this can be increased in future.

For example, in Figure 2 if, given the input shown, a time signature of 4/4 and a crochet up-beat, the output would be as in the first column. If the program is given a time signature of 4/4 and a quaver anacrusis, the output is as given in the third column.¹³



INPUT: 1 1 1 1 4 2 2 4 2 2 4 4 4 1 1 1 1 4 2 2 4 2 2 4 4 4 1 1 1 1			
OUTPUT (crotchet anacrusis)	Score per pattern	Alternative OUTPUT (quaver anacrusis)	Score per pattern
1 1 1 1	0	1 1	0
4 2 2 4 2 2	1	1 1 4 2 2 4 2	0
4 4 4 1 1 1 1	2	2 4 4 4 1 1	0
4 2 2 4 2 2	3	1 1 4 2 2 4 2	0
4 4 4 1 1 1 4	1	2 4 4 4 1 1	0
		1 1	0
Score	4	Score	0
Total Score for whole piece	24	Total Score for whole Piece	2

Fig. 2. The beginning of Bourrée no.1, Suite for 'cello no. 4. bars 1-4 out of 48 bars, taken from the test data

4.4 The Memory-Based Pattern Matcher (MBPM)

The MBPM is based on the ideas of learning through experience: certain patterns of note lengths are typical of a phase within the context of a time signature. Some patterns are common among phases but there are enough that occur most often in one time signature and phase.

This approach begins with a process of learning through exposure to a corpus resulting in the construction of a rule-base or a lexicon. The choice of development corpus therefore influences the results and so the choice of movements for development and testing was random within the corpus. Despite the small size of the corpus there were 79 different patterns identified from the movements in 3/4 and 89 from the movements in 4/4. The different patterns for each time signature were stored in separate lexicons. Reflecting the equal importance of all patterns, each pattern in the lexicon is assigned a rating of 1 (see section 2).

The time signature found by the rule based grammar and the lists of patterns given by the segmenter are used as input to the pattern matcher. The lexicon for the appropriate time signature is accessed and the segments are compared to the lexicon. An exact match occurs if all the relative lengths of the notes in a proposed bar are identical. For example, the two whole bars in each of the examples in Figure 3 have the same pattern of relative note lengths despite having different pitches and is therefore considered an exact match.

¹³ 1 represents 1 semi-quaver, so 2 is a quaver and 4 is a crotchet.



Fig. 3. Comparison and exact matching between different pieces.

If an exact match is made between the pattern in the input data and the patterns in the lexicon, the score for that hypothesis is increased by 1. The hypothesis of phase with the highest score at the end is the winning one. In Figure 2, which features the same pattern of relative note-lengths as that shown in Figure 3, the rule-based grammar selected the 4/4 time signature. Then the lexicon for the 4/4 time signature was used and the crotchet anacrusis and quaver anacrusis were assigned in two separated passes. If the pattern was matched with one in the lexicon, a score of one was added to the total score (columns 2 and 4). The winning pass was then presented as list of the note-lengths with bar lines entered at the appropriate places. If the time signature were not given by the rule-based grammar, both of the lexicons would be accessed in turn. The winning pass would then propose a time signature as well as a phase.

A similar technique of segmentation and pattern matching was used for the subsequent explicit identification of the rhythmic structure within the bar (The beat segmenter). The beginning of each beat is marked and it is possible to distinguish between simple and compound time signatures.¹⁴

5 Results and Discussion

The two programs were run on the test data and the results were compared to those obtained from running the programs developed in [1] and [2] on the same data. The corpus belongs to a single, well-defined dialect that lends itself to the formation of a lexicon. A larger corpus size is necessary for full statistical testing.

The time signature and length of the anacrusis written in the original musical score were identified correctly by the Internal Structure Program 11 out of 16 times for the development data and, as shown in the table below, 10 out of 16 times for the test data.

For the Combined Program, the time signature was identified by the rule-based grammar correctly for 13 out of 16 movements for the development data and increased to 15 out of 16 movements for the test data. For the MBPM each piece was tested for up to 5 different up-beat lengths for both of the time signatures including all the scenarios in the corpus. Once given the time signature, the MBPM successfully identifies the length of the anacrusis in 12 out of the 15 cases where the time signature is correctly identified by the rule-based grammar for the test data (and 13 out of 13 for the development data). When combined, the identifications of time signature (by the rule-based grammar) and phase (by

¹⁴ Results will be presented in the future.

the pattern matcher) are correct for 12 out of 16 cases for the test data (and 13 out of 16 for the development data).

These results are compared with those of the programs by Longuet-Higgins and Steedman and Temperley and Sleator in the table below. In the program by Longuet-Higgins and Steedman, the level of the complete bar is reached in 4 cases. In 3 further cases lower levels of the tree are reached - arriving at partially correct results. In the table, these are combined to give 7 correct results. For the Internal Structure Program only those with completely correct results (i.e. correct total time signature, subdivision and up-beat length) are counted as correct in the table. The program by Temperley and Sleator can give much more information than time signature and the length of the anacrusis but in this study only the results for these parameters are displayed. The program by Temperley and Sleator was designed for MIDI input and so required actual note-length. Therefore the input values were multiplied by 250 for slower movements and 100 for faster ones.¹⁵

Table 1. Table showing the results of the Combined Program and the Internal Structure Program compared with the original program by Longuet-Higgins and Steedman and that by Temperley and Sleator, on the current test data.

Program	No. of movements with correct Time Sig. identified	No. of movements with correct Time Sig. & Anacrusis Length identified
Longuet-Higgins & Steedman	7	6
Temperley & Sleator	9	7
Internal Structure Program (Grammar)	10	10
Combined Program (Grammar followed by Pattern Matcher)	15	12
Total Test Movements	16	16

Bearing in mind the small size of the corpus, the Internal Structure Program shows substantial improvement over the program by Longuet-Higgins and Steedman. This improvement results from the increase in the number of identifying rules, the use of the position in the bar for local analysis and the change in procedure which now employs parallel parsing and scoring mechanisms. The table also shows the improvement over the program by Temperley and Sleator, especially in the correct identification of both time signature and the length of the anacrusis.

Three pieces in compound time were added to the corpus to test the success of the Internal Structure Program on such input if only on limited data. In two cases the correct time signature and subdivision of the bar were identified and in one the correct length of the anacrusis was identified. In comparison,

¹⁵ These figures were reached by a study of the corpus used in [3].

in the original program by Longuet-Higgins and Steedman, none of results were completely correct. The program by Temperley and Sleator identified the correct time signature, subdivision and the length of the anacrusis in one case.

The results for the Combined Program show that these results can be further improved with the use of the MBPM to identify the length of the anacrusis (phase). Now there is a doubling of the success rate when compared to the Longuet-Higgins and Steedman program. In comparison with the program by Temperley and Sleator there is also an improvement.

Some of the incorrect results for the test movements were the same for different programs. This hints at the possibility of ambiguity in the music and suggests that the programs are reflecting results that listeners may also give.

The comparison between the Internal Structure grammar-based model and the Combined one shows that the Combined Program is completely accurate more often than the solely grammar-based one.

6 Conclusions and Further Developments

The Internal Structure Program and the Combined Program show similar success rates indicating that the two are useful approaches in the identification of time signature and phase. The better results obtained from the Combined Program than from the Internal Structure one indicate that the combination of rule-based and memory-based approaches is a beneficial development.

Time signature and phase are vital components of music and music perception. Note-length is not the only feature that determines time signature and phase for the listener, particularly in polyphonic music. For a more complete model of the perception of time signature and phase, it is essential to combine this program with those that parse other elements such as pitch that also influence their perception. The model only uses note-length and so in some sense is only making ‘inexact’ matches. The pattern matcher could be extended to deal with more components (such as note pitch) where the use of inexact matches is more important. There should also be further developments in directions including: feedback between the MBPM and the rule-based grammar, extending the size of the corpus to allow for testing of the applicability of the combined program to a larger corpus and for the use of probabilistic techniques, and introducing the ability to identify and record point of change of time signature. The results of such developments will be presented in due course.

The results of this study, and in particular the feature that some of the ‘incorrect’ results were identified as such by all the programs tested, suggest that there are musical (and perceptual) explanations for these misidentifications. They show that the approaches developed in the Internal Structure Program may be useful in the future for the modeling of music perception and that the theories involved in the combination of memory-based and grammar-based approaches in the Combined Program are a further improvement.

Acknowledgments. Thanks go to Mark Steedman, supervisor of my MSc Thesis, and Rens Bod, University of Amsterdam, for their help and advice. To David Temperley and Daniel Sleator for the use of their program and advice on its

capabilities. To members of the Division of Informatics at the University of Edinburgh, Colin Matheson, Richard Shillcock and James Curran. Studentship no. K42200013185 from the ESRC is gratefully acknowledged.

References

1. Longuet-Higgins, C., Steedman, M.: On Interpreting Bach. In Longuet-Higgins, C., ed.: *Mental Processes*. MIT Press (1971) 82–104
2. Temperley, D., Sleator, D.: Modeling Meter and Harmony: A Preference-Rule Approach. *Computer Music Journal* **23** (1999) 10–27
3. Temperley, D., Sleator, D.: Music Analyzer. www.link.cs.cmu.edu/music-analysis/ (2000)
4. Steedman, M.: The Perception of Musical Rhythm and Metre. *Perception* **6** (1977) 555–569
5. Povel, D.J., Essens, P.: Perception of Temporal Patterns. *Music Perception* **2** (1985) 411–440
6. Parncutt, R.: A Perceptual Model of Pulse Salience and Metrical Accent in Musical Rhythms. *Music Perception* **11** (1994) 409–64
7. Bod, R.: Memory-Based Models of Melodic Analysis: Challenging the Gestalt Principles. *Journal of New Music Research* **30** (2001)
8. Crawford, T., Iliopoulos, C., Raman, R.: String Matching Techniques for Musical Similarity and Melodic Recognition. *Computing in Musicology* **11** (1998) 71–100
9. Charniak, E.: *Statistical Language Learning*. MIT Press (1993)
10. Cambouropoulos, E.: Extracting ‘Significant’ Patterns from Musical Strings: Some interesting Problems. Invited talk presented at LSD 2000 (2000)
11. Cambouropoulos, E., Crawford, T., C.S., I.: Pattern Processing in Melodic Sequences: Challenges, Caveats & Prospects. In: *Proceedings of the AISB’99 Convention*. (1999)
12. Bach, J.S.: Six suites for Violincello solo, BWV 1007-1012. Bärenreiter, 1950 (c. 1720)
13. Bach, J.S.: Six sonatas for Violin solo, BWV 1001-1006. Bärenreiter, 1950 (1720)
14. Sloboda, J.: *The Musical Mind: The cognitive psychology of music*. Oxford University Press (1985)
15. Spiro, N.: The Perception of Metre and Rhythm by the Listener. Manuscript, MSc. Thesis, University of Edinburgh (2001)
16. Temperley, D., Sleator, D.: An Algorithm for Harmonic Analysis. *Music Perception* **15** (1997) 21–68
17. Desain, P., Honing, H., eds.: *Music, Mind and Machine*. Thesis Publishers, Amsterdam (1992)

A Bayesian Approach to Key-Finding

David Temperley

Department of Music Theory
Eastman School of Music
26 Gibbs St.
Rochester, NY

dtemp@theory.esm.rochester.edu

Abstract. The key-profile model (originally proposed by Krumhansl and Schmuckler, and modified by Temperley) has proven to be a highly successful approach to key-finding. It appears that the key-profile model can be reinterpreted, with a few small modifications, as a Bayesian probabilistic model. This move sheds interesting light on a number of issues, including the psychological motivation for the key-profile model, other aspects of musical cognition such as metrical analysis, and issues such as ambiguity and expectation.

1. Introduction

How do listeners determine the key of a piece? This question has been the subject of considerable attention in recent years. A number of computational models have been proposed, reflecting a variety of approaches ([11], [1], [2], [9], [16]). One important proposal has been the key-profile model of Krumhansl and Schmuckler [8]. In this model, the distribution of pitch-classes in a piece is compared with an ideal distribution or “key-profile” for each key; the key whose profile best matches the pitch-class distribution of the piece is the preferred key. Elsewhere ([13], [14]) I have proposed a modified version of the original Krumhansl-Schmuckler (hereafter K-S) model which yields significantly improved performance; both the K-S model and my modified version will be discussed further below.

The purpose of the current paper is to examine the connection between the key-profile model of key-finding and the Bayesian method of cognitive modeling. Examination of these two approaches shows that they have a great deal in common; indeed, it could practically be argued that the key-profile model (particularly my own version of it) simply *is* a Bayesian probabilistic model. If this is true, then it is well worth noting for several reasons. First, the Bayesian perspective provides a more compelling psychological motivation for the key-profile model than anything that has been provided before. Secondly, the connection explored here may point to a broader connection between Bayesian modeling and a general approach in music cognition, the preference rule approach, which has been applied to a variety of problems in musical analysis, such as grouping, metrical analysis, harmonic analysis, and stream segregation. The Bayesian perspective may also be relevant to other aspects of

musical cognition, such as ambiguity and expectation. We will return to these ideas at the end of the paper.

I will begin by presenting the key-profile model, followed by a brief introduction to Bayesian modeling. I will then consider how the key-profile model might be reinterpreted within the Bayesian framework.

2. The Key-Profile Model of Key-Finding

The original key-profile model is based on a set of twelve-valued vectors, called key-profiles, representing the stability or compatibility of each pitch-class relative to each key. Key-profiles for C major and C minor are shown in Figure 1; profiles for other keys can be generated by simply shifting the values over by the appropriate number of steps. (For example, while in C major the value for C is 5.0 and the value for C# is 2.0, in C# major the value for C# would be 5.0 and the value for D would be 2.0.) These are the profiles used in my modified version of the key-profile model; they differ slightly from those used in Krumhansl and Schmuckler's original version, which were based on experimental data. (See [14] for an explanation of why these modified values were proposed.) It can be seen that the key-profiles reflect basic theoretical principles of tonal music. In the major profile, the values for the major scale are higher than those of chromatic pitches; within the major scale, values for pitches of the tonic triad are higher than for other diatonic pitches, and the value for the tonic is highest of all. The same is true of the minor profile (assuming the harmonic minor scale).

Given these profiles, the model judges the key of a piece by generating an "input vector" for the piece; this is, again, a twelve-valued vector, showing the total duration of each pitch-class in the piece. The correlation value is then calculated between each key-profile vector and the input vector. In essence, this involves taking the product of each key-profile value with the corresponding input-vector value, and summing these products.¹ The key yielding the maximum correlation value is the preferred key. Informally speaking, if the peaks in the input vector correspond with the peaks in the key-profile vector, the score for that key will be large. Figure 2a shows a simple example; for this passage, the model chooses C major as the correct key, just as it should.

My own tests of the Krumhansl-Schmuckler model revealed several problems with it, which were addressed in a modified version of the model (see [14] for details). One problem with the original model was that repeated notes appeared to carry too much weight. In a case such as Figure 2b, the repetitions of the E give a strong advantage to E major and E minor, even though C major is clearly the correct judgment. (A similar problem occurs when a pitch-class is duplicated in different octaves.) It seems that,

¹ This is a simplified version of the correlation formula, but appears to give essentially the same result. For discussion, see [14], pp. 173-6.

for a small segment of music at least, what matters most is the pitch-classes that are present, rather than how much each one is present. This problem was addressed in the following way. The model assumes some kind of segmentation of the piece which has to be provided in the input. (It works best to use segments of one to two seconds in length. In the tests reported below, I used metrical units—measures, half-measures, etc.—always choosing the smallest level of metrical unit that was longer than 1 second.) In constructing the input vector for a segment, the model simply gives each pitch-class a value of 1 if it is present in the segment and 0 if it is not. It then uses this “flat” input vector to calculate the correlations with the key-profiles. Since the input vector values are all 1 or 0, this simply amounts to adding the key-profile values for the pitch-classes that score 1 in the input vector. Consider Figure 2a; in this case, the score for C major is produced by adding the values in the C major profile for C, D, E, and F, yielding a score of $5.0 + 3.5 + 4.5 + 4.0 = 17.0$. This “flat-input” approach proved to achieve substantially better results than the “weighted-input” approach of the original K-S model.

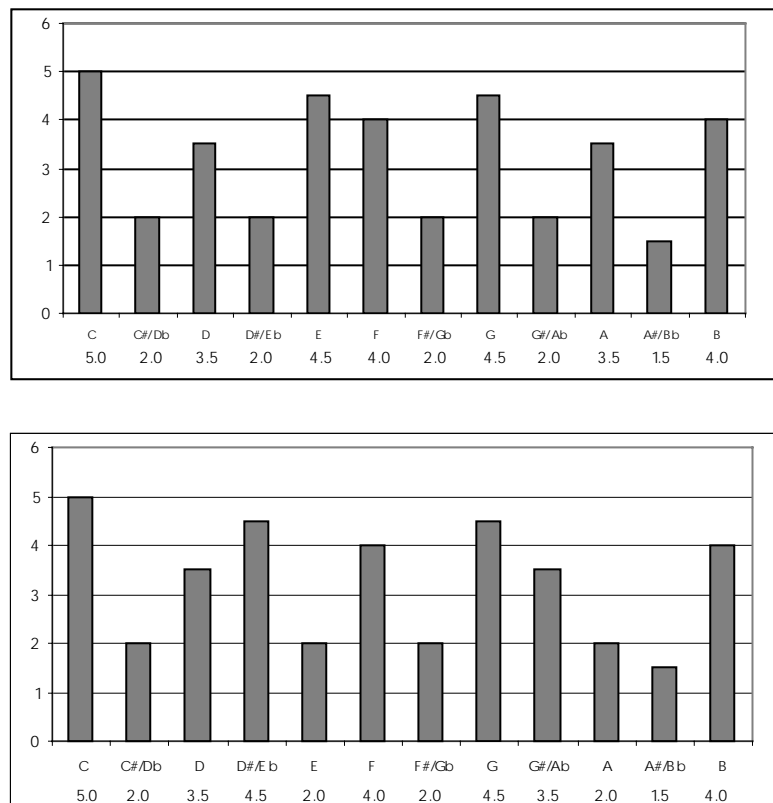


Fig. 1. Key-Profiles for C major (above) and C minor (below).



Fig. 2. (A) Bach, *Invention No. 1*, m. 1. (B) A problematic passage for the K-S model.

A further modification of the original key-profile model concerned modulations (changes in key). The original key-profile model is incapable of handling modulation; it simply produces a single key judgment for the entire input. Since the modified version of the model requires a segmentation of the piece in the input, one solution would be to judge the key of each segment independently; then modulations would simply emerge at points where the key of one segment differed from that of the previous one. However, this is not very satisfactory. Intuitively, key has a kind of inertia; once we are in a key, we prefer to remain in that key unless there is strong evidence to the contrary. To handle this, the model evaluates each segment independently, but imposes a “change penalty” if the key for one segment differs from the key for the previous segment. These penalties are then combined additively with the key-profile scores to choose the best key for each segment.

Once the change penalty is introduced, this means that the analysis for a particular segment is sensitive to context: the optimal analysis for one segment might depend on what precedes it. Moreover, if the model’s goal is to find the highest-scoring analysis overall, the analysis of a particular segment might depend on what follows, as well. If the model is to be assured of finding the optimal analysis then, it is necessary for it to consider all possible analyses of the entire piece, where an analysis is simply a labeling of every segment with a key, and choosing the highest-scoring one overall. Since the number of possible analyses increases exponentially with the number of segments, this approach is not feasible in practice, either for a computer or for a human mind. In the computational implementation I devised for the model, dynamic programming is used to find the highest-scoring analysis without actually generating them all.

The model was tested using the workbook and instructors’ manual accompanying the textbook *Tonal Harmony* by Stefan Kostka and Dorothy Payne ([7], [6]). The workbook contains a number of excerpts from pieces in the tonal repertory; the instructors’ manual provides harmonic analyses done by the authors, showing keys and modulations. The model was run on 46 excerpts from the workbook, and its output was compared with the analyses in the instructors’ manual. (The key-profiles were set prior to this test. However, the change penalty was adjusted to achieve optimal performance on the test.) Out of 896 segments, the program labeled 751 correctly, a rate of 83.8% correct. Inspection of the results showed that the program’s errors were mainly due to three things. First, the program’s rate of modulation was sometimes wrong: it sometimes modulated where the correct analysis did not (perhaps treating something only as a secondary harmony or “tonicization” instead), or vice

versa. Second, the program frequently had trouble with chromatic harmonies such as augmented sixth chords. It might be desirable to build special rules into the program for handling such cases, though this has not so far been attempted. A third source of error in the model concerned pitch spelling. The original key-profile model did not distinguish between different spellings of the same pitch: for example, *Ab* and *G#*. (I have called these categories “tonal pitch-classes” as opposed to the “neutral pitch-classes” of conventional theory.) However, it seemed likely that allowing the model to make such distinctions—so that, for example, *E* is more compatible with *C* major than *Fb* is—would improve the model’s performance. A version of the model was developed which recognized such distinctions, and its level of performance was indeed slightly higher (87.4% on the Kostka-Payne corpus). However, if our aim is to model perception, giving the program such information could be considered cheating, since the spelling of a pitch might in some cases only be inferable by using knowledge of the key. In the tests that follow, the “neutral-pitch-class” version of the key-profiles will be used (exactly as shown in Figure 1), thus avoiding this problematic issue.

We now turn to a brief introduction to the concepts of Bayesian modeling. We will then consider the connection between Bayesian models and the key-profile model.

3. Bayesian Modeling

Communication generally involves the transmission of a message of some kind from a producer to a perceiver. As perceivers, we are often given some kind of surface representation of a message (what I will simply call a *surface*); our task is to recover the underlying content that gave rise to it—the information that the sender was trying to convey—which I will simply call a *structure*. The problem is probabilistic in the sense that a single surface might arise from many different structures. We wish to know the structure that is most probable, given a particular surface—in the conventional notation of probability, we need to determine

$$\operatorname{argmax}_{\text{structure}} p(\text{structure} \mid \text{surface}) \quad (1)$$

A solution to this problem lies in Bayes’ rule. Bayes’ rule states that, for any two events *A* and *B*, the probability of *A* given *B* can be computed from the probability of *B* given *A*, as well as the overall probabilities (known as the “prior probabilities”) of *A* and *B*:

$$p(A \mid B) = \frac{p(B \mid A) p(A)}{p(B)} \quad (2)$$

In our terms, for a given surface and a given structure:

$$p(\text{structure} \mid \text{surface}) = \frac{p(\text{surface} \mid \text{structure}) p(\text{structure})}{p(\text{surface})} \quad (3)$$

To find the structure that maximizes the left side of the equation, we need only find the structure that maximizes the right side—and this turns out to be easier. Note, first

of all, that “ $p(\text{surface})$ ”—the overall probability of a given surface—will be the same for all values of “structure”. This means that it can simply be disregarded. Thus

$$\begin{aligned} & \operatorname{argmax}_{\text{structure}} p(\text{structure} \mid \text{surface}) \\ &= \operatorname{argmax}_{\text{structure}} p(\text{surface} \mid \text{structure}) p(\text{structure}) \end{aligned} \quad (4)$$

Thus, to find the most probable structure given a particular surface, we need to know—for every possible structure—the probability of the surface given the structure, and the prior probability of the structure.

The Bayesian approach has proven useful in modeling a number of perceptual and cognitive processes. One example is speech recognition. In listening to speech, we are given a sequence of phonetic units—phones—and we need to determine the sequence of words that the speaker intended. In this case, then, the sequence of phones is the surface and the sequence of words is the structure. The problem is that a single sequence of phones could result from many different words. Consider the phone sequence [ni] (this example is taken from [5]). Various words can be pronounced [ni], under certain circumstances: “new”, “neat”, “need”, “knee”, and even “the”. However, not all of these words are equally likely to be pronounced [ni]; $p(\text{surface} \mid \text{structure})$ may be higher for some words than others. The words also differ in their prior probabilities—that is to say, $p(\text{structure})$ is higher for some words than others. Once we know $p(\text{surface} \mid \text{structure})$ and $p(\text{structure})$ for each word (relative to the surface [ni]), we can take the product of these values; the structure maximizing this product is the most likely structure given the surface.

4. The Key-Profile Model as a Bayesian Model

Like speech recognition or other perceptual processes, key-finding requires inferring a structure from a surface. In this case, the structure is a sequence of keys; the surface is a pattern of notes. The problem is to infer the most likely structure, given a particular surface. According to Bayes’ rule, we can do this if we know—for all possible structures—the probability of the structure itself and the probability of the surface given the structure.

First consider the probability of a structure itself: a labeling of each segment with a key. (We will continue to assume a segmentation of the piece in the input.) Assume that for the initial segment of a piece, all 24 keys are equally probable. For subsequent segments, there is a high probability of remaining in the same key as the previous segment; switching to another key carries a lower probability. (We consider all key changes to be equally likely, though this may be an oversimplification.) The probability of a given key structure can then be calculated as the product of these “modulation scores” (S_m) for all segments. Let us assume, for any segment except the first, a probability of .8 of remaining in the same key as the previous segment, and a probability of .2/23 of changing to any other key. For a structure of four segments, C major - C major - C major - G major, the score will be

$$1/24 \times .8 \times .8 \times .2/23 = .000232 \quad (5)$$

Now, how do we calculate the probability of a surface given a structure? This problem could be solved in several different ways. I will propose one solution here, and then consider a second possibility later on. Let us suppose that, in each segment, the composer makes twelve independent decisions as to whether or not to use each pitch class. These probabilities can be expressed in a key-profile. We could base these key-profiles on actual data as to how often each pitch-class is used in segments of a particular key. Such data is shown in Table 1 for the Kostka-Payne corpus. As with the original key-profile model, the data is collapsed over all major keys and all minor keys, so that the profiles represent pitch-classes relative to keys—scale degrees, essentially. As an example, scale degree 1 (the tonic) occurs in .748 (74.8%) of segments in major keys; scale degree #4, by contrast, occurs in only .096 (9.6%) of segments. (It can be seen that the basic hierarchy of scale degrees in the profiles of Figure 1—with the tonic at the highest level, then other degrees of the tonic chord, then other diatonic degrees, then chromatic degrees—is reflected in these key-profiles as well; one odd exception is that 5 scores higher than 1 in minor.)

Table 1. The frequency of occurrence of each scale degree (relative to the current key) in the Kostka-Payne corpus, for major and minor keys. Numbers represent the proportion of segments in which the scale-degree occurs.

Scale degree	Major keys	Minor keys
1	.748	.712
#1/b2	.060	.084
2	.488	.474
#2/b3	.082	.618
3	.670	.049
4	.460	.460
#4/b5	.096	.105
5	.715	.747
#5/b6	.104	.404
6	.366	.067
#6/b7	.057	.133
7	.400	.330

The probability of a scale degree *not* occurring in a segment is, of course, 1 minus the score in the profile: for scale degree 1 in major keys, $1 - .748 = .252$. For a given key, the probability of a certain pitch-class set being used is then given by the product of the key-profile values—we could call these “pc scores” (S_{pc})—for all pitch-classes present in the segment (p), multiplied by the product of “absent-pc” scores ($S_{\sim pc}$) for all pitch-classes not present ($\sim p$).

$$\text{key-profile score} = \prod_p S_{pc} \prod_{\sim p} S_{\sim pc} \quad (6)$$

To find the most probable structure given a surface, we need to calculate $p(\text{structure}) p(\text{surface} \mid \text{structure})$. This can be calculated, for an entire piece, as the product of the modulation scores (S_m) and the key-profile scores for all segments (s):

$$p(\text{structure}) p(\text{surface} \mid \text{structure}) = \prod_s (S_m) \left(\prod_p (S_{pc}) \right) \left(\prod_{\sim p} (S_{\sim pc}) \right) \quad (7)$$

A standard move in Bayesian modeling is to express such a formula in terms of logarithms. Since the function $\ln x$ is monotonic, two values of $\ln x$ will always have the same ranking of magnitude as the corresponding values of x ; if our only aim is to find the maximum value of x , then using $\ln x$ instead works just as well. The logarithm for the formula above can be expressed as

$$\sum_s (\ln S_m + \sum_p \ln S_{pc} + \sum_{\sim p} \ln S_{\sim pc}) \quad (8)$$

Now the score is a sum of segment scores; each segment score is itself the sum of a modulation score, pc scores for present pc's, and absent-pc scores for absent pc's.

It can be seen that this is very similar to the key-profile model proposed earlier. If we pretend that the key-profile values and modulation penalties from the earlier model are really logarithms of other numbers, then the two models are virtually identical. There are some superficial differences. Since the scores in the Bayesian model are all logarithms of probabilities (numbers between 0 and 1), they will all be negative numbers. Also, the Bayesian model adds modulation scores for all segments, not just modulating segments. These are simply cosmetic differences which could be removed by scaling the values differently in the original model, without changing the results. There is one significant difference, however. In the earlier model, I simply summed the key-profile scores for the pc's that were present. In the Bayesian model, we also add “absent-pc” scores for pc's that are absent. It appears that this may be a significant difference between the two models.

A Bayesian version of the key-profile model was developed, exactly as just proposed. I used the key-profiles generated empirically from the Kostka-Payne corpus, as shown in Table 1. (Normally the corpus used for estimating the parameters should not be used for testing, but this was deemed necessary given the small amount of data available.) The only parameter to be set was the change penalty. Different values of the change penalty were tried, and the one that yielded optimal results was chosen. On the Kostka-Payne corpus, the Bayesian key-finding model achieved a correct rate of 77.1%, somewhat lower than correct rate of the earlier version (83.8%). Again, the main difference between the two models appears to be that in the second model, “absent-pc” scores are added to the key-profile scores. (The key-profiles are also different, but the key-profiles in the Bayesian model were generated directly from the test data; thus one would expect them to be optimal.) It is unclear why this would result in lesser performance for the Bayesian model. I intend to investigate this further.

One issue to consider, before continuing, is the probabilistic interpretation of key-profiles. In the model above, a key-profile is treated, essentially, as 12 independent probability functions indicating the probability of each scale degree occurring in a segment (and, thus, the probability of each pitch-class relative to each key). This approach is not ideal, since it requires a prior segmentation of the piece; there is little reason to think that such a segmentation is involved in human key-finding. An alternative approach—simpler, in some ways—would be to treat each key-profile as a

single probability function (so that the 12 values of the profile would sum to 1). This function could then be used to estimate the scale-degree probabilities of an event given a certain key. Events could be treated as independent; the probability of a note sequence given a key would then be given by the product of the key-profile scores for all events—or, in logarithmic terms, the sum of scores for all events. This method resembles the “weighted-input” approach of Krumhansl and Schmuckler’s original model, discussed earlier, in which the input vector reflects the number and duration of events of each pitch-class. The problem with this approach has already been noted: it tends to give excessive weight to repeated events. Initial tests of the key-profile model showed significantly better performance for the flat-input model than the weighted-input model. Thus it appears that treating the key-profiles as probability functions for independent events is unlikely to work very well. (Intuitively, in Figure 2b, the weighted-input approach assumes a generative model in which the composer decides to use C and G once, and then makes eight independent decisions to use E. But a more plausible model is that the composer decides to use certain pitch-classes, and then decides to repeat one of them.)

It is possible, however, that a more successful model could be developed based on the “weighted-input vector” idea. One way would be to assume that a musical surface is generated from a sparser, “reduced” representation of pitches, something like a “middleground” representation in a Schenkerian analysis. In such a representation, immediate repetitions of pitches such as those in Figure 2b (and also perhaps octave doublings and the like) would be removed. Possibly, a “weighted-input” model applied to such a reduction would produce better results; and it would also avoid the arbitrary segmentation required by the “flat-input” model. However, such an approach would present serious methodological problems, since it would require the middleground representation to be derived before key-finding could take place.²

5. Further Implications

My main aim in this paper has been simply to show that the key-profile model as proposed in [14] can be construed, with some small modifications, as a Bayesian probabilistic model. This connection is of interest for several reasons. First, the Bayesian approach provides a new perspective on the key-finding process; in a sense, it explains why the key-profile method is a sensible and effective way of determining key. The Bayesian viewpoint suggests that we can think of composition as a stochastic process in which a sequence of keys are generated and musical surfaces are then generated from these keys. (It is not *really* a stochastic process, but it can be effectively modeled in this way). From the perceiver’s point of view, given knowledge of scale-degree distributions and the likelihood of key changes, the intended sequence of keys can then be recovered. In this way, the key-profile model emerges as a very natural and logical way of recovering key structure.

² It is also instructive to compare the Bayesian formula with the correlation formula used in Krumhansl and Schmuckler’s original model (which, as already mentioned, differs slightly from the formula used in my model). However, space limitations prevent such a comparison here.

The relevance of Bayesian modeling to music cognition may go well beyond key-finding. The key-profile model as proposed in [14] can be viewed as a preference rule system—a model involving several rules which are used to evaluate many possible representations and select the preferred one. Preference rule models have been proposed for a variety of aspects of musical perception, including metrical analysis, grouping analysis, pitch reduction, harmonic analysis, and stream segregation ([10], [14]). In the case of the key-profile model, just two preference rules are involved:

Key-Profile Rule: Prefer to choose a key for each segment which is compatible with the pitches of the segment (according to the key-profiles);

Modulation Rule: Prefer to minimize the number of key changes.

It appears, in fact, that preference rule models generally can be construed as Bayesian models. One insight offered by the Bayesian view is that it suggests a distinction between two categories of preference rules. Some rules relate to the probability of a certain structure; we could call these “structure rules”. Others relate to the probability of a surface given a structure; we could call these “surface-to-structure rules”. In the case of the key-profile model, the Modulation Rule is a structure rule; the Key-Profile Rule is a structure-to-surface rule. Consider another example: metrical analysis. In this case, the structure is a row of beats (or a framework of levels of beats, but we will consider just a single level of beats for now), and the surface is once again a pattern of notes. The metrical model proposed in [14] (see also [15]) involves three main rules:

Event Rule: Prefer for beats to coincide with event-onsets;

Length Rule: Prefer for beats to coincide with longer events;

Regularity Rule: Prefer for beats to be roughly evenly spaced.

The process of deriving a row of beats involves optimizing over these three rules: choosing the metrical level which aligns beats with as many events as possible, especially long events, while maximizing the regularity of beats. In this case then, the Regularity Rule is a structure rule, indicating the probability of structures (more regular structures are more probable); the Event Rule and Length Rule are structure-to-surface rules, indicating the probability of surface patterns given a certain structure (patterns are more probable which align events with beats, especially longer events). The model in [14] evaluates a possible analysis by assigning to it scores from each of these three rules and summing these scores. It can be seen how a model of this kind could be reconstrued as a Bayesian model, much as we have reinterpreted the key-profile model in Bayesian terms. (Cemgil et al. ([3], [4]) propose a Bayesian model of metrical analysis, somewhat along these lines.)

Aside from problems of musical information extraction, I have discussed several other applications of preference rule models [14]. One is in the representation of ambiguity. With regard to key-finding, ambiguity is a phenomenon of recognized importance. Some pitch-sets are relatively clear in their tonal implications, such as major or minor scales or triads; others are ambiguous, in that they are compatible with several different scales (such as a diminished seventh chord, C-Eb-F#-A). In the

model proposed in [14], the ambiguity of a passage is captured in the numerical scores output by the model for different analyses: an ambiguous passage is one in which two or more different analyses are more or less “tied for first place”. Once again, this aspect of the key-profile model transfers straightforwardly to the Bayesian version; an ambiguous passage is one in which several analyses are more or less equally probable.

Another useful aspect of the Bayesian approach concerns the estimation of probabilities of musical surfaces. Bayesian theory tells us that the probability of a surface and a structure occurring in combination equals $p(\text{surface} \mid \text{structure}) p(\text{structure})$. It further tells us that the overall probability of a surface is equal to its probability in combination with a structure, summed over all possible structures:

$$p(\text{surface}) = \sum_{\text{structure}} p(\text{surface} \mid \text{structure}) p(\text{structure}) \quad (9)$$

Recall that the quantity $p(\text{surface} \mid \text{structure}) p(\text{structure})$ is exactly what must be generated, for all possible structures, in order to calculate the most probable structure for the surface. Thus it is not implausible to suppose that perceivers generate a measure of the probability of the surface itself, by summing these scores. In terms of key structure, a highly probable surface is one for which there is one (or perhaps more than one) highly probable key structure—one with relatively few modulations—which might, with high probability, give rise to it. Other surfaces are less probable, because no such structure exists. (Imagine a passage with a great deal of chromaticism, or rapid modulations, or both.) One application of this idea concerns expectation. It is well known that, in hearing a piece of music, listeners generally form expectations as to what is coming next; and some theorists have argued that the way music fulfills and denies expectations is an important part of its meaning and effect. It is natural to suppose that listeners’ expectations are governed by the same probabilistic models which (by hypothesis) govern their processing of musical input. With regard to key structure, we expect a continuation which is relatively probable given the probable structure—that is, one which adheres to the scale of the currently established key (though undoubtedly other constraints are involved in expectation as well). (Indeed, something along these lines has already been experimentally demonstrated; Schmuckler [12] has shown that listeners’ expectations for melodic continuations correspond closely with the key-profiles of the original K-S model.) The Bayesian approach would seem to offer a very natural method for modeling this process.³

Thus the Bayesian approach appears to offer a number of benefits. It provides a probabilistic basis for preference rule models generally, not merely for key-finding but for metrical analysis and other aspects of structure as well. It also suggests a way of modeling ambiguity and expectation. While many of the ideas in the paper are conjectural, the Bayesian framework seems to offer a promising avenue for modeling music cognition which deserves further exploration.

³ It might also be interesting to consider the probabilities of actual pieces or passages according to the key-profile model. This would indicate the probability of a passage actually being generated by the key-profile model—reflecting, perhaps, the “tonalness” of the passage. (A similar, but less satisfactory, method of measuring this was proposed in [14].)

References

1. Bharucha, J. J.: Music Cognition and Perceptual Facilitation: A Connectionist Framework. *Music Perception* 5 (1987) 1-30
2. Butler, D.: Describing the Perception of Tonality in Music: A Critique of the Tonal Hierarchy Theory and a Proposal for a Theory of Intervallic Rivalry. *Music Perception* 6 (1989) 219-42
3. Cemgil, A. T., Desain, P., Kappen, B.: Rhythm Quantization for Transcription. *Computer Music Journal* 24(2) (2000) 60-76
4. Cemgil, A. T., Desain, P., Kappen, B., Honing, H.: On Tempo Tracking: Tempogram Representation and Kalman Filtering. *Journal of New Music Research* 29 (2000) 259-73
5. Jurafsky, D., Martin, J. H.: *Speech and Language Processing*. Prentice-Hall, Upper Saddle River, NJ (2000)
6. Kostka, S.: *Instructor's Manual to Accompany Tonal Harmony*. McGraw-Hill, New York (1995)
7. Kostka, S., Payne, D.: *Workbook for Tonal Harmony*. McGraw-Hill, New York (1995)
8. Krumhansl, C.: *Cognitive Foundations of Musical Pitch*. Oxford University Press, Oxford (1990)
9. Leman, M.: *Music and Schema Theory*. Springer-Verlag, Berlin (1995)
10. Lerdahl, F., Jackendoff, R.: *A Generative Theory of Tonal Music*. MIT Press, Cambridge MA (1983)
11. Longuet-Higgins, C., Steedman, M.: On Interpreting Bach. *Machine Intelligence* 6 (1971) 221-41
12. Schmuckler, M.: Expectation in Music: Investigation of Melodic and Harmonic Processes. *Music Perception* 7 (1989) 109-50
13. Temperley, D.: What's Key for Key? The Krumhansl-Schmuckler Key-Finding Algorithm Reconsidered. *Music Perception* 17 (1999) 65-100
14. Temperley, D.: *The Cognition of Basic Musical Structures*. MIT Press, Cambridge MA (2001)
15. Temperley, D., Sleator, D.: Modeling Meter and Harmony: A Preference-Rule Approach. *Computer Music Journal* 23(1) (1999) 10-27
16. Vos, P., Van Geenen, E. W.: A Parallel-Processing Key-Finding Model. *Music Perception* 14 (1996) 185-224

Author Index

- Balaban, Mira 1
Bod, Rens 5
- Camurri, Antonio 4
Chew, Elaine 18
Conklin, Darrell 32
- Dannenberg, Roger B. 43
Dixon, Simon 58
- Furukawa, Koichi 94
- Goebl, Werner 58
Gouyon, Fabien 69
- Herrera, Perfecto 69
Höthker, Karin 168
Horton, Tim 81
Hu, Ning 43
- Igarashi, Soh 94
- Kamarotos, Dimitris 133
- Ozaki, Tomonobu 94
- Pachet, François 119
Phon-Amnuaisuk, Somnuk 155
Pikrakis, Aggelos 133
Povel, Dirk-Jan 144
- Reck Miranda, Eduardo 107
- Spevak, Christian 168
Spiro, Neta 183
- Temperley, David 195
Theodoridis, Sergios 133
Thom, Belinda 168
- Widmer, Gerhard 58
- Yeterian, Alexandre 69