2014 ISMIR
Taipei , Taiwan
Oct 27-31, 2014
The 15th International Society for Music Information Retrieval Conference

Proceedings of the 15th Conference of the International Society for Music Information Retrieval (ISMIR 2014)

October 27 - 31, 2014
Taipei, Taiwan

EDITED BY

Hsin-Min Wang
Yi-Hsuan Yang
Jin Ha Lee

Proceedings of the 15th Conference of the International Society for Music Information Retrieval (ISMIR)

EDITED BY

Hsin-Min Wang
Yi-Hsuan Yang
Jin Ha Lee

October 27 - 31, 2014

Taipei, Taiwan

http://www.terasoft.com.tw/conf/ismir2014/

ISMIR2014 is organized by the International Society for Music Information Retrieval.

Website: http://www.terasoft.com.tw/conf/ismir2014/

EDITED BY

Hsin-Min Wang (Academia Sinica, Taipei, Taiwan)
Yi-Hsuan Yang (Academia Sinica, Taipei, Taiwan)
Jin Ha Lee (University of Washington, Seattle, WA, USA)

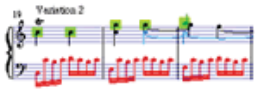
Sponsors

Gold Sponsors

Silver Sponsors

Bronze Sponsors

Co-Hosts



Corporate Partner

Organizing Committee

General Chairs
Jyh-Shing Roger Jang (National Taiwan University)
Masataka Goto (National Institute of Advanced Industrial Science and Technology (AIST))
Xiao Hu (University of Hong Kong)

Program Chairs
Hsin-Min Wang (Academia Sinica)
Yi-Hsuan Yang (Academia Sinica)
Jin Ha Lee (University of Washington)

Steering Committee
Shigeki Sagayama (University of Tokyo)
Malcolm Slaney (Microsoft)
Frank Soong (Microsoft Research Asia)
Shyh-Kang Jeng (National Taiwan University)
Arbee Chen (National Chengchi University)
Homer Chen (National Taiwan University)
Alvin Su (National Cheng Kung University)

Tutorial Chair
Li Su (Academia Sinica)

Music Chairs
Jeff (Chih-Fang) Huang (Kainan University, Chair of Taiwan Computer Music Association)
Yi-Wen Liu (National Tsing Hua University)
Yu-Chung Tseng (National Chiao Tung University)

Late-breaking and Demo Chair
Ju-Chiang Wang (University of California, San Diego)

Unconference Chair
Eric Humphrey (New York University)

Finance Chair
Ming-Feng Tsai (National Chengchi University)

Local Arrangement Chairs
Jia-Lien Hsu (Fu Jen Catholic University)
Wei-Ho Tsai (National Taipei University of Technology)

Web/Publication Chair
Ching-Hua Chuan (University of North Florida)
Gang Ren (University of Rochester)

Publicity Chair
Ye Wang (National University of Singapore)

Registration Chair
Jia-Ching Wang (National Central University)

Technical Program Committee

Jean-Julien Aucouturier (Institut de Recherche et Coordination Acoustique/Musique, France)
Juan Pablo Bello (New York University, USA)
Elaine Chew (Queen Mary University of London, UK)
Darrell Conklin (Universidad del Pais Vasco, Spain)
Sally Jo Cunningham (University of Waikato, New Zealand)
Matthew Davies (Institute for Systems and Computer Engineering of Porto, Portugal)
J. Stephen Downie (University of Illinois at Urbana-Champaign, USA)
Zhiyao Duan (Rochester University, USA)
Andreas Ehmann (Pandora, USA)
Dan Ellis (Columbia University, USA)
Slim Essid (ParisTech, France)
Arthur Flexer (Austrian Research Institute for Artificial Intelligence, Austria)
Rebecca Fiebrink (Princeton University, USA)
Ichiro Fujinaga (McGill University, Canada)
Emilia Gomez (Universitat PompeuFabra, Spain)
Perfecto Herrera (Universitat PompeuFabra, Spain)
Andre Holzapfel (University of Crete, Greece)
Hirokazu Kameoka (Tokyo University, Japan)
Peter Knees (Johannes Kepler University, Austria)
Audrey Laplante (University of Montreal, Canada)
Luis Gustavo Martins (Catholic University of Porto, Portugal)
Brian Mcfee (Columbia University, USA)
Meinard Mueller (University Erlangen-Nuremberg, Germany)
Geoffroy Peeters (Institut de Recherche et Coordination Acoustique/Musique, France)
Markus Schedl (Johannes Kepler University, Austria)
Erik Schmidt (Pandora Internet Radio, USA)
Joan Serra (Artificial Intelligence Research Institute, Spain)
Mohamed Sordo (Universitat PompeuFabra, Spain)
Bob Sturm (Aalborg University Copenhagen, Denmark)
Li Su (Academia Sinica, Taiwan)
Douglas Turnbull (Ithaca College, USA)
George Tzanetakis (University of Victoria, Canada)
Emmanuel Vincent (Institut National de Recherche en Informatique et en Automatique, France)
Anja Volk (Utrecht University, The Netherlands)
Ju-Chiang Wang (University of California, San Diego, USA)
Frans Wiering (Utrecht University, The Netherlands)
Kazuyoshi Yoshii (Kyoto University, Japan)

Reviewers

Samer Abdallah
Jakob Abesser
Teppo Ahonen
Joshua Albrecht
Anna Aljanaki
Vinoo Alluri
Joakim Anden
Tom Arjannikov
Andreas Arzt
Jean-Julien Aucouturier
Roland Badeau
Isabel Barbancho
Ana Maria Barbancho
Mathieu Barthet
Eric Battenberg
Dogac Basaran
Juan Pablo Bello
Emmanouil Benetos
Ciril Bohak
Antonio Bonafonte
Juanjo Bosch
Baris Bozkurt
Ashley Burgoyne
Sebastian Bock
Marcelo Caetano
Emilios Cambouropoulos
Estefania Cano
Mark Cartwright
Tak-Shing Chan
Chih-Ming Chen
Alex Chen
Elaine Chew
Ching-Hua Chuan
Andrea Cogliati
Gina Collecchia
Tom Collins
Darrell Conklin
Arshia Cont
Courtenay Cotton
Emanuele Coviello
Sally Jo Cunningham
Michael Scott Cuthbert
Roger Dannenberg
Matthew Davies
Bas de Haas
Alceu de Souza Britto Jr.
Norberto Degara
Francois Deliege
Arnaud Dessein
Johanna Devaney
Sander Dieleman
Christian Dittmar
J. Stephen Downie
Jonathan Driedger
Zhiyao Duan
Georgi Dzhambazov
Tuomas Eerola
Andreas Ehmann
Katherine Ellis
Dan Ellis
Valentin Emiya
Paulo Esquef
Slim Essid
Sebastian Ewert
Angel Faraldo
George Fazekas
Rebecca Fiebrink
Thomas Fillon
Derry Fitzgerald
Nicole Flaig
Arthur Flexer
Nuno Fonseca
Frederic Font
Jose Fornari
Ichiro Fujinaga
Satoru Fukayama
Daniel Gaertner
Martin Gasser
Ali Cenk Gedik
Mathieu Giraud
Aggelos Gkiokas
Fabien Gouyon
Maarten Grachten
Garth Griffin
Thomas Grill
David Grunberg
Enric Guaus
Sankalp Gulati
Michael Gurevich
Emilia Gomez
Philippe Hamel
Jinyu Han
Andrew Hankinson
Pierre Hanna
Mikael Henaff
Martin Hermant
Perfecto Herrera
Jason Hockman
Matt Hoffman
Andre Holzapfel
Yajie Hu
Po-Sen Huang
Anna Huang
Eric Humphrey
Fernando Iazzetta
Vaiva Imbrasaite
Vignesh Ishwar
Akinori Ito
Katsutoshi Itoyama
Ozgur Izmirli
Jordi Janer
Tristan Jehan
Kristoffer Jensen
Cyril Joder
Sergi Jorda
Hirokazu Kameoka
Kunio Kashino
Haruhiro Katayose
Damian Keller
Johannes Kepper
Corey Kereliuk
Tetsuro Kitahara
Anssi Klapuri
Peter Knees
Ian Knopke
Gopala Krishna Koduri
Noam Koenigstein
Alessandro Koerich
Filip Korzeniowski

Florian Krebs
Nadine Kroher
Lun-Wei Ku
Frank Kurth
Mathieu Lagrange
Paul Lamere
Thibault Langlois
Audrey Laplante
Olivier Lartillot
Kyogu Lee
Andreas Lehmann
Bernhard Lehner
Kjell Lemstrom
David Lewis
Dawen Liang
Cynthia Liem
Daryl Lim
Jen-Yu Liu
Yi-Wen Liu
Antoine Liutkus
Jorn Loviscach
Hanna Lukashevich
Esteban Maestre
Adolfo Maia Jr.
Michael Mandel
Matija Marolt
Luis Gustavo Martins
Agustin Martorell
Matthias Mauch
Rudolf Mayer
Brian McFee
Cory McKay
Matt McVicar
Nicola Montecchio
Josh Moore
Marcela Morvidone
Manuel Moussallam
Meinard Mueller
Daniel Mullensiefen
Hidehisa Nagano
Masahiro Nakano
Tomoyasu Nakano
Juhan Nam
Eric Nichols
Oriol Nieto
Mitsunori Ogihara
Yasunori Ohishi
Nobutaka Ono
Carthach ONuanain
Nicola Orio
Alexei Ozerov
Francois Pachet
Rui Pedro Paiva
Helene Papadopoulos
Tae Hong Park
Jouni Paulus
Johan Pauwels
Steffen Pauws
Geoffroy Peeters
Graham Percival
Antonio Pertusa
Pedro Pestana
Aggelos Pikrakis
Thomas Praetzlich
Matthew Prockup
Laurent Pugin
Marcelo Queiroz
Stanislaw Raczynski
Colin Raffel
Zafar Rafii
Mathieu Ramona
Andreas Rauber
Ana Rebelo
Gang Ren
Christophe Rhodes
Gael Richard
David Rizo
Matthias Robine
Martin Rocamora
Marcelo Rodriguez Lopez
Perry Roland
Gerard Roma
Justin Salamon
Andy Sarroff
Markus Schedl
Jan Schlueter
Erik Schmidt
Bjoern Schuller
Jeff Scott
Sertan Senturk
Xavier Serra
Joan Serra
Ian Simon
George Sioros
Paris Smaragdis
Jordan Smith
Lloyd Smith
Yading Song
Reinhard Sonnleitner
Mohamed Sordo
Ajay Srinivasamurthy
Adam Stark
Sebastian Stober
Dan Stowell
Bob Sturm
Bob Sturm
Li Su
Alvin Wen Yu Su
Atau Tanaka
damien tardieu
David Temperley
Steve Tjoa
Marko Tkalcic
Petri Toiviainen
Godfried Toussaint
Wei-Ho Tsai
Douglas Turnbull
George Tzanetakis
Julian Urbano
Yonatan Vaizman
Jan Van Balen
Remco Veltkamp
Emmanuel Vincent
Richard Vogl
Anja Volk
Ge Wang
Xing Wang
Ju-Chiang Wang
Ron Weiss
Felix Weninger
Tillman Weyde
Ian Whalley
Frans Wiering
Geraint Wiggins
Ben Wu
Fu-Hai Frank Wu
Guangyu Xia
Kazuyoshi Yoshii

Contents

Preface xvii

Keynotes xxi

Keynote 1: Automatic Music Transcription: From Music Signals to Music Scores xxii
Axel Roebel

Keynote 2: Sound and Music Computing for Exercise and (Re-)Habilitation xxiv
Ye Wang

Tutorials xxv

Tutorial 1: Why is Greek Music Interesting? Towards an Ethics of MIR xxvi
Andre Holzapfel and George Tzanetakis

Tutorial 2: Musical Structure Analysis xxvii
Meinard Mueller and Jordan Smith

Tutorial 3: Jingju Music: Concepts and Computational Tools for Analysis xxviii
Rafael Caro Repetto, Ajay Srinivasamurthy, Sankalp Gulati, and Xavier Serra

Tutorial 4: MiningSuite, a Comprehensive Framework for Music Analysis..... xxix
Articulating Audio (MIRtoolbox 2.0) and Symbolic Approaches
Olivier Lartillot

Oral Session 1: Classification 1

On Cultural, Textual and Experiential Aspects of Music Mood 3
Abhishek Singhi and Daniel Brown

Sparse Cepstral and Phase Codes for Guitar Playing Technique Classification 9
Li Su, Li Fan Yu, Yi-Hsuan Yang

Automated Detection of Single- and Multi-Note Ornaments in Irish Traditional Flute Playing 15
Münevver Köküer, Peter Jančovič, Islah Ali-MacLachlan, Cham Athwal

The Kiki-Bouba Challenge: Algorithmic Composition for Content-Based MIR 21
Research & Development
Bob Sturm and Nick Collins

Poster Session 1 27

Transfer Learning by Supervised Pre-Training for Audio-Based Music Classification 29
Aäron van den Oord, Sander Dieleman, Benjamin Schrauwen

Estimting Musical Time Information from Performed MIDI Files 35
Harald Grohganz, Michael Clausen, Meinard Müller

Estimation of the Direction of Strokes and Arpeggios 41
Isabel Barbancho, George Tzanetakis, Lorenzo J. Tardón, Peter F. Driessen, Ana M. Barbancho

Predicting Expressive Dynamics in Piano Performances Using Neural Networks 47
Sam van Herwaarden, Maarten Grachten, W. Bas de Haas

An RNN-based Music Language Model for Improving Automatic Music Transcription 53
Siddharth Sigtia, Emmanouil Benetos, Srikanth Cherla, Tillman Weyde, Artur S. d'Avila Garcez, Simon Dixon

Towards Modeling Texture in Symbolic Data 59
Mathieu Giraud, Florence Levé, Florent Mercier, Marc Rigaudière, Donatien Thorez

Computational Models for Perceived Melodic Similarity in A Cappella Flamenc 65
Nadine Kroher, Emilia Gómez, Catherine Guastavino, Francisco Gómez-Martín, Jordi Bonada

The VIS Framework: Analyzing Counterpoint in Large Datasets 71
Christopher Antila and Julie Cumming

Hierarchical Approach to Detect Common Mistakes of Beginner Flute Players 77
Yoonchang Han and Kyogu Lee

Robust Joint Alignment of Multiple Versions of a Piece of Music 83
Siying Wang, Sebastian Ewert, Simon Dixon

Formalizing the Problem of Music Description 89
Bob Sturm, Rolf Bardeli, Thibault Langlois, Valentin Emiya

An Association-based Approach to Genre Classification in Music 95
Tom Arjannikov and John Z. Zhang

Multiple Viewpiont Melodic Prediction with Fixed-Context Neural Networks 101
Srikanth Cherla, Tillman Weyde, Artur d'Avila Garcez

Verovio: A library for Engraving MEI Music Notation into SVG 107
Laurent Pugin, Rodolfo Zitellini, Perry Roland

Music Classification by Transductive Learning Using Bipartite Heterogeneous Networks 113
Diego Furtado Silva, Rafael Geraldeli Rossi, Solange Oliveira Rezende, Gustavo Enrique de Almeida Prado Alves Batista

Automatic Melody Transcription based on Chord Transcription 119
Antti Laaksonen

Audio-to-Score Alignment at the Note Level for Orchestral Recordings 125
Marius Miron, Julio José Carabias-Orti, Jordi Janer

A Compositional Hierarchical Model for Music Information Retrieval 131
Matevž Pesek, Aleš Leonardis, Matija Marolt

An Analysis and Evaluation of Audio Features for Multitrack Music Mixtures 137
Brecht De Man, Brett Leonard, Richard King, Joshua D. Reiss

Detecting Drops in Electronic Dance Music: Content Based Approaches 143
to a Socially Significant Music Event
Karthik Yadati, Martha Larson, Cynthia C. S. Liem, Alan Hanjalic

Towards Automatic Content-Based Separation of DJ Mixes into Single Tracks 149
Nikolay Glazyrin

MedleyDB: A Multitrack Dataset for Annotation-Intensive MIR Research 155
Rachel M. Bittner, Justin Salamon, Mike Tierney, Matthias Mauch, Chris Cannam, Juan P. Bello

Melody Extraction from Polyphonic Audio of Western Opera 161
A Method based on Detection of the Singer's Formant
Zheng Tang and Dawn A. A. Black

Codebook-Based Scalable Music Tagging with Poisson Matrix Factorization 167
Dawen Liang, John Paisley, Daniel P. W. Ellis

Oral Session 2: Transcription 173

Template Adaptation for Improving Automatic Music Transcription 175
Emmanouil Benetos, Roland Badeau, Tillman Weyde, Gaël Richard

Note-Level Music Transcription by Maximum Likelihood Sampling 181
Zhiyao Duan and David Temperley

Drum Transcription via Classification of Bar-Level Rhythmic Patterns 187
Lucas Thompson, Simon Dixon, Matthias Mauch

Oral Session 3: Symbolic 193

Developing Tonal Perception Through Unsupervised Learning 195
Carlos Eduardo Cancino Chacón, Stefan Lattner, Maarten Grachten

Exploiting Instrument-Wise Playing/Non-Playing Labels 201
for Score Synchronization of Symphonic Music
Alessio Bazzica, Cynthia C. S. Liem, Alan Hanjalic

Multi-Strategy Segmentation of Melodies 207
Marcelo Rodríguez-López, Anja Volk, Dimitrios Bountouridis

A Data Set for Computational Studies of Schenkerian Analysis 213
Phillip Kirlin

Systematic Multi-Scale Set-Class Analysis 219
Agustín Martorell and Emilia Gómez

Oral Session 4: Retrieval 225

Spotting a Query Phrase from Polyphonic Music Audio Signals 227
Based on Semi-Supervised Nonnegative Matrix Factorization
Taro Masuda, Kazuyoshi Yoshii, Masataka Goto, Shigeo Morishima

Bayesian Audio Alignment based on a Unified Model of Music Composition and Performance 233
Akira Maezawa, Katsutoshi Itoyama, Kazuyoshi Yoshii, Hiroshi Okuno

Automatic Set List Identification and Song Segmentation for Full-Length Concert Videos 239
Ju-Chiang Wang, Ming-Chi Yen, Yi-Hsuan Yang, Hsin-Min Wang

On Inter-Rater Agreement in Audio Music Similarity 245
Arthur Flexer

Poster Session 2 251

Emotional Predisposition of Musical Instrument Timbres with Static Spectra 253
Bin Wu, Andrew Horner, Chung Lee

Panako - A Scalable Acoustic Fingerprinting System 259
Handling Time-Scale and Pitch Modification
Joren Six and Marc Leman

Perceptual Analysis of the F-Measure to Evaluate Section Boundaries in Music 265
Oriol Nieto, Morwaread Farbood, Tristan Jehan, Juan Pablo Bello

Keyword Spotting in A-Capella Singing 271
Anna Kruspe

The Importance of F0 Tracking in Query-by-Singing-Humming 277
Emilio Molina, Lorenzo J. Tardón, Isabel Barbancho, Ana M. Barbancho

Vocal Separation using Singer-Vowel Priors Obtained from Polyphonic Audio 283
Shrikant Venkataramani, Nagesh Nayak, Preeti Rao, Rajbabu Velmurugan

Improving Query by Tapping via Tempo Alignment 289
Chun-Ta Chen, Jyh-Shing Roger Jang, Chun-Hung Lu

Automatic Instrument Classification of Ethnomusicological Audio Recordings 295
Dominique Fourer, Jean-Luc Rouas, Pierre Hanna, Matthias Robine

Music Analysis as a Smallest Grammar Problem 301
Kirill Sidorov, Andrew Jones, David Marshall

Frame-Level Audio Segmentation for Abridged Musical Works 307
Thomas Prätzlich and Meinard Müller

Creating a Corpus of Jingju (Beijing Opera) Music and Possibilities for Melodic Analysis 313
Rafael Caro Repetto and Xavier Serra

Modeling Temporal Structure in Music for Emotion Prediction using Pairwise Comparisons 319
Jens Madsen, Bjørn Sand Jensen, Jan Larsen

Musical Structural Analysis Database Based on GTTM 325
Masatoshi Hamanaka, Keiji Hirata, Satoshi Tojo

Theoretical Framework of A Computational Model of Auditory Memory 331
for Music Emotion Recognition
Marcelo Caetano and Frans Wiering

Improving Music Structure Segmentation using lag-priors 337
Geoffroy Peeters and Victor Bisot

Study of the Similarity between Linguistic Tones and Melodic Pitch Contours 343
in Beijing Opera Singing
Shuo Zhang, Rafael Caro Repetto, Xavier Serra

A Proximity Grid Optimization Method to Improve Audio Search for Sound Design 349
Christian Frisson, Stéphane Dupont, Willy Yvart, Nicolas Riche, Xavier Siebert, Thierry Dutoit

Introducing a Dataset of Emotional and Color Responses to Music 355
Matevž Pesek, Primož Godec, Mojca Poredoš, Gregor Strle, Jože Guna, Emilija Stojmenova, Matevž Pogačnik, Matija Marolt

In-Depth Motivic Analysis based on Multiparametric Closed Pattern 361
and Cyclic Sequence Mining
Olivier Lartillot

MIR_EVAL: A Transparent Implementation of Common MIR Metrics 367
Colin Raffel, Brian McFee, Eric J. Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, Daniel P. W. Ellis

Computational Modeling of Induced Emotion Using GEMS 373
Anna Aljanaki, Frans Wiering, Remco C. Veltkamp

Cognition-Inspired Descriptors for Scalable Cover Song Retrieval 379
Jan Van Balen, Dimitrios Bountouridis, Frans Wiering, Remco Veltkamp

A Cross-Cultural Study on the Mood of K-POP Songs 385
Xiao Hu1, Jin Ha Lee, Kahyun Choi, J. Stephen Downie

Cadence Detection in Western Traditional Stanzaic Songs using Melodic and Textual Features 391
Peter Van Kranenburg and Folgert Karsdorp

Discovering Typical Motifs of a Raga from One-Liners of Songs in Carnatic Music 397
Shrey Dutta and Hema A Murthy

Oral Session 5: Structure 403

Analyzing Song Structure with Spectral Clustering 405
Brian McFee and Daniel P.W. Ellis

Identifying Polyphonic Musical Patterns From Audio Recordings 411
Using Music Segmentation Techniques
Oriol Nieto and Morwaread Farbood

Boundary Detection in Music Structure Analysis using Convolutional Neural Networks 417
Karen Ullrich, Jan Schlüter, Thomas Grill

Oral Session 6: Cultures 423

Tracking the "Odd": Meter Inference in a Culturally Diverse Music Corpus 425
Andre Holzapfel, Florian Krebs, Ajay Srinivasamurthy

Transcription and Recognition of Syllable based Percussion Patterns 431
The Case of Beijing Opera
Ajay Srinivasamurthy, Rafael Caro Repetto, Harshavardhan Sundar, Xavier Serra

Oral Session 7: Recommendation & Listeners 437

Taste Space Versus the World: an Embedding Analysis of Listening Habits and Geography 439
Joshua Moore, Thorsten Joachims, Douglas Turnbul

Enhancing Collaborative Filtering Music Recommendation 445
by Balancing Exploration and Exploitation
Zhe Xing, Xinxi Wang, Ye Wang

Improving Music Recommender Systems: What Can We Learn from Research on Music Tastes? . 451
Audrey Laplante

Social Music in Cars 457
Sally Jo Cunningham, David M. Nichols, David Bainbridge, Hassan Ali

Poster Session 3 463

A Combined Thematic and Acoustic Approach for a Music Recommendation Service 465
in TV Commercials
Mohamed Morchid, Richard Dufour, Georges Linarès

Are Poetry and Lyrics All That Different? 471
Abhishek Singhi and Daniel G. Brown

Singing-Voice Separation from Monaural Recordings Using Deep Recurrent Neural Networks 477
Po-Sen Huang, Minje Kim, Mark Hasegawa-Johnson, Paris Smaragdis

Impact of Listening Behavior on Music Recommendation 483
Katayoun Farrahi, Markus Schedl, Andreu Vall, David Hauger, Marko Tkalčič

Towards Seamless Network Music Performance: Predicting an Ensemble's 489
Expressive Decisions for Distributed Performance
Bogdan Vera and Elaine Chew

Detection of Motor Changes in Violin Playing by EMG Signals 495
Ling-Chi Hsu, Yu-Lin Wang, Yi-Ju Lin, Cheryl D. Metcalf, Alvin W.Y. Su

Automatic Key Partition Based on Tonal Organization Information of Classical Music 501
Wang Kong Lam and Tan Lee

Bayesian Singing-Voice Separation 507
Po-Kai Yang, Chung-Chien Hsu, Jen-Tzung Chien

Probabilistic Extraction of Beat Positions from a Beat Activation Function 513
Filip Korzeniowski, Sebastian Böck, Gerhard Widmer

Geographical Region Mapping Scheme Based on Musical Preferences 519
Sanghoon Jun, Seungmin Rho, Eenjun Hwang

On Comparative Statistics for Labelling Tasks: What Can We Learn from MIREX ACE 2013? 525
John Ashley Burgoyne, W. Bas de Haas, Johan Pauwels

Merged-Output HMM for Piano Fingering of Both Hands 531
Eita Nakamura, Nobutaka Ono, Shigeki Sagayama

Modeling Rhythm Similarity for Electronic Dance Music 537
Maria Panteli, Niels Bogaards, Aline Honingh

MuSe: A Music Recommendation Management System 543
Martin Przyjaciel-Zablocki, Thomas Hornung, Alexander Schätzle, Sven Gauß, Io Taxidou, Georg Lausen

Tempo- and Transposition-Invariant Identification of Piece and Score Position 549
Andreas Arzt, Gerhard Widmer, Reinhard Sonnleitner

Gender Identification and Age Estimation of Users Based on Music Metadata 555
Ming-Ju Wu, Jyh-Shing Roger Jang, Chun-Hung Lu

Information-Theoretic Measures of Music Listening Behaviour 561
Daniel Boland and Roderick Murray-Smith

Evaluation Framework for Automatic Singing Transcription 567
Emilio Molina, Ana M. Barbancho, Lorenzo J. Tardón, Isabel Barbancho

What is the Effect of Audio Quality on the Robustness of MFCCs and Chroma Features? 573
Julián Urbano, Dmitry Bogdanov, Perfecto Herrera, Emilia Gómez, Xavier Serra

Music Information Behaviors and System Preferences of University Students in Hong Kong 579
Xiao Hu, Jin Ha Lee, Leanne Ka Yan Wong

LyricsRadar: A Lyrics Retrieval System Based on Latent Topics of Lyrics 585
Shoto Sasaki, Kazuyoshi Yoshii, Tomoyasu Nakano, Masataka Goto, Shigeo Morisihima

JAMS: A JSON Annotated Music Specification for Reproducible MIR Research 591
Eric J. Humphrey, Justin Salamon, Oriol Nieto, Jon Forsyth, Rachel M. Bittner, Juan P. Bello

On The Changing Regulations of Privacy and Personal Information in MIR 597
Pierre Saurel, Francis Rousseaux, Marc Danger

A Multi-Model Approach to Beat Tracking Considering Heterogeneous Music Styles 603
Sebastian Böck, Florian Krebs, Gerhard Widmer

Oral Session 8: Source Separation 609

Extending Harmonic-Percussive Separation of Audio Signals 611
Jonathan Driedger, Meinard Müller, Sascha Disch

Singing Voice Separation Using Spectro-Temporal Modulation Features 617
Frederick Yen, Yin-Jyun Luo, Tai-Shih Chi

Harmonic-Temporal Factor Decomposition Incorporating Music Prior Information 623
for Informed Monaural Source Separation
Tomohiko Nakamura, Kotaro Shikata, Norihiro Takamune, Hirokazu Kameoka

Oral Session 9: Rhythm & Beat 629

Design And Evaluation of Onset Detectors using Different Fusion Policies 631
Mi Tian, György Fazekas, Dawn A. A. Black, Mark Sandler

Evaluating the Evaluation Measures for Beat Tracking 637
Matthew Davies and Sebastian Böck

Improving Rhythmic Transcriptions via Probability Models Applied Post-OMR 643
Maura Church and Michael Scott Cuthbert

Classifying EEG Recordings of Rhythm Perception 649
Sebastian Stober, Daniel J. Cameron, Jessica A. Grahn

MIEX Oral Session 655

Ten Years of MIREX (Music Information Retrieval Evaluation eXchange) 657
Reflections, Challenges and Opportunities
J. Stephen Downie, Xiao Hu, Jin Ha Lee, Kahyun Choi, Sally Jo Cunningham, Yun Hao

Author Index 663

Preface

Welcome to the 15th ISMIR (International Society for Music Information Retrieval) Conference in Taipei, the capital city of Taiwan where you'll enjoy a mild sub-tropical climate, a multitude of exotic fruits, gourmet cuisine, easy and informal hospitality and a thriving and fascinating cultural scene.

The present volume contains all the peer-reviewed papers presented at ISMIR 2014.

- 252 papers were received, of which 222 were complete and well-formatted. These 222 papers were subjected to a double-blind review process in which both the authors and reviewers remained anonymous.
- 106 papers were accepted based on reviews and meta-reviews provided by 267 reviewers and 37 PC members. The overall acceptance rate is 47.7% (=106/222).
- 33 papers were selected for oral presentations based on both research quality and topic; the remainder 73 were selected for poster presentations.

The following table summarizes the publication statistics over the past ISMIRs:

Location	Oral	Poster	Total Papers	Total Pages	Total Authors	Unique Authors	Pages/ Paper	Authors/ Paper	U. Authors/ Paper
Plymouth	19	16	35	155	68	63	4.4	1.9	1.8
Indiana	25	16	41	222	100	86	5.4	2.4	2.1
Paris	35	22	57	300	129	117	5.3	2.3	2.1
Baltimore	26	24	50	209	132	111	4.2	2.6	2.2
Barcelona	61	44	105	582	252	214	5.5	2.4	2
London	57	57	114	697	316	233	6.1	2.8	2
Victoria	59	36	95	397	246	198	4.2	2.6	2.1
Vienna	62	65	127	486	361	267	3.8	2.8	2.1
Philadelphia	24	105	105	630	296	253	6	2.8	2.4
Kobe	38	85	123	729	375	292	5.9	3	2.4
Utrecht	24	86	110	656	314	263	6	2.	2.4
Miami	36	97	133	792	395	322	6	3	2.4
Porto	36	65	101	606	324	264	6	3.2	2.6
Curitiba	31	67	98	587	395	236	5.9	3	2.4
Taipei	**33**	**73**	**106**	**635**	**343**	**271**	**6**	**3.2**	**2.6**

As in past ISMIR conferences, the selected papers will be presented over a period of 3.5 days, preceded by a day of tutorials and followed by a half-day late-breaking/demo & unconference sessions. Moreover, we have two satellite events, including a music hack day of Hacking Audio and Music Research (HAMR) on Oct. 25 and 26, and Workshop on Computer Music and Audio Technology (WOCMAT) on Oct. 28 and 29. Highlights of the conference include:

Tutorials

Four tutorials will take place on Monday, with two on ethnic music and two on music analysis, providing a good balance between culture and technology.

Morning sessions:

- "Why is Greek music interesting? Towards an ethics of MIR" by Andre Holzapfel and George Tzanetakis
- "Musical structure analysis" by Meinard Müller and Jordan Smith

Afternoon sessions:

- "Jingju music: Concepts and computational tools for its analysis" by Rafael Caro Repetto, Ajay Srinivasamurthy, Sankalp Gulati and Xavier Serra
- "MiningSuite, a comprehensive framework for music analysis, articulating audio (MIRtoolbox 2.0) and symbolic approaches" by Olivier Lartillot

Keynote Speakers

We are honored to have two distinguished keynote speakers, Dr. Axel Roebel from IRCAM and Prof. Ye Wang from the National University of Singapore. Dr. Roebel will talk about audio music transcription, a challenging task and one of the ultimate goals of MIR, while Prof. Wang will describe innovative applications using music for exercise and rehabilitation.

- Axel Roebel: Automatic Music Transcription: From Music Signals to Music Scores
- Ye Wang: Sound and Music Computing for Exercise and (Re-)habilitation

MIREX

Music Information Retrieval Evaluation eXchange (MIREX) is a collective effort to evaluate cutting-edge methods for various MIR tasks, which is an integral part of ISMIR. This year we

are celebrating the 10th anniversary of MIREX with the following events on Friday morning:

- Prof. J. Stephen Downie's talk on "Ten Years of MIREX: Reflections, Challenges and Opportunities"
- A poster session on 20 MIREX tasks, including the Grand Challenge 2014 for user experience, and several other new tasks related to audio downbeat estimation, audio fingerprinting, and singing voice separation.

Late-breaking/Demo & Unconference

Friday afternoon is dedicated to late-breaking papers and MIR system demonstrations. Abstracts for these presentations will be available online. Moreover, after the late-breaking/demo session, we have a special "unconference" session (following the late-breaking sessions in ISMIR 2012 and 2013) in which participants can break into groups to discuss MIR issues of particular interest. This will be an informal and informative opportunity to get to know your peers and colleagues from around the world.

Music Program

On Wednesday night, a 2.5-hour concert will be held in the main conference hall. The goal of this year's music program is not only to encourage the use of MIR techniques in creating new music, but also to promote the composition of music that reflects Asian philosophy. The concert will feature 10 pieces selected from participant submissions, and 6 pieces specially-commissioned for the conference.

Social Events

As in past ISMIRs, a reception will be held on Monday night and a banquet on the following Thursday night. Moreover, we have "Women in MIR meeting" (for connecting female researchers in MIR) on Wednesday early morning, and "Mixer" (for people to get to know one another) on Wednesday late afternoon.

Get to know Taipei

The Bureau of Foreign Trade has kindly provided all foreign participants vouchers for English-language half- and whole-day local tours, including an all-around tour of Taipei (including visits to National Palace Museum, Shilin Night Market, Taipei 101, etc), a sky lantern tour in the mountains outside Taipei, a tea culture tour (including Maokong Gondola), a luxurious foot massage (with dinner at world-famous Michelin-starred Din Tai Fung), and a spa in Taipei's famous hot springs. More tour options can be found at ISMIR-2014 website. Be sure to take the chance to explore Taipei City and enjoy your stay at Taiwan!

The proceedings of ISMIR 2014 were made possible by the hard work of the organizing team, the PC members, the reviewers, and the authors, to whom we would like to express our deep gratitude. Specials thanks also go to this year's sponsors and supporters, including

MediaTek, KKBox, Pandora, Shazam, FormosaSoft, Merry, Realtek, Gracenote, iKala, Dolby, Samsung, Deansoft, Google, Doreso, Terasoft, III, ITRI, iNDIEVOX, and ACLCLP.

We hope you all have a wonderful and unforgettable stay at Taipei!

General Chairs

Jyh-Shing Roger Jang
National Taiwan University, Taiwan

Masataka Goto
National Institute of Advanced Industrial Science and Technology (AIST), Japan

Xiao Hu
University of Hong Kong, Hong Kong S.A.R., China

Program Chairs

Hsin-Min Wang
Academic Sinica, Taiwan

Yi-Hsuan Yang
Academic Sinica, Taiwan

Jin Ha Lee
University of Washington, USA


2014
ISMIR
Taipei , Taiwan
Oct 27-31, 2014
The 15th International Society for Music
Information Retrieval Conference

Keynotes

Keynote 1: Automatic Music Transcription: From Music Signals to Music Scores

Axel Roebel
Analysis/Synthesis Team, IRCAM
Paris, France

Abstract

Deriving the symbolic annotation of a piece of music form the audio signal is one of the important long term objectives of research in music information retrieval. The related signal processing task is denoted in short as: Automatic Music Transcription. It consists of deriving a complete score including the timing and frequency information of the notes (instruments and drums) present, and the instruments that have produced each note. A solution of this task would have an important impact on the research on MIR because it would open the door to use a symbolic music representation for the analysis of arbitrary audio signals. On the other hand one may note that the solution of the AMT task may benefit from results of many individual MIR tasks: e.g. tonality, chords, tempo, structure (notably repetitions), instrumentation.

The present talk aims to situate today's research related to the AMT problem. It will start with an introduction into the problem and the main obstacles to be resolved. Then a brief summary of the history of research related to Automatic Music Transcription will be presented leading to a description of the state of the art. An overview of the algorithms that are currently employed will be given together with a few examples using existing software implementations. Finally, potential directions for improving the state of the art AMT algorithms will be discussed covering instrument models (ANR project SOR2), multi channel audio analysis (EU FP7 project 3DTVS), as well as music theoretic constraints.

Biography

Axel Roebel is the head of the research team analysis/synthesis of sound at IRCAM. He received the Diploma in electrical engineering from Hannover University in 1990 and the Ph.D. degree (summa cum laude) in computer science from the Technical University of Berlin in 1993. In 1994 he joined the German National Research Centre for Information Technology (GMD-First) in Berlin where he continued his research on adaptive modeling of time series of nonlinear dynamical systems. In 1996 he became assistant professor for digital signal processing in the communication science department of the Technical University of Berlin. Since 2000 he is working at IRCAM doing research on spectral domain algorithms for sound analysis, synthesis and transformation. In summer 2006 he was Edgar-Varese guest professor for computer music at the Electronic studio of the Technical University of Berlin and in 2011 he became the head of the analysis/synthesis team.

His research centers around problems in audio signal analysis, synthesis and transformation covering music and speech. His recent research projects are related to spectral modeling of musical instruments (ANR project Sample Orchestrator II), audio to midi transcription (industrially funded project Audio2Note), detection and classification of sound events in multi channel audio (EU FP7 project 3DTVS), modeling and transformation of sound textures (ANR project PHYSIS), synthesis of singing voice (ANR project CHANTER). He is the main author of IRCAM's SuperVP software library for sound analysis and transformation.

Keynote 2: Sound and Music Computing for Exercise and (Re-)Habilitation

Ye Wang
Sound and Music Computing Lab, National University of Singapore
Singapore

Abstract

The use of music as an aid in healing body and mind has received enormous attention over the last 20 years from a wide range of disciplines, including neuroscience, physical therapy, exercise science, and psychological medicine. We have attempted to transform insights gained from the scientific study of music and medicine into real-life applications that can be delivered widely, effectively, and accurately. We have been trying to use music in evidence-based and/or preventative medicine. In this talk, I will describe three clinically-focused tools to facilitate the delivery of established music-enhanced therapies, harnessing the synergy of sound and music computing (SMC), mobile computing, and cloud computing technologies to promote healthy lifestyles and to facilitate disease prevention, diagnosis, and treatment in both developed countries and resource-poor developing countries. I will present some of our past and ongoing research projects that combine wearable sensors, smartphone apps, and a cloud-based therapy delivery system to facilitate music-enhanced physical and speech therapy, as well as the joys and pains working in such a multidisciplinary environment.

Biography

Ye Wang is an Associate Professor in the Computer Science Department at the National University of Singapore (NUS) and NUS Graduate School for Integrative Sciences and Engineering (NGS). He established and directed the sound and music computing (SMC) Lab (www.smcnus.org). Before joining NUS he was a member of the technical staff at Nokia Research Center in Tampere, Finland for 9 years. His research interests include sound analysis and music information retrieval (MIR), mobile computing, and cloud computing, and their applications in music edutainment and e-Health, as well as determining their effectiveness via subjective and objective evaluations. His most recent projects involve the design and evaluation of systems to support 1) therapeutic gait training using Rhythmic Auditory Stimulation (RAS), and 2) Melodic Intonation Therapy (MIT). He is also affiliated with the School of Computer Science of Fudan University and Harvard Medical School.


2014
ISMIR
Taipei , Taiwan
Oct 27-31, 2014
The 15th International Society for Music
Information Retrieval Conference

Tutorials

Tutorial 1: Why is Greek Music Interesting? Towards an Ethics of MIR

Andre Holzapfel
Department of Computer Science
University of Crete
Crete, Greece

George Tzanetakis
Department of Computer Science
University of Victoria
Victoria, BC, Canada

The initial goal of this tutorial is to provide an overview of musical styles in Greek culture, and to indicate various features of these musics that make them challenging and interesting for research in Music Information Retrieval (MIR). This tutorial is addressed to everybody interested in extending the diversity of her/his evaluation data, this way targeting generality of MIR approaches. On the other hand, the tutorial is aimed to provide a lively overview over a range of styles, that we hope will be informative and inspiring for any music listener. The tutorial will initially provide an overview of various styles of rural and urban music styles in the various areas of Greece. Then, we will focus on some styles we are particularly familiar with, and point out a variety of research tasks that is apparently quite challenging for those musics, such as beat tracking, mood estimation, transcription and chord estimation. In conclusion, inspired by the diversity of Greek music and the problems such diversity poses for our research, we reflect on the possibility of universal approaches to music processing, and discuss ethical implications for our work on recommendation systems for the musics of the world.

Tutorial 2: Musical Structure Analysis

Meinard Mueller
International Audio Laboratories Erlangen
Erlangen, Germany

Jordan Smith
Centre for Digital Music
Queen Mary University of London
London, UK

One of the attributes distinguishing music from other sound sources is the hierarchical structure in which music is organized. On the lowest level, one may have sound events such as individual notes, which are characterized by the way they sound, their timbre, pitch and duration. Such sound events combine to form larger structures such as motives, phrases, and chords, and these elements again form larger constructs that determine the overall layout of the composition. This higher structural level is specified in terms of musical parts and their mutual relations. For example, in popular music such parts can be the intro, chorus, and verse sections of the song. Or, in classical music, it can be the exposition, development, and recapitulation of a sonata movement. The goal of music structure analysis is to divide a given music representation into temporal segments that correspond to musical parts and to group these segments into musically meaningful categories.

In this tutorial, we review the most important segmentation and structure analysis principles and then discuss state-of-the-art techniques—many published in just the last few years—that exploit specific characteristics of music. The goals of this tutorial are: first, to explicitly discuss the simplifying model assumptions that each computational procedure is based on; second, to present recent research directions within music structure analysis and to show how the various principles can be applied and combined; and third, to discuss problems involving the evaluation of automated procedures and the use of so-called "ground-truth" reference annotations.

Tutorial 3: Jingju Music: Concepts and Computational Tools for Analysis

Rafael Caro Repetto, Ajay Srinivasamurthy, Sankalp Gulati, and Xavier Serra
Music Technology Group
University of Pompeu Fabra
Barcelona, Spain

Jingju (also known as Peking or Beijing opera) is one of the most representative genres of Chinese traditional music. From an MIR perspective, jingju music offers interesting research topics that challenge current MIR tools. The singing/acting characters in jingju are classified into predefined role-type categories with characteristic singing styles. Their singing is accompanied by a small instrumental ensemble, within which a high pitched fiddle, the jinghu, is the most prominent instrument within the characteristic heterophonic texture. The melodic conventions that form jingju modal systems, known as shengqiang, and the percussion patterns that signal important structural points in the performance offer interesting research questions. Also the overall rhythmic organization into pre-defined metrical patterns known as banshi makes tempo tracking and rhythmic analysis a challenging problem. Being Chinese a tonal language, the intelligibility of the text would require the expression of tonal categories in the melody, what offers an appealing scenario for the research of lyrics-melody relationship. The role of the performer as a core agent of the music creativity gives jingju music a notable space for improvisation. The lyrics and scores cannot be taken as authoritative sources, but as transcriptions of particular performances.

In this tutorial we will give an overview of Jingju music, of the relevant problems that can be studied from an MIR perspective and of the use of specific computational tools for its analysis. The tutorial will be organized in three parts. The first will be an introduction to Jingju from a musicological perspective, the second will cover diverse audio analysis tools of relevance to the study of Jingju (using http://essentia.upf.edu), and finally in the last part we will present and discuss specific examples of analyzing Jingju arias using those tools (work done in the context of http://compmusic.upf.edu).

Tutorial 4: MiningSuite, a Comprehensive Framework for Music Analysis, Articulating Audio (MIRtoolbox 2.0) and Symbolic Approaches

Olivier Lartillot
Department of Architecture, Design and Media Technology
Aalborg University
Aalborg, Denmark

This tutorial presents an in-depth introduction to MiningSuite, a continuation of MIRtoolbox, an innovative environment featuring a large range of audio and music analysis tools. Thanks to an adaptive syntactic layer on top of Matlab, complex design of audio or music analysis operations can be written in a very concise way through a simple assemblage of operators featuring a large set of options. The integration of expertise developed in separate areas of study into common modules encourages further reuse of these individual methods and their intermingling into a common framework. The MiningSuite features an innovative and integrative set of symbolic-based musicological tools related to, among others, segmentation in the form of hierarchical grouping, melodic reduction and modal analysis. An innovative method for exhaustive pattern mining allows detailed motivic and metrical analyses. Audio and symbolic representations (in MIDI and score-like formats) and processes are tightly interconnected: Operators dedicated to high-level musical features extraction (tonal, metrical, structural analyses) integrate signal processing, statistical and symbolic-based methods, and accept both symbolic and audio input.

The tutorial, suitable for both novices and experts, will give an overview of these different audio and symbolic approaches available in the framework, and will explain how to take benefit of the capabilities of the environment via the user-friendly syntax. At the last part of the tutorial, we will dwell a little into the description of the architecture of the MiningSuite (significantly different from the previous MIRtoolbox project) and of the core classes that govern the general capabilities of the framework. Will be described for instance the rich format of the output results, or a syntactic layer within the operators' Matlab code that simplifies and clarifies the code while taking care of the matrix optimisations in the background. We will explain how you can write new modules, and will present the open-source collaborative platform hosting the MiningSuite project, with versioning control, integrated source code browsing and code review, issue tracker and user's manual available in a wiki environment.

Oral Session 1

Classification

This Page Intentionally Left Blank

ON CULTURAL, TEXTUAL AND EXPERIENTIAL ASPECTS OF MUSIC MOOD

Abhishek Singhi **Daniel G. Brown**

University of Waterloo
Cheriton School of Computer Science
{asinghi,dan.brown}@uwaterloo.ca

ABSTRACT

We study the impact of the presence of lyrics on music mood perception for both Canadian and Chinese listeners by conducting a user study of Canadians not of Chinese origin, Chinese-Canadians, and Chinese people who have lived in Canada for fewer than three years. While our original hypotheses were largely connected to cultural components of mood perception, we also analyzed how stable mood assignments were when listeners could read the lyrics of recent popular English songs they were hearing versus when they only heard the songs. We also showed the lyrics of some songs to participants without playing the recorded music. We conclude that people assign different moods to the same song in these three scenarios. People tend to assign a song to the mood cluster that includes "melancholy" more often when they read the lyrics without listening to it, and having access to the lyrics does not help reduce the difference in music mood perception between Canadian and Chinese listeners significantly. Our results cause us to question the idea that songs have "inherent mood". Rather, we suggest that the mood depends on both cultural and experiential context.

1. INTRODUCTION

Music mood detection has been identified as an important Music Information Retrieval (MIR) task. For example, there is a MIREX audio mood classification task [12]. Though most automatic mood classification systems are solely based on the audio content of the song, some systems have used lyrics or have combined audio and lyrics features (*e.g.,* [3-5] and [6-7]) Previous studies have shown that combing these features improves classification accuracy (*e.g.,* [6-7] and [9]) but as mentioned by Downie et al. in [3], there is no consensus on whether audio or lyrical features are more useful.

Implicit in "mood identification" is the belief that songs have "inherent mood," but in practice this assignment is unstable. Recent work has focused on associating songs with more than one mood label, where similar mood tags are generally grouped together into the same label (*e.g.,* [2]), but this still tends to be in a stable listening environment.

Our focus is instead on the cultural and experiential context in which people interact with a work of music. People's cultural origin may affect their response to a work of art, as may their previous exposure to a song, their perception of its genre, or the role that a song or similar songs has had in their life experiences.

We focus on people's cultural origin, and on how they interact with songs (for example, seeing the lyrics sheet or not). Listening to songs while reading lyrics is a common activity: for example, there are "lyrics videos" (which only show lyrics text) on YouTube with hundreds of millions of views (*e.g.* "Boulevard of Broken Dreams"), and CD liner notes often include the text of lyrics. Our core hypothesis is that there is enough plasticity in assigning moods to songs, based on context, to argue that many songs have no inherent "mood".

Past studies have shown that there exist differences in music mood perception among Chinese and American listeners (*e.g.,* [8]). We surmised that some of this difference in mood perception is due to weak English language skills of Chinese listeners: perhaps such listeners are unable to grasp the wording in the audio. We expected that they might more consistently match the assignments of native English-speaking Canadians when shown the lyrics to songs they are hearing than in their absence. We addressed the cultural hypothesis by exploring Canadians of Chinese origin, most of whom speak English natively but have been raised in households that are at least somewhat culturally Chinese. If such Chinese-Canadians match Canadians not of Chinese origin in their assignments of moods to songs, this might at least somewhat argue against the supposition that being Chinese in culture had an effect on mood assignment, and would support our belief that linguistic skills account for at least some of the differences. Our campus has many Chinese and Chinese-Canadians, which also facilitated our decision to focus on these communities.

In this study we use the same five mood clusters as are used in the MIREX audio mood classification task and ask the survey participants to assign a song to only one mood cluster. A multimodal mood classification could be a possible extension to our work here. Earlier works in MIR [11] had used Russell's valence-arousal model where the mood is determined by the valence and arousal scores of the song; we stick to the simpler classification here.

In practice, our hypotheses about language expertise were not upheld by our experimental data. Rather, our data support the claim that both cultural background and experiential context have significant impact on the mood assigned by listeners to songs, and this effect makes us question the meaningfulness of "mood" as a category in MIR.

2. RELATED WORK

Mood classification is a classic task in MIR, and is one of the MIREX challenges. Several projects have used lyrics as part of the mood prediction task. Lu et al. [1] and Trohidis et al. [2] come up with an automatic mood classification system solely based on audio. Several projects like Downie et al. [3], Xiong et al. [4] and Chen et al. [5], have used lyrics as part of the mood prediction task. Downie et al. [3] show that features derived from lyrics outperform audio features in 7 out of the 8 categories. Downie et al. [6], Laurier et al. [7] and Yang et al. [9] show that systems which combine audio and lyrics features outperform systems using only audio or only lyrics features. Downie et al. [6] show that using a combination of lyrics and audio features reduces the need of training data required to achieve the same or better accuracy levels than only-audio or only-lyrics systems.

Lee et al. [8] study the difference in music mood perception between Chinese and American listeners on a set of 30 songs and conclude that mood judgment differs between Chinese and American participants and that people belonging to the same culture tend to agree more on music mood judgment. That study primarily used the common Beatles data set, which may have been unfamiliar to all audiences, given its age. Their study collected mood judgments solely based on the audio; we also ask participants to assign mood to a song based on its lyrics or by presenting both audio and lyrics together. To our knowledge no work has been done on the mood of a song when both audio and lyrics of the song is made available to the participants, which as we have noted is a common experience. Kosta et al. [11] study if Greeks and non-Greeks agree on arousal and valence rating for Greek music. They conclude that there is a greater degree of agreement among Greeks compared to non-Greeks possibly because of acculturation to the songs.

Downie et al. [3], Laurier et al. [7] and Lee et al. [8] use 18 mood tags derived from social tags and use multimodal mood classification system. Trohidis et al. [2] use multi modal mood classification into six mood clusters. Kosta et al. [11] use Russell's valence-arousal model which has 28 emotion denoting adjectives in a two dimensional space. Downie et al. [10] use the All Music Guide datasets to come up with 29 mood tags and cluster it into five groups. These five mood clusters are used in the MIREX audio music mood classification task. We use these clusters where each song is assigned a single mood cluster.

Mood Clusters	Mood Tags
Cluster 1	passionate, rousing, confident, boisterous, rowdy
Cluster 2	rollicking, cheerful, fun, sweet, amiable/good natured
Cluster 3	literate, poignant, wistful, bittersweet, autumnal, brooding
Cluster 4	humorous, silly, campy, quirky, whimsical, witty, wry
Cluster 5	aggressive, fiery, tense/anxious, intense, volatile, visceral

Table 1. The mood clusters used in the study.

3. METHOD

3.1 Data Set

We selected fifty very popular English-language songs of the 2000's, with songs from all popular genres, and with an equal number of male and female singers. We verified that the selected songs were international hits by going to the songs' Wikipedia pages and analyzing the peak position reached in various geographies.

We focus on English-language popular music in our study, because it is the closest to "universally" popular music currently extant, due to the strength of the music industry in English-speaking countries. Our data set includes music from the US, Canada and the UK and Ireland.

3.2 Participants

The presence of a large Chinese and Canadian population at our university, along with obvious cultural differences between the two communities, convinced us to use them for the study. We also include Canadians of Chinese origin; we are unaware of any previous MIR work that has considered such a group. We note that the Chinese-Canadian group is diverse: while some speak Chinese languages, others have comparatively little exposure to Chinese language or culture.

We recruited 100 participants, mostly university students, from three groups:

- 33 Chinese, living in Canada for less than 3 years.
- 33 Canadians, not of Chinese origin, born and brought up in Canada, with English as their mother tongue.
- 34 Canadians, of Chinese origin, born and brought up in Canada.

3.3 Survey

Each participant was asked to assign a mood cluster to each song in a set of 10 songs. For the first three songs

they saw only the lyrics; for the next three songs they only heard the first 90 seconds of the audio; and for the last four songs they had access to both the lyrics and the first 90 seconds of the audio simultaneously. They assigned each song to one of the five mood clusters shown in Table 1. We collected 1000 music mood responses for 50 songs, 300 each based solely either on audio or lyrics and 400 based on both audio and lyrics together. We note that due to their high popularity, some songs shown only via lyrics may have been known to some participants. We did not ask participants if this was the case.

4. RESULTS

We hypothesized that the difference in music mood perception between American and Chinese listeners demonstrated by Hu and Lee [8] is because of the weak spoken English language skills of Chinese students, and that this might give them some difficulty in understand the wording of songs; this is why we allowed our participants to see the lyrics for seven out of ten songs.

We had the following set of hypotheses before our study:

- People often assign different mood to the same song depending on whether they read the lyrics, or listen the audio or both simultaneously.
- Chinese-born Chinese listeners will have less consistency in the assignment of moods to songs than do Canadian-born non-Chinese when given only the recording of a song.
- Chinese-born Chinese will more consistently match Canadians when they are shown the lyrics to songs.
- Just reading the lyrics will be more helpful in matching Canadians than just hearing the music for Chinese-born Canadians.
- Canadian-born Chinese participants will be indistinguishable from Canadian-born non-Chinese participants.
- A song does not have an inherent mood: its "mood" depends on the way it is perceived by the listener, which is often listener-dependent.

4.1 Lyrics and music mood perception between cultures

We started this study with the hypothesis that difference in music mood perception between Chinese and Canadian cultures is partly caused by English language skills, and that if participants are asked to assign mood to a song based on its lyrics, we will see much more similarity in judgment between two different groups.

We used the Kullback-Leibler distance between the distribution of responses from one group and the distribution of responses from that group and another group to identify how similar the two groups' assignments of moods to songs were, and we used a permutation test to identify how significantly similar or different the two groups were. In Table 2, we show the number of songs for which different population groups are surprisingly similar. What we find is that the three groups actually agree quite a bit in uncertainty of assigning mood to songs when they are presented only with the recording: if one song has uncertain mood assignment for Canadian listeners, our Chinese listeners also typically did not consistently assign a single mood to the same song.

Our original hypothesis was that adding presented lyrics to the experience would make Chinese listeners agree more with the Canadian listeners, due to reduced uncertainty in what they were hearing. In actuality, this did not happen at all: in fact, presence of both audio and lyrics resulted in both communities having both more uncertainty and disagreeing about the possible moods to assign to a song.

This confusion in assigning a mood might be because a lot of hit songs (“Boulevard of Broken Dreams”, “Viva La Vida”, “You’re Beautiful”, *etc.*) use depressing words with very upbeat tunes. It could also be that by presenting both lyrics and audio changes the way a song is perceived by the participants and leads to a completely new experience. (We note parenthetically that this argues against using lyrics only features in computer prediction of song mood.)

The number of songs with substantial agreement between Chinese and Canadian, not of Chinese origin, participants remains almost the same with lyrics only and audio only, but falls drastically when both are presented together. (Note again: in this experiment, we are seeing how much the distribution of assignments differs for the two communities.) This contradicts our hypothesis that the difference in music mood perception between Chinese and Canadians is because of their difference in English abilities. It could of course be the case that many Chinese participants did not understand the meaning of some of the lyrics.

We had hypothesized that Canadians, of Chinese and non-Chinese origin would have very similar mood judgments because of similar English language skills but they do tend to disagree a lot on music mood. The mood judgment agreement between Chinese and Canadian, of Chinese and non-Chinese origin seem to be similar and we conclude that we can make no useful claims about the Chinese-Canadian participants in our sample.

On the whole we conclude that the presence of lyrics does not significantly increase the music mood agreement between Chinese and Canadian participants: in fact, being able to read lyrics while listening to a recording seems to significantly decrease the music mood agreement between the groups.

		lyrics	audio	audio+lyrics
Chinese	Canadians	25	22	14
Chinese	Canadian-Chinese	36	31	27
Chinese	non-Chinese Canadians	31	32	23
non-Chinese Canadians	Canadian-Chinese	36	29	31

Table 2. The number of statistically significantly similar responses between the different cultures for the three different ways they interact with the songs. “Canadians” refer to Canadians of both Chinese and non-Chinese origin.

4.2 Stability across the three kinds of experiences

We analyze the response from participants when they are made to listen to the lyrics, hear the audio or both simultaneously across all the three groups. We calculate Shannon entropy of this mood assignment for each of the 50 songs for the three ways we presented a song to the participants: some songs have much more uncertainty in how the participants assign mood cluster to them. We then see if this entropy is correlated across the three kinds of experience, using Spearman’s rank correlation coefficient of this entropy value between the groups. A rank correlation of 1.0 would mean that the song with the most entropy in its mood assignment in one experience category is also the most entropic in the other, and so on.

	Spearman’s rank correlation coefficient
only lyrics & only audio	0.0504
only lyrics & audio+lyrics	0.1093
only audio & audio+lyrics	0.0771

Table 3. Spearman’s rank correlation coefficient between the groups. The groups "only lyrics" and "only audio" identify when participants had access to only lyrics and audio respectively while “audio+lyrics” refers to when they had access to both simultaneously.

The low value of the correlation analysis suggests that there is almost no relationship be-tween "certainty" in music mood across the three different kinds of experiences: for songs like “Wake Up” by Hillary Duff and “Maria Maria” by Santana, listeners who only heard the song were consistent in their opinion that the song was from the second cluster, “cheerful”, while listeners who heard the song and read the lyrics were far more uncertain as to which class to assign the song to.

4.3 “Melancholy” lyrics

For each song, we identify the mood cluster to which it was most often assigned, and show these in Table 4.

Mood Clusters	only lyrics	only audio	audio+lyrics
Cluster 1	8	9	13
Cluster 2	5	15	11
Cluster 3	28	14	18
Cluster 4	4	6	3
Cluster 5	5	6	5

Table 4. The most commonly assigned mood clusters for each experimental context. Most songs are assigned to the third mood cluster when participants are shown only the lyrics.

Songs experienced only with the lyrics are most often assigned to the third mood cluster, which includes the mood tags similar to "melancholy". In the presence of audio or both audio and lyrics there is a sharp decline in the number of songs assigned to that cluster; this may be a consequence of "melancholy" lyrics being attached to surprisingly cheery tunes that cause listeners to assign them to the first two clusters. The number of songs assigned to the fourth and fifth cluster remains more similar across all experiential contexts. Even between the two contexts where the listener does hear the recording of the song, there is a good deal of inconsistency in assignment of mood to songs: for 27 songs, the most commonly identified mood is different between the "only audio" and “audio+lyrics” data.

4.4 Rock songs

We explored different genres in our test set, to see if our different cultural groups might respond in predictable ways when assigning moods to songs.

Things that might be considered loud to Chinese listeners could be perceived as normal to Canadian listeners. Thus, we examined how responses differed across these two groups for rock songs, of which we had twelve in our data set. We calculate the Shannon entropy of the response of the participants and present the result in table 5.

We see that for many rock songs, there is high divergence in the mood assigned to the song by our listeners from these diverse cultures. For seven of the twelve rock songs, the most diversity of opinion is found when listeners both read lyrics and hear the audio, while for three songs, all participants who only read the lyrics agreed exactly on the song mood (zero entropy).

We see that for 3 of 12 cases all the participants tend to agree on the mood for the song when they are given access to the lyrics. The data for lyrics only have lower entropy than audio for 5 of 12 cases and all five of these songs are "rebellious" in style. For the five cases where the audio-only set has lower entropy than lyrics-only, the song has a more optimistic feel to it. This is consistent with our finding in the last section about melancholy song lyrics.

For example, the lyrics of “Boulevard of Broken Dreams”, an extremely popular Green Day song, evoke isolation and sadness, consistent with the third mood cluster. On the other hand the song's music is upbeat which

may give the increased confusion when the participant has access to both the audio and lyrics for the song.

Song	only lyrics	only audio	audio+lyrics
"Complicated"	1.0	**0.918**	1.148
"American Idiot"	1.792	**1.459**	1.792
"Apologize"	**1.0**	1.25	1.0
"Boulevard of Broken Dreams"	**0.0**	1.792	2.155
"Bad Day"	1.792	1.459	**1.061**
"In the End"	**0.65**	1.459	1.061
"Viva La Vida"	**0.0**	1.5849	1.75
"It's My life"	**0.0**	0.65	1.298
"Yellow"	1.792	**0.65**	1.351
"Feel"	0.918	**0.650**	1.148
"Beautiful Day"	1.584	**1.459**	1.836
"Numb"	1.25	1.918	**0.591**

Table 5. Entropy values for rock songs for the three different categories.

4.5 Hip-Hop/ Rap

Lee et al. [8] show that mood agreement among Chinese and American listeners is least for dance songs. Our test set included five rap songs, and since this genre is often used at dance parties, we analyzed user response for this genre. Again, we show the entropy of mood assignment for the three different experiential contexts in Table 6.

What is again striking is that seeing the lyrics (which in the case of rap music is the primary creative element of the song) creates more uncertainty among listeners as to the mood of the song, while just hearing the audio recording tends to yield more consistency. Perhaps this is because the catchy tunes of most rap music pushes listeners to make a spot judgment as to mood, while being reminded of lyrics pushes them to evaluate more complexity.

In general we see that there is high entropy in mood assignment for these songs, and so we confirm the previous claim that mood is less consistent for "danceable" songs.

5. DOES MUSIC MOOD EXIST?

For music mood classification to be a well-defined task, the implicit belief is that songs have "inherent mood(s)," that are detectable by audio features. Our hypothesis is that many songs have no inherent mood, but that the perceived mood of a song depends on cultural and experiential factors. The data from our study supports our hypothesis.

We have earlier shown that the mood judgment of a song depends on whether it is heard to or its lyrics is read or both together, and that all three contexts produce mood assignments that are strikingly independent.

We have shown that participants are more likely to assign a song to the "melancholic" mood cluster when only reading its lyrics, and we have shown genre-specific cultural and experiential contexts that affect how mood appears to be perceived. Together, these findings suggest that that the concept of music mood is fraught with uncertainty.

The result of the MIREX audio mood classification task has had a maximum classification accuracy of less than 70% [12], with no significant recent improvements. Perhaps, this suggests that the field is stuck at a plateau, and we need to redefine "music mood" and change our approach to the music mood classification problem. Music mood is highly affected by external factors like the way a listener interacts with the song, the genre of the song, the mood and personality of the listener, and future systems should take these factors into account.

Song	only lyrics	only audio	audio+lyrics
"London Bridge"	1.459	**0.918**	1.405
"Don't Phunk With My Heart"	1.459	**1.251**	1.905
"I Wanna Love You"	**0.918**	1.459	1.905
"Smack That"	1.918	**1.792**	1.905
"When I'm Gone"	1.251	**0.918**	1.448

Table 6. Entropy values for hip-hop/ rap songs for the three different categories.

6. CONCLUSION

Our experiment shows that the presence of lyrics has a significant effect on how people perceive songs. To our surprise, reading lyrics alongside listening to a song does not significantly reduce the differences in music mood perception between Canadian and Chinese listeners. Also, while we included two different sets of Canadian listeners (Canadian-Chinese, and Canadians not of Chinese origin), we can make no useful conclusions about the Chinese-Canadian group.

We do consistently see that presence of both audio and lyrics reduces the consistency of music mood judgment between Chinese and Canadian listeners. This phenomenon may be because of irony caused by negative words presented in proximity to upbeat beats, or it could be that presenting both audio and lyrics together might be a completely different experience for the listener. This is an obvious setting for further work.

We have shown that the mood of a song depends on its experiential context. Interestingly, songs where listeners agree strongly about the mood of the song when only listening to the recording are often quite uncertain in their mood assignments when the lyrics are shown alongside the recording. Indeed, there is little correlation between the entropy in mood assignment between the different ways we presented songs to participants.

We also show that many "melancholy" lyrics are found in songs assigned to a more cheerful mood by listeners, again suggesting that for such songs, the extent to which listeners focus on the lyrics may influence how sad they view a song to be. We analyzed the mood assignments of participants on rock and hip-hop/rap songs. We see that people tend to agree much more to the mood of a hip-hop/rap song when they are made to listen to the song. We found that for rebellious/negative rock songs lyrics leads to more agreement in music mood but audio is better for positive songs. In both the genres we found that hearing audio while reading lyrics lead to less agreement on music mood of songs.

Our results suggest that music mood is so dependent on cultural and experiential context to make it difficult to claim it as a true concept. With the classification accuracy of mood classification systems reaching a plateau with no significant improvements we suggest that we need to redefine the term "music mood" and change our approach toward music mood classification problem.

A possible extension to our work could be running a similar study using a larger set of songs and more participants, possibly from more diverse cultures than the ones we studied. Future studies could focus on multi-modal music mood classification where a song could belong to more than one mood, to see if even in this more robust domain there is a stable way to assign songs to clusters of moods when they are experienced in different contexts. We also wonder if other contextual experiments can show other effects about mood: for example, if hearing music while in a car or on public transit, or in stores, makes the "mood" of a song more uncertain.

We fundamentally also wonder if "mood" as an MIR concept needs to be reconsidered. If listeners disagree more or less about the mood of a song when it is presented alongside its lyrics, that suggests a general uncertainty in the concept of "mood". We leave more evidence gathering about this concept to future work as well.

7. ACKNOWLEDGEMENTS

Our research is supported by a grant from the Natural Sciences and Engineering Research Council of Canada to DGB.

8. REFERENCES

[1] L. Lu, D. Liu, and H. Zhang, "Automatic mood detection and tracking of music audio signals", in *IEEE Transactions on Audio, Speech, and Language Processing*, volume 14, number 1, pages 5-18, 2006.

[2] K. Trohidis, G. Tsoumakas, G. Kalliris and I. Vlahvas, "Multi-label classification of music into emotions", in *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR '08),* pages 325-330, 2008.

[3] X. Hu and J.S. Downie, "When lyrics outperform audio for music mood classification: a feature analysis", in *Proceedings of the 11th International Conference on Music Information Retrieval (ISMIR '10)*, pages 1–6, 2010.

[4] H. He, J. Jin, Y. Xiong, B. Chen, W. Sun, and L. Zhao, "Language feature mining for music emotion classification via supervised learning from lyrics", in *Proceedings of Advances in the 3rd International Symposium on Computation and Intelligence (ISICA '08),* pages 426-435, 2008.

[5] Y. Hu, X. Chen and D. Yang, "Lyric-based song emotion detection with affective lexicon and fuzzy clustering method", in *Proceedings of the 10th International Conference on Music Information Retrieval (ISMIR '09),* pages 123-128, 2009.

[6] X. Hu and J. S. Downie, "Improving mood classification in music digital libraries by combining lyrics and audio", in *Proceedings of Joint Conference on Digital Libraries (JCDL),* pages 159–168, 2010.

[7] C. Laurier, J. Grivolla, and P. Herrera, "Multimodal music mood classification using audio and lyrics", in *Proceedings of the International Conference on Machine Learning and Applications (ICMLA '08),* pages 688-693, 2008.

[8] X. Hu and J.H. Lee, "A cross-cultural study of music mood perception between American and Chinese listeners," in *Proceedings of International Society for Music Information Retrieval (ISMIR '12),* pages 535-540, 2012.

[9] Y.H. Yang, Y.C. Lin, H.T. Cheng, I.B. Liao, Y.C. Ho, and H.H. Chen, "Toward multi-modal music emotion classification", in *Proceedings of Pacific Rim Conference on Multimedia (PCM '08),* pages 70-79, 2008.

[10] X. Hu. and J.S. Downie, "Exploring mood metadata: relationships with genre, artist and usage metadata", in *Proceedings of the 8th International Conference on Music Information Retrieval, (ISMIR '07),* pages 67-72, 2007.

[11] K. Kosta, Y. Song, G. Fazekas and M. Sandler, "A study of cultural dependence of perceived mood in Greek music", in *Proceedings of the 14th International Society for Music Information Retrieval Conference*, *(ISMIR '13),* pages 317-322, 2013.

[12] X. Hu, J. S. Downie, C. Laurier, M. Bay, and A. F. Ehmann: "The 2007 MIREX audio mood classification task: Lessons learned," in *Proceedings of the 9th International Society for Music Information Retrieval Conference*, *(ISMIR '08)* , pages 462–467, 2008.

SPARSE CEPSTRAL AND PHASE CODES FOR GUITAR PLAYING TECHNIQUE CLASSIFICATION

Li Su, Li-Fan Yu and Yi-Hsuan Yang
Research Center for Information Technology Innovation, Academia Sinica, Taipei, Taiwan
lisu@citi.sinica.edu.tw, a999frank@gmail.com, yang@citi.sinica.edu.tw

ABSTRACT

Automatic recognition of guitar playing techniques is challenging as it is concerned with subtle nuances of guitar timbres. In this paper, we investigate this research problem by a comparative study on the performance of features extracted from the magnitude spectrum, cepstrum and phase derivatives such as group-delay function (GDF) and instantaneous frequency deviation (IFD) for classifying the playing techniques of electric guitar recordings. We consider up to 7 distinct playing techniques of electric guitar and create a new individual-note dataset comprising of 7 types of guitar tones for each playing technique. The dataset contains 6,580 clips and 11,928 notes. Our evaluation shows that sparse coding is an effective means of mining useful patterns from the primitive time-frequency representations and that combining the sparse representations of logarithm cepstrum, GDF and IFD leads to the highest average F-score of 71.7%. Moreover, from analyzing the confusion matrices we find that cepstral and phase features are particularly important in discriminating highly similar techniques such as pull-off, hammer-on and bending. We also report a preliminary study that demonstrates the potential of the proposed methods in automatic transcription of real-world electric guitar solos.

1. INTRODUCTION

The use of various instrumental techniques is essential in music. A practical, interpretable automatic transcription system should provide information about playing techniques in addition to information about pitch or onset. For example, various fingering styles of the guitar, such as pull-off, hammer-on or bending, are all important elements of a guitar performance. A novice guitar player might be eager to learn the playing techniques employed in a musical excerpt of interest. Similar to some popular online automatic chord recognizer (e.g. Chordify [1]), a tool transcribing the note-by-note playing techniques of a guitar recording enhances the interactivity of music learning or listening experiences, and thereby offers important educational, recreational and even cultural values.

While extracting the pitch, onset, chord and instrument information from a musical excerpt has received great attention in the music information retrieval (MIR) community [3, 5, 16–18, 24], relatively little effort has been invested in transcribing the playing technique of instruments [23]. In addition, due to the use of various guitar tones (i.e. audio effects such as distortion, reverb, delay, and chorus effect) in everyday guitar performances, conventional timbre descriptors extracted from the spectrum might not be enough in modeling the electric guitar playing techniques. For instance, as the chorus effect is usually implemented by temporal delay [6], information about the phase spectrum might be important. On the other hand, for distortions that involve a filtering effect, cepstral features might be useful to characterize the respective source and filter components [8].

Motivated by the above observations, we present in this paper a comparative study evaluating the accuracy of playing technique classification of electric guitar using a variety of spectral, cepstral and phase features. The contribution of the paper is three-fold. First, to investigate more subtle variation of musical timbre, we compile an open dataset of 7 playing techniques of electric guitar, covering a variety of pitches and 7 tones (cf. Section 4). We have made the full dataset and its detailed information available online. [2] Second, as feature learning techniques such as dictionary learning and deep learning have garnered increasing attention in audio signal processing [12, 18, 22, 25], we evaluate the performance of sparse representations of audio signals using a dictionary adapted to the signals of interest (Section 5). Our evaluation shows that, to better model the playing techniques, it is useful to combine the sparse representation of different types of features, such as logarithm cepstrum and phase derivatives (Section 6). Finally, a preliminary study using a guitar solo demonstrates the potential of the proposed methods in automatic guitar transcription (Section 7).

2. RELATED WORK

Designing useful musical timbre descriptors has been a long-studied topic, and has achieved high performance in some fundamental problems such as instrument classifica-

[1] http://chordify.net/

[2] http://mac.citi.sinica.edu.tw/GuitarTranscription

tion of monophonic signals [13]. Nowadays, researchers turn to more challenging problems like multiple instrument recognition, which deals with a highly complicated timbre space [10]. Besides the complexity of multiple instruments, another challenge in timbre classification is to identify all the styles of timbre that one instrument can produce, such as to identify the playing techniques of an instrument. For exmaple, Abeßer *et al.* and Reboursière *et al.* [1, 21] pioneered the problem of automatic guitar playing technique classification, and used timber descriptors such as spectral flux, weighted phase divergence, spectral crust factors, brightness, and irregularity, amongst others. Most of these features are physically related to the characteristics of a plucked, vibrating string. However, these studies were not evaluated using a dataset comprising of various playing techniques and guitar tones.

In addition to larger and more realistic datasets, novel feature learning techniques might be helpful for modeling subtle timbre variations. Recently, sparse coding (SC) as a feature learning technique has been shown effective for MIR. This approach uses a predefined dictionary (codebook) to encode the prominent information of a given low-level feature representation of an input signal. One can encode any sensible audio representation by SC to capture different signal characteristics. For instance, Nam *et al.* [17] applied SC on short-time mel-spectra for music auto-tagging; Yu *et al.* [25] applied SC on logarithm cepstra and power-scale cepstra for predominant instrument recognition. Our work goes one step further and exploits phase information for SC.

3. ELECTRIC GUITAR PLAYING TECHNIQUE

Table 1 lists the 7 playing techniques we consider in this work. Most guitar solos are constructed with these techniques. For example, muting is widely used alternatively in place of normal in guitar *riffs* for rhythmic and punched phrases in rock and metal music, and bending is commonly considered to be the most important technique for expressing emotion.

To gain more insights into the signal-level properties of the playing techniques, in Fig. 1 we show the spectrograms (the first row) and the short-time cepstra (the second row) of the individual-note examples played with the 7 playing techniques. The first three columns are individual notes F4 of normal, vibrato and mute, the fourth column the consecutive notes F4–E4 of pull-off, and the last three columns the consecutive notes F4–#F4 of hammer-on, sliding and bending. The length of all samples is 0.6s. The window size is 46ms and the hop size is 10ms. From the spectrograms and the short-time cepstra, we see that muting has a 'noisier' attack and a faster decay comparing to normal. Moreover, hammer-on, sliding and bending have quite different transition behaviors, although they have the same note progression. The transition is sharp for hammer-on; smooth for bending; and there is a two-stage transition for sliding. Therefore, it seems that both the spectrogram and the cepstra contain useful information that can be exploited for automatic classification.

Technique	Description	# clips
Normal	Normal sound	2,009
Muting	Sounds muted (by right hand) to create great attenuation	385
Vibrato	Trilled sound produced by twisting left hand finger on the string	637
Pull-off	Sound similar to normal but with the smoother attack created by pulling off the string by left hand finger	525
Hammer-on	Sound similar to normal but with the smoother attack created by hammering on the string by left hand finger	581
Sliding	Discrete change to the target note with a smooth attack by left hand finger sliding through the string	1,162
Bending	Continuous change to the target note without an apparent attack by bending the string by left hand fingers	1,281

Table 1. Description of the playing techniques considered.

4. DATASET

While there is no publicly available dataset for guitar playing technique classification across different tones, we establish our own one with the aforementioned 7 playing techniques. The dataset is recorded by a professional guitarist using a recording interface, PreSonus' AudioBox USB, with bit depth of 24 bits and frequency response from 14 Hz to 70 kHz. We directly line-in the guitar to recording interface to catch every nuance of sound and exclude environmental noise. The guitar for recording is ESP's MII with Seymour Duncan's pickup and Ebony finger board, which is a high-quality guitar especially for metal and rock music. To make the quality of the sound recordings akin to that of real-world performance, we augment the single clean tone source to different guitar tones, which is done in the post-production stage using music production software Cubase. In addition, we assign each audio clips to 7 different guitar tones, which involve different levels of *distortion*, *reverb*, *delay* and *chorus*. Such tones may represent different genres such as rock, metal, funk, and country music solos. Moreover, the tones are carefully tuned to meet the quality for listening.

Because of the different characteristics of the techniques, the clips are recorded in slightly different ways. All the clips of sliding and bending have 2 notes for each clip with both whole step (2 semitones) and half step (1 semitone); all the clips of hammer-on and pull-off have 2 notes with only half step; and the clips of vibrato and muting have only one note for each clip. As for normal, we record whole steps, one steps, and single notes to cover all possible cases which might occur in the other 6 techniques. For sliding and bending, we record the clips only with the first three strings of the guitar since these techniques are less frequently applied on the last 3 strings. Similarly, we record muting clips with only the last 3 strings because it is commonly used in rhythm guitar with low pitch. Other

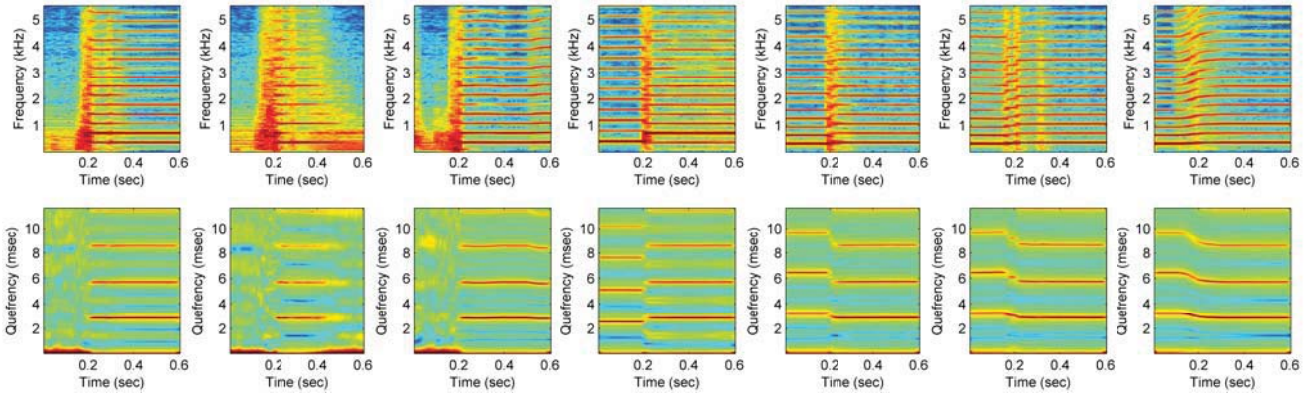


Figure 1. Spectrograms (the first row) and short-time cepstra (the second row) of the seven playing techniques considered in this study. From left to right: normal, muting, vibrato, pull-off, hammer-on, sliding, bending.

playing techniques are recorded with all the 6 strings. As a result, we can see from Table 1 that the numbers of clips of the 7 techniques are different, where normal has the largest number of 2,009 notes and muting has the smallest number of 385 notes. In total there are 6,580 clips.

5. METHODS

5.1 Feature representation

Our feature processing procedures have two steps: low-level feature extraction and sparse coding. In low-level feature extraction, we select spectrogram (SG), group-delay function (GDF), instantaneous frequency deviation (IFD), logarithm cepstrum (CL) and power cepstrum (CP), all of which are derived quantities from the short-time Fourier transformation (STFT):

$$\mathrm{S}^h(t,\omega) = \int x(\tau)\, h(\tau - t)\, e^{-j\omega\tau} d\tau = \mathrm{M}^h(t,\omega)\, e^{j\Phi^h(t,\omega)}, \quad (1)$$

where $x(t) \in \mathbb{R}$ is the input signal, $\mathrm{S}^h(t,\omega) \in \mathbb{C}$ stands for the two-dimensional STFT representation on time-frequency plane, and $h(t)$ refers to the window function. SG is the magnitude part of the STFT representation: $\mathrm{SG}^h(t,\omega) = |\mathrm{S}^h(t,\omega)|$. Phase spctrum is the imaginary part of the logarithm spectrum: $\Phi^h(t,\omega) = \mathrm{Im}\left(\log \mathrm{S}^h(t,\omega)\right)$. IFD and GDF are the derivative of phase Φ over time and frequency, respectively:

$$\mathrm{IFD}^h(t,\omega) = \frac{\partial \Phi^h}{\partial t} = \mathrm{Im}\left(\frac{\mathrm{S}^{\mathcal{D}h}(t,\omega)}{\mathrm{S}^h(t,\omega)}\right), \quad (2)$$

$$\mathrm{GDF}^h(t,\omega) = -\frac{\partial \Phi^h}{\partial \omega} - t = \mathrm{Re}\left(-\frac{\mathrm{S}^{\mathcal{T}h}(t,\omega)}{\mathrm{S}^h(t,\omega)}\right), \quad (3)$$

where $\mathcal{D}$ and $\mathcal{T}$ represent operators on window functions: $\mathcal{D}h(t) = h'(t)$ and $\mathcal{T}h(t) = t \cdot h(t)$. Detailed derivation procedures of GDF and IFD can be found in [2]. On the other hand, CL and CP are calculated as

$$\mathrm{CL}^h(t,q) = (\mathrm{S}^h)^{-1}\left(\log|\mathrm{S}^h(t,\omega)|\right), \quad (4)$$

$$\mathrm{CP}^h(t,q) = (\mathrm{S}^h)^{-1}\left(|\mathrm{S}^h(t,\omega)|^{1/3}\right), \quad (5)$$

where $(\mathrm{S}^h)^{-1}(\cdot)$ denotes the inverse STFT and q denotes quefrency [19]. Features derived from CL, such as the Mel-frequency cepstral coefficients (MFCCs), are often employed in audio signal processing [8, 16].

5.2 Sparse coding and dictionary learning

For any one of the aforementioned low-level features, denoted as $\mathbf{y} \in \mathbb{R}^m$, we further convert it to a sparse representation $\boldsymbol{\alpha} \in \mathbb{R}^k$ by SC. Specifically, SC involves the following l_1-regularized LASSO problem [7] to encode $\mathbf{y}$ over a given dictionary $\mathbf{D} \in \mathbb{R}^{m\times k}$.

$$\hat{\boldsymbol{\alpha}} = f_{\mathrm{SC}}(\mathbf{D},\mathbf{y}) = \arg\min_{\boldsymbol{\alpha}} \|\mathbf{y} - \mathbf{D}\boldsymbol{\alpha}\|_2^2 + \lambda\|\boldsymbol{\alpha}\|_1 . \quad (6)$$

The LASSO problem can be efficiently solved by for example the least angle regression (LARS) algorithm [7]. Moreover, the dictionary $\mathbf{D}$ is learned by the online dictionary learning (ODL) [15] implemented by the open-source package SPAMS (`http://spams-devel.gforge.inria.fr/`). The SC result when the input $\mathbf{y}$ is CL has been referred to as the sparse cepstral code [25].

6. EXPERIMENT

6.1 Experimental setup of individual notes

As Fig. 1 illustrates, the playing techniques can be better identified around the onsets for most cases. Therefore, our system starts from detecting the onset of each clip and then extracts features from each segment starting from the time before the onset by t_a second to the time after the onset by t_b second. We use the well-known spectral flux method [11] for onset detection, and empirically set $t_a = 0.1$ and $t_b = 0.2$ for all the clips. For STFT, we use Hanning window of window size 46 ms (1,024 samples) and hop size of 10 ms (441 samples). Under the sampling rate of 44.1 kHz, the dimension of all the low level features is 512 (i.e. considering only positive frequency).

We adopt a five-fold jack-knife cross-validation (CV) scheme for the evaluation. For all the fold partitions, the distribution of clips over the playing techniques is balanced. We learn both the classifier and the ODL dictionary from the training folds only, without using the test fold. The number of atoms k of each dictionary is set to

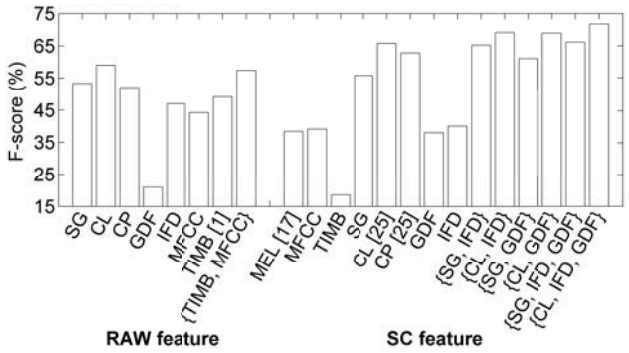


Figure 2. Average accuracies (in F-scores) of playing technique classification using various feature combination. Left part: RAW features; right part: SC features.

512. [3] After obtaining the frame-level sparse codeword $\boldsymbol{\alpha}$, a clip-level feature representation is constructed by mean pooling. Finally, the features, either with or without sparse coding, are fed into linear support vector machine (SVM) [9], with the parameter C optimized through an inside CV on the training data from the range $\{2^{-10}, 2^{-9}, \cdots, 2^{10}\}$. The evaluation results on the test set are reported in terms of F-score, which is the harmonic mean of precision and recall. All the evaluation is done at the clip-level.

We consider a number of baseline approaches for comparison. First, we use the MIRtoolbox (version 1.3.4) [14] to compute a total number of 41 features covering the temporal, spectral, cepstral and harmonic aspects of music signals (denoted as 'TIMB' in Fig. 2) as an implementation of a prior art on guitar playing technique classification [1]. Second, the conventional MFCC, ΔMFCC and $\Delta\Delta$MFCC are also used for their popularity (denoted as 'MFCC'). Third, we try the early fusion of MFCC and TIMB (i.e. by concatenating the corresponding clip-level representations to form a longer feature vector). Finally, for the features learned by SC, we note that the sparse representation of the mel-spectra (denoted as 'MEL') was used in [17], and the sparse representations of CL and CP were used in [25]. However, please note that the focus here is to compare the performance of using different features for the task, so our implementation does not faithfully follow the ones described in the prior arts. For example, Nam *et al.* uses automatic gain control as a pre-processing and uses multiple frame representation instead of frame-level features as input to feature encoding [17]. For simplicity the feature extraction and classification pipelines have been kept simple in this study.

We apply SC to all the five low-level features described in Section 5.1 and consider a number of early fusion of them. No normalization is performed for SC features. However, for non-SC features (referred to as 'RAW'), it is useful to apply a z-score normalization so that each feature dimension has zero mean and unit variance.

[3] Using an over-complete dictionary (i.e. $k \gg m$) usually improves the performance of SC features [25], but we leave this as a future work.

(a) SC+SG

	predicted class							F-score
actual class	nor	mut	vib	pul	ham	sli	ben	
nor	**92.6**	3.26	1.07	1.01	0.85	0.90	0.38	55.2
mut	44.0	**43.7**	6.94	1.13	0.32	0.97	2.90	56.1
vib	31.0	4.93	**63.8**	0.27	0.00	0.00	0.00	74.1
pul	21.0	1.75	0.00	**21.8**	16.9	34.2	4.47	29.7
ham	31.4	0.36	0.18	12.6	**25.8**	25.6	4.14	33.1
sli	11.9	0.94	0.00	7.92	10.9	**52.7**	15.6	46.1
ben	3.56	0.92	0.11	2.18	1.26	14.5	**77.5**	75.6

(b) SC+CL

	predicted class							F-score
actual class	nor	mut	vib	pul	ham	sli	ben	
nor	**95.6**	1.01	0.41	0.82	0.63	1.20	0.30	58.6
mut	38.9	**54.4**	4.35	0.16	0.00	0.65	1.45	66.3
vib	14.3	6.03	**79.7**	0.00	0.00	0.00	0.00	86.3
pul	27.2	0.58	0.19	**28.2**	14.6	25.6	3.69	38.9
ham	31.4	0.00	0.00	9.55	**38.2**	18.2	2.70	47.2
sli	14.9	1.42	0.00	4.43	7.26	**61.8**	10.2	56.7
ben	3.79	0.69	0.00	1.84	1.03	10.5	**82.2**	81.9

(c) SC+{CL,GDF,IFD}

	predicted class							F-score
actual class	nor	mut	vib	pul	ham	sli	ben	
nor	**95.6**	1.59	0.33	0.55	0.79	0.79	0.36	64.1
mut	35.0	**57.9**	4.52	0.32	0.16	0.32	1.77	68.7
vib	12.3	6.85	**80.8**	0.00	0.00	0.00	0.00	86.9
pul	19.6	0.58	0.19	**41.2**	11.7	22.5	4.27	52.0
ham	24.3	0.18	0.00	10.5	**45.8**	17.5	1.80	55.2
sli	10.2	1.13	0.19	5.66	6.60	**70.4**	5.85	65.0
ben	1.38	0.23	0.00	0.23	0.80	5.17	**92.2**	89.4

Table 2. Confusion matrix (in %) of playing technique classification of electric guitar individual notes using different feature combinations.

6.2 Experiment results

From the left hand side of Figure 2, we find that both RAW+TIMB [1] and RAW+MFCC perform worse than RAW+SG, RAW+CL and RAW+CP, possibly because the feature dimension of the latter three is larger. However, after fusing TIMB and MFCC, the F-score is improved to 57.4%, which is not significantly worse than the result of RAW+CL (i.e. 59.0%) under the two-tailed t-test. It turns out that using sophisticated features such as those computed by the MIRtoolbox does not offer gain for this task. Note that the F-score of random guess would be 1/7=14.3%, because each fold is balanced across the 7 techniques. The performance of most RAW features is greatly better than the chance level.

In contrast, from the right hand side of Figure 2, we find that SC features usually performs much better than the non-SC (i.e. RAW) counterparts. For example, SC+SG, SC+CL and SC+CP are better than RAW+SG, RAW+CL and RAW+CP, respectively. These improvements are all significant under the two-tailed t-test ($p<0.01$, d.f.=8). Similar observations have been made in existing works that

apply SC features to MIR tasks (e.g. [17, 25]). We also find that using SC+CL already leads to significantly better F-score than RAW+{TIMB,MFCC} ($p<0.0001$, d.f.$=8$). Moreover, from the data of SC features we see that fusing GDF and IFD generally improves the accuracy, and that the best F-score 71.7% is obtained by fusing sparse-coded cepstral and phase features (i.e. SC+{CL,GDF,IFD}). The F-score of SC+{SG,GDF,IFD} is worse (66.1%) than SC+{CL,GDF,IFD}, but is still significantly better than SC+SG. We also note that SC does not improve the performance for MEL, MFCC, TIMB and IFD. This implies that sophisticated features like TIMB are not suitable for SC. Although SC+IFD is worse than IFD, its fusion with other SC features still results in better performance. In a nutshell, this evaluation shows that it is promising to use SC for playing technique classification, especially when we fuse multiple features derived from STFT.

Table 2 displays the confusion matrices for three different feature combinations with sparse coding. Table 2(a) shows the result of SC+SG, from which we see that normal and bending have relatively high F-scores of 74.1% and 75.6% (see the rightmost column), yet the other five techniques have F-scores lower. We see that many playing techniques can be easily misclassified as normal. We also see ambiguities between for example pull-off versus sliding and hammer-on versus sliding, showing that such techniques are difficult to be discriminated from one another in the logarithm-scale spectrogram.

In contrast, we see from Table 2(b) that SC+CL leads to consistent improvement in F-score for all the playing techniques, comparing to SC+SG. The largest performance gain (+14.1%) is obtained for hammer-on. We also see that the ambiguity between normal and vibrato is mitigated.

Finally, comparing Tables 2 (b) and (c) we see that SC+{CL,GDF,IFD} consistently improves the F-score for all the playing techniques. More interestingly, it seems that adding phase derivatives effectively alleviate the aforementioned confusions without compromising the discriminability of other classes. The F-scores of all the playing techniques are now above 50.0%.

7. REAL-WORLD MUSIC

The automatic transcription flow contains frame-level pitch detection, onset detection, and playing technique classification, one after another. We adopt the method proposed by Peeters [20] and use spectral and cepstral features for pitch detection. For onset detection, we use again the spectral flux method [4, 11]. Finally, we apply the playing technique classifier trained from the individual note dataset to classify the playing techniques of the guitar solo.

We present a qualitative evaluation of a real-world electric guitar solo excerpt performed by same professional guitarist. It is an interpretation of Sonata Artica's Tallulah released in 2001, for the fragment 3:59–4:08. We show in the first two subfigures of Fig. 3 its scoresheet and spectrogram. In the third subfigure we show the pitch and onset, using black horizontal bars, gray horizontal bars, and vertical dashed lines to denote the estimated frame-level pitches, ground truth pitches, and estimated onsets, respectively. We see that the estimated pitches and onsets match the ground truth quite well, except for some cases such as the mismatch between the onset at 7.70s and the change of pitch at 7.84s, which probably results from the ambiguity of the onset of bending.

The last subfigure of Fig. 3 compares the result of SC+SG and SC+{CL,GDF,IFD} for playing technique classification. Since our classification is performed with respect to the detected onsets, the errors in the stage of onset detection will fully propagate into the stage of playing technique classification. Therefore, the techniques which are not characterized by onset (e.g., a long-sustaining vibrato) cannot be transcribed. A true positive of onset is defined as an onset position which is detected within 100ms of the ground truth onset time. A true positive of playing technique is accordingly defined as a correct prediction of playing technique at a true positive of onset. We can see that the performance of playing classification degrades a lot in comparison to the case of individual notes. Specifically, we have 7 true positives (4 normal and 3 bending) for SC+{CL,GDF,IFD} and 5 true positives (2 sliding, 2 bending and 1 normal) for SC+SG, while there are in total 17 targets in the ground truth. The 2 muting at 2.38s and 4.60s and the hammer-on at 9.24 second are not recalled by both methods. Although SC+{CL,GDF,IFD} fails to recall sliding, SC+SG recalls 2 sliding. While SC+{CL,GDF,IFD} has many false positives of vibrato, SC+SG has many false positives of sliding. In general, SC+{CL,GDF,IFD} performs better.

The two estimated events at 4.11s and 5.80s are interesting. Although the two events do not present in the ground truth, the prediction of SC+{CL,GDF,IFD} is musically correct as the two false alarms of onset indeed occur in a long-sustaining vibrato. In contrast, SC+SG misclassifies the two events as pull-off and sliding, respectively.

8. CONCLUSION

In this study, we have reported a comparative study on the performance of a number of timbre modeling methods for the relatively unexplored task of guitar playing technique classification. The evaluation is performed on a large-scale individual-note dataset comprising of 6,580 clips and a real-world guitar solo recording. Our evaluation shows that sparse coding works well in learning features that are useful for the task, and that using features extracted from the cepstra and phase derivatives helps resolve the confusion among similar playing techniques. We also report a qualitative evaluation on guitar solo transcription. We are currently collecting more individual notes and solos to deeply understand the signal-level characteristics for these playing techniques. Although the present study might be at best preliminary, we hope it can call for more attention towards playing technique modeling.

9. ACKNOWLEDGMENTS

This work was supported by the Academia Sinica Career Development Award 102-CDA-M09.

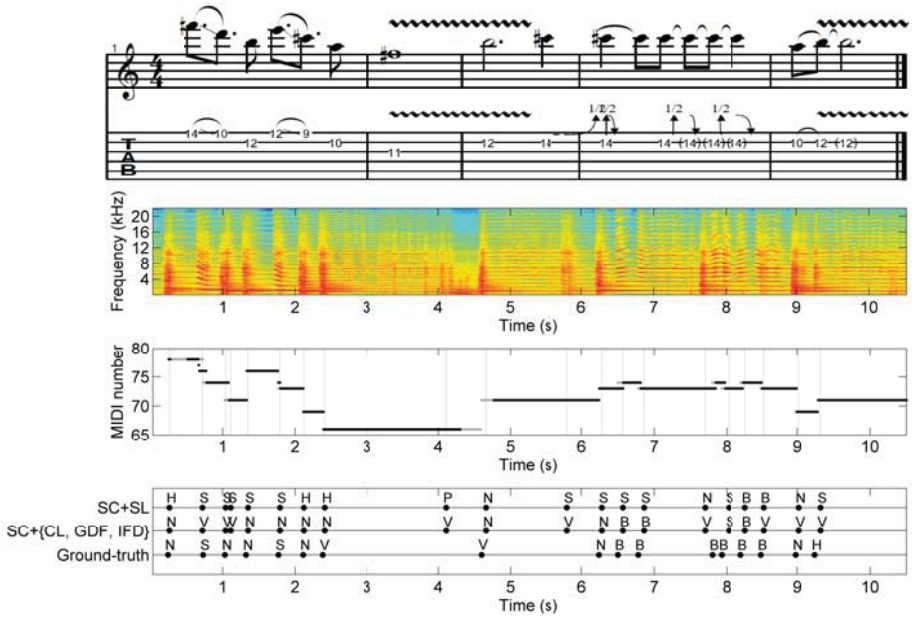


Figure 3. Result of transcribing a real-world guitar solo excerpt. From top to bottom: scoresheet, guitar tab, spectrogram, pitch and onset (gray bar: ground truth; black bar: estimated pitch; vertical dashed line: estimated onset), and result of playing technique classification by using SC+SG and SC+{CL,GDF,IFD}. Abbreviation: N=normal, V=vibrato, M=muting, P=pull-off, H=hammer-on, S=sliding, B=bending.

10. REFERENCES

[1] J. Abeßer et al. Feature-based extraction of plucking and expression styles of the electric bass guitar. In *ICASSP*, pages 2290–2293, 2010.

[2] F. Auger and P. Flandrin. Improving the readability of time-frequency and time-scale representations by the method of reassigment. *IEEE Trans. Sig. Proc.*, 43(5):1068–1089, 1995.

[3] A. M. Barbancho et al. Automatic transcription of guitar chords and fingering from audio. *IEEE Trans. Audio, Speech, and Language Processing*, 20(3):915–921, 2012.

[4] J. P. Bello et al. A tutorial on onset detection in music signals. *IEEE Speech Audio Process.*, 13(5-2):1035–1047, 2005.

[5] E. Benetos et al. Automatic music transcription: challenges and future directions. *J. Intelligent Information Systems*, 41(3):407–434, 2013.

[6] J. Dattorro. Effect design, part 2: Delay line modulation and chorus. *J. Audio engineering Society*, 45(10):764–788, 1997.

[7] B. Efron et al. Least angle regression. *Annals of Statistics*, 32:407–499, 2004.

[8] A. Eronen and A. Klapuri. Musical instrument recognition using cepstral coefficients and temporal features. In *ICASSP*, pages 753–756, 2000.

[9] R.-E. Fan et al. LIBLINEAR: A library for large linear classification. *J. Machine Learning Research*, 2008.

[10] P. Hamel et al. Automatic identification of instrument classes in polyphonic and pply-instrument audio. In *ISMIR*, 2009.

[11] A. Holzapfel et al. Three dimensions of pitched instrument onset detection. *IEEE Trans. Audio, Speech, Language Process.*, 18(6):1517–1527, 2010.

[12] E. J. Humphrey et al. Feature learning and deep architectures: new directions for music informatics. *J. Intelligent Information Systems*, 41(3):461–481, 2013.

[13] A. Klapuri and M. Davy, editors. *Signal Processing Methods for Music Transcription*, chapter 6. Springer, 2006.

[14] O. Lartillot and P. Toiviainen. A Matlab toolbox for musical feature extraction from audio. In *DAFx*, 2007.

[15] J. Mairal et al. Online dictionary learning for sparse coding. In *Int. Conf. Machine Learning*, pages 689–696, 2009.

[16] M. Müller et al. Signal processing for music analysis. *IEEE J. Sel. Topics Signal Processing*, 5(6):1088–1110, 2011.

[17] J. Nam et al. Learning sparse feature representations for music annotation and retrieval. In *ISMIR*, pages 565–560, 2012.

[18] K. O'Hanlon and M. D Plumbley. Automatic music transcription using row weighted decompositions. In *ICASSP*, 2013.

[19] A. V. Oppenheim and R. W. Schafer. *Discrete-Time Signal Processing*. Prentice Hall, 2010.

[20] G. Peeters. Music pitch representation by periodicity measures based on combined temporal and spectral representations. In *ICASSP*, 2006.

[21] L. Reboursière et al. Left and right-hand guitar playing techniques detection. In *NIME*, 2012.

[22] L. Su and Y.-H. Yang. Sparse modeling for artist identification: Exploiting phase information and vocal separation. In *ISMIR*, pages 565–560, 2013.

[23] L. Su and Y.-H. Yang. Sparse modeling of subtle timbre: a case study on violin playing technique. In *WOCMAT*, 2013.

[24] K. Yazawa et al. Automatic transcription of guitar tablature from audio signals in accordance with player's proficiency. In *ICASSP*, 2014.

[25] L.-F. Yu et al. Sparse cepstral codes and power scale for instrument identification. In *ICASSP*, 2014.

AUTOMATED DETECTION OF SINGLE- AND MULTI-NOTE ORNAMENTS IN IRISH TRADITIONAL FLUTE PLAYING

Münevver Köküer[1,2], Peter Jančovič[2], Islah Ali-MacLachlan[1], Cham Athwal[1]
[1]DMT Lab, Birmingham City University, UK
[2]School of Electronic, Electrical & Systems Engineering, University of Birmingham, UK
{munevver.kokuer, islah.ali-maclachlan, cham.athwal}@bcu.ac.uk
p.jancovic@bham.ac.uk

ABSTRACT

This paper presents an automatic system for the detection of single- and multi-note ornaments in Irish traditional flute playing. This is a challenging problem because ornaments are notes of a very short duration. The presented ornament detection system is based on first detecting onsets and then exploiting the knowledge of musical ornamentation. We employed onset detection methods based on signal envelope and fundamental frequency and customised their parameters to the detection of soft onsets of possibly short duration. Single-note ornaments are detected based on the duration and pitch of segments, determined by adjacent onsets. Multi-note ornaments are detected based on analysing the sequence of segments. Experimental evaluations are performed on monophonic flute recordings from Grey Larsen's CD, which was manually annotated by an experienced flute player. The onset and single- and multi-note ornament detection performance is presented in terms of the precision, recall and F-measure.

1. INTRODUCTION

Within Irish traditional music, ornaments are used extensively by all melody instruments. They are central to the style of the music, adding to its liveliness and expression. Amongst traditional players, the melody is merely a framework [3, 4] – dynamics, ornamentation and context will be added in real time. This is often different from classical music where a standard notation for each piece of music usually includes ornaments as written by the composer.

Ornaments are notes of a very short duration. They can be categorised into single-note and multi-note ornaments. Single-note ornaments are amongst the most common in Irish traditional music. Multi-note ornaments consist of a specific sequence of note and single-note ornaments.

Methods for ornament detection are typically based on detection of note onsets. Note onsets may be categorised as hard or soft. A hard onset, typical in percussive instruments, is characterised by a sudden change in energy. A soft onset shows a more gradual change in energy and it occurs in wind instruments, like flute. A variety of methods have been proposed for the detection of note onsets in music recordings, e.g., [1, 8, 11, 13, 17]. The methods typically exploit the change in the energy of the signal, which may be estimated in temporal or spectral domain. The use of phase has also been investigated, e.g., [1, 11], and combined with the fundamental frequency in [11]. It has been reported that reliable note onset detection for non-percussive instruments is more difficult to obtain due to the soft nature of the onsets [11].

An automated detection of ornaments is a challenging problem. This is because ornaments are of very short durations, which may cause them being easily omitted or falsely detected. Unlike note onset detection, this research area has received relatively little attention. An automatic location of ornaments for flute recordings based on MPEG-7 features was investigated in [5]. Transcription of baroque ornaments in two piano recordings by analysing rhythmic groupings and expressive timing was studied in [2]. This work used onset values from manually edited time-tagged audio. Several works employed spectral-domain energy-based onset detection, e.g., [9, 10, 16]. The work in [16] analysed ornamentation from Bassoon recordings. The work of a group from Dublin Institute of Technology, summarised in [9], is the only study on the detection of ornaments in Irish traditional flute music. This provided only some initial results and on a considerably smaller dataset.

In this paper, we extend our recent work presented in [14] and investigate automatic detection of single- and multi-note ornaments in flute playing. The presented ornament detection system is based on first detecting onsets and then exploiting knowledge of musical ornamentation. We explore the use of several different methods for onset detection and customisation of their parameters to detection of soft onsets of notes which may be also of very short duration. The detected onsets provide segmentation of the signal, where a segment is defined by the adjacent detected onsets. This segmentation, together with the musical knowledge of ornamentation is then used for the detection of single- and multi-note ornaments. Experimental evalua-

tions are performed using recordings of Irish traditional tunes played by flute from Grey Larsen's CD [15]. Results of ornament detection are presented in terms of the precision, recall and F-measure. The average F-measure performance for single- and multi-note ornaments is over 76% and 67%, respectively.

2. SINGLE- AND MULTI-NOTE ORNAMENTS IN IRISH TRADITIONAL FLUTE PLAYING

Ornaments are used as embellishments in Irish traditional music [15]. They are notes of a very short duration, created through the use of special fingered articulations.

Single-note ornaments, namely 'cut' and 'strike', are pitch articulations. The 'cut' involves quickly lifting and replacing a finger from a tonehole, and corresponds to a higher note than the ornamented note. The 'strike' is performed by momentarily closing an open hole, and corresponds to a lower note than the ornamented note.

Multi-note ornaments are successive use of single-note ornaments. To simplify the description, we refer to the ornamented note as the base note throughout the rest of this paper. The 'roll' consists of the base note, a 'cut', base note, a 'strike' and then returning to the base note. A shorter version of the roll, referred to as short-roll, omits the starting base note. The 'crann' consists of the base note that is cut three times in rapid succession and then returning to the base note. The short-crann omits the starting base note. The 'shake' commences with a 'cut', followed by a base note and a second 'cut' and then returning to the base note.

A schematic visualisation of the single- and multi-note ornaments is given in Figure 1. In the multi-note ornaments figure, the proportions of the length of the individual parts aim to approximately indicate the typical duration proportions. For instance, in theory, a roll would be split equally into three parts by the cut and the strike but in reality different players will time this differently according to the 'swing' of the tune, their muscle control and a host of other attributes that make up their personal style.

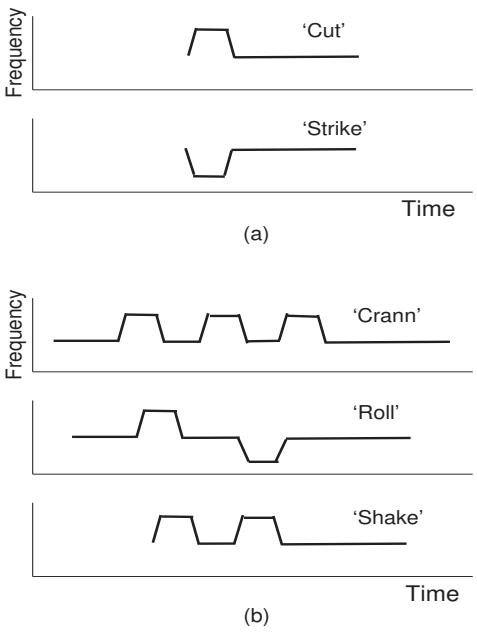


Figure 1. A schematic representation of single-note (a) and multi-note (b) ornaments.

3. AUTOMATIC DETECTION OF ORNAMENTS

This section presents the developed automatic ornament detection system. We first give a brief description of the onset detection methods we employed and then describe how the detected onsets are used for the detection of single- and multi-note ornaments.

3.1 Methods for detection of onsets

Here we briefly describe three onset detection methods we employed. Two of the methods exploit the change of the signal amplitude over time, with processing performed in the temporal and spectral domain [1, 8]. The third method is based on the fundamental frequency [6, 11]. Each of the method requires several parameters to be set and their values are explored during experimental evaluations and presented later in Section 4.3. The implementation of the temporal domain energy-based method used in parts some functions from the MIRtoolbox.

3.1.1 Signal energy: spectral domain

This method, also sometimes referred to as spectral-flux method, performs onset detection in the spectral domain. The signal is segmented into overlapping frames. Each signal frame is multiplied by Hamming window. The windowed frames are then zero padded and the Fourier transform is applied to provide the short-term Fourier spectrum. For each frequency bin, the differences between the short-term magnitude spectra of successive signal frames is computed. This is then half-wave rectified and the L_2 norm is calculated to provide the value of the detection function at the current frame. The peaks of the detection function, whose amplitude is above a threshold are used as the detected onsets. We explored the use of a fixed threshold value as well as computing the value adaptively based on the median of the detection function values around the current frame. Finally, if two consecutive peaks are found within a given time distance, only the first peak is used.

3.1.2 Signal energy: temporal domain

Another method we employed performs the detection in temporal domain. The signal is passed through a bank of fourteen band pass filters, each tuned to a specific note on the flute in the range from $D4$ to $B5$. The filters have non-overlapping bands, with the lower and the upper frequency being half way between the adjacent note frequencies. These fourteen notes are readily playable on an unkeyed concert flute. The signal in each band is full-wave rectified and then smoothed, resulting in amplitude envelope. The time derivative of the amplitude envelope is calculated in each band and this is smoothed by convolving

it with a half-Hanning window. We explored several ways of making decision about detected onsets. The information from all bands can be combined by summing together their smoothed derivative signals. Alternatively, a single band can be chosen as a representative at each time based on assessment of amplitudes of peaks around that time across all bands. Onsets are obtained by comparing the values of peaks to a threshold, which may be fixed or adaptive over time.

3.1.3 Fundamental frequency

In addition to methods exploiting the signal envelope, we also explore the use of the fundamental frequency (F_0). This has been reported to be beneficial for soft onset detection in [11]. Among a large variety of existing F_0 estimation algorithms, we employed the YIN algorithm [7] in this work. The F_0 estimation may result in so called doubling / halving errors. To help dealing with these errors, the F_0 estimates are postprocessed using a median filter. The length of this filter needs to be set sensitively – a longer filter may be preferable to deal with the F_0 estimation errors but this may also cause filtering out ornaments, which are characterised by their short duration.

The detection function at the frame time n, denoted as R_n, is based on calculating the change of F_0 over time. This can be performed by taking the difference between the F_0 estimate at the frame $(n+\Theta)$ and $(n-\Theta)$. The onset is detected as the first frame for which $abs(R_n) > \alpha_{F_0}$, where the value of the threshold α_{F_0} relates to the difference between frequencies of the closest possible notes.

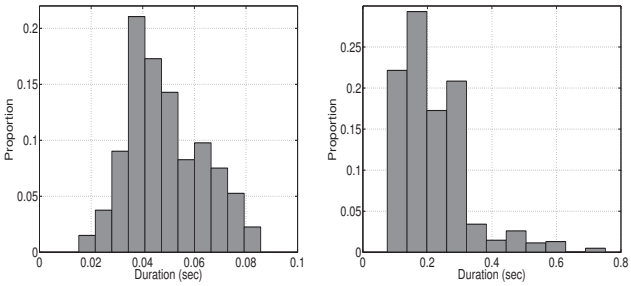
3.2 Ornament detection

The detected onsets, as obtained using the methods described in Section 3.1, provide a segmentation of the signal, where each segment is formed based on the adjacent detected onsets.

We characterise each detected segment by some features, specifically, here we use the duration of the segment and its segmental fundamental frequency. For a given segment, its duration, denoted by D^{seg}, is obtained based on the detected onsets and its fundamental frequency, denoted by F_0^{seg}, is calculated as the median value of the F_0s corresponding to all signal frames assigned to that segment. Finally, these segment features are used to determine whether the detected segment corresponds to a note or a single-note ornament and whether the sequence of segments corresponds to a multi-note ornament, and if single- or multi-note ornament is detected, then to determine its type.

3.2.1 Single-note ornament detection

As single-note ornaments are expected to be of a shorter duration than notes, we examined whether the duration of the detected segments can be used to discriminate these ornaments from notes. We conducted statistical analysis of the duration of notes and single-note ornaments in our recordings. This was performed using the manual onset annotations. The obtained distributions of the durations are depicted in Figure 2 – these indicate that the duration can indeed provide a good discrimination between notes and ornaments. Based on these results, we consider that a segment is classified as a single-note ornament when its duration is below 90 ms, otherwise it is classified as a note.



(a) (b)

Figure 2. The distribution of the duration of single-note ornaments (a) and notes (b) obtained using the development set.

The decision whether the detected single-note ornament is a 'cut' or 'strike' can be made based on comparing the values of the F_0^{seg} of the current and the following segment. This reflects the musical knowledge of ornamentation. If F_0^{seg} of the segment detected as ornament is higher than F_0^{seg} of the following segment, the ornament is classified as 'cut' and as 'strike' otherwise.

3.2.2 Multi-note ornament detection

The detection of multi-note ornaments, namely 'crann', 'roll' and 'shake', is based on analysing the features of a sequence of detected consecutive segments. We used a set of rules to determine whether the sequence corresponds to one of the multi-note ornament types or not. These rules reflect the definition of the multi-note ornaments as presented in Section 2 and for each ornament type are described below. Let us consider that r denotes the index of the first segment in the sequence of detected segments we are currently analysing. Let us denote by $\Delta F_0^{seg}(j, j+2)$ the difference between the F_0^{seg} for the segment $(r+j)$ and F_0^{seg} for the segment $(r+j+2)$, where j is an index to be set.

'*Crann*' is detected if the following is fulfilled: i) the sequence of F_0^{seg} follows the pattern '*BHBHBHB*', where '*B*' stands for a base note and '*H*' for a note higher than the base note; ii) the segmental F_0^{seg} is similar for segments corresponding to the base note, i.e., the $\Delta F_0^{seg}(j, j+2)$ is within the given tolerance range β_{F_0} when j is individually set to 0, 2, and 4; and iii) the segment duration D^{seg} is below β_D for segments given by setting j from 1 to 5 and is above β_D for j set to 0 and 6. The '*Short-Crann*' is using the same rules but taking into account that the starting base note is omitted.

'*Roll*' is detected if the following is fulfilled: i) the sequence of F_0^{seg} follows the pattern '*BHBLB*', where '*L*' stands for a note lower than the base note; ii) the value of $\Delta F_0^{seg}(j, j+2)$ is within the tolerance range β_{F_0} for j set

to 0 and 2; and iii) the segment duration D^{seg} is above β_D for j being 0 and 2 and is below β_D when j is 1 and 3. Again, the '*Short-Roll*' is using the same rules but taking into account that the starting base note is omitted.

'*Shake*' is detected if the following is fulfilled: i) the sequence of notes follows the pattern '*HBHB*'; ii) the value of $\Delta F_0^{seg}(j, j+2)$ is within a given tolerance range β_{F_0} for j set to 1; and iii) the segment duration D^{seg} is below β_D when j is 1 and 2, and is above β_D when j is 3.

The parameters β_{F_0} and β_D were set to 20 Hz (except for 'crann' when 30 Hz was used) and 90 ms, respectively.

4. EXPERIMENTAL RESULTS

4.1 Data description

Evaluations are performed using recordings of Irish traditional tunes and training exercises played by flute from Grey Larsen's CD which accompanied his book "Essential Guide to Irish Flute and Tin Whistle" [15]. The tunes are between 20 sec and 1 min 11 sec long. All recordings are monophonic and are sampled at 44.1 kHz sampling frequency. Manual annotation of the recordings to indicate the times of onsets and offsets and the identity of notes and ornaments was performed by the third author of this paper, who is a highly experienced musician with over 10 years of flute playing. The manual annotation is used as the ground truth in evaluations. The data was split into separate development and evaluation sets. The development set, consisting of 6 tunes (namely 'Study5', 'Study6', 'Study17', 'Lady on the Island', 'The Lonesome Jig', 'The Drunken Landlady'), was used for finding the best parameter values of onset detection methods. The evaluation set, consisting of 13 tunes, was used to obtain the presented results. The list of the tunes from the evaluation set, with the number of notes and ornaments, is given in Table 1. In total, this set contains 3025 onsets, including notes and ornaments. Out of these there are 301 single-note ornaments, consisting of 257 cuts and 44 strikes, and 152 multi-note ornaments, consisting of 117 rolls (including short-rolls), 19 cranns (including short-cranns), and 16 shakes.

4.2 Evaluation measures

Performance of the onset and ornament detection is evaluated in terms of the precision (P), recall (R) and F-measure. The definition of these measures is the same as used in MIREX onset detection evaluations, specifically,

$$P = \frac{N_{tp}}{N_{tp} + N_{fp}}, R = \frac{N_{tp}}{N_{tp} + N_{fn}}, F = \frac{2PR}{P + R}$$

where N_{tp} is the number of correctly detected onsets / ornaments and N_{fp} and N_{fn} is the number of inserted and deleted onsets / ornaments, respectively. The onset detection is considered as correct when it is within ± 50 ms around the onset annotation.

The single-note and multi-note ornaments are considered to be detected correctly when the onsets, corresponding to the start and to the end of the ornament are within ± 50 ms and ± 100 ms range, respectively.

Tune Title	Number of		Time
	Notes	Ornaments (C-S-Ro-Cr-Sh)	(sec.)
Study 11	76	20–0–0–0–0	26
Study 22	127	0–28–0–0–0	47
Maids of Ardagh	98	23–0–5–0–0	32
Hardiman the ..	112	12–0–7–1–0	28
The Whinny Hills ..	117	15–1–5–2–4	30
The Frost is All ..	151	27–2–12–0–0	41
The Humours of ..	289	59–7–12–14–0	82
The Rose in the ..	152	22–2–11–0–0	39
Scotsman over ..	153	18–0–9–2–0	38
A Fig for a Kiss	105	17–3–6–0–2	28
Roaring Mary	176	15–1–21–0–3	44
The Mountain Road	105	8–0–6–0–3	25
The Shaskeen	181	21–0–23–0–4	42

Table 1. The list of tunes contained in the evaluation set, with the number of onsets and ornaments and duration of each tune. The notation 'C', 'S', 'Ro', 'Cr' and 'Sh' stands for 'cut', 'strike', 'roll', 'crann' and 'shake', respectively.

4.3 Results of onset detection

We have performed extensive evaluations on the development set with different parameter values for each of the onset detection method. The best values of parameters for each of the method are given in Table 2. The achieved performance on the evaluation set using these parameters for each method is presented in Table 3. Note that these results include the onsets corresponding to both notes and ornaments. Performance difference of less than 1% was observed when the parameters were tuned specifically for the evaluation set. It can be seen that all methods provide good onset detection performance, with the F_0-based method being slightly better than the energy-based methods. A method based on F_0 was shown to perform best for wind instruments also in [11], where it was also shown that its combination with other methods provided only slight improvement at similar P and R values. As such, in the following, we use only the F_0-based method for evaluating the ornament detection performance. An example of a signal extract from one of the tune and the corresponding F_0 estimate and the detection function, with indicated true label and detected onsets, are depicted in Figure 3.

4.4 Results of single-note ornament detection

The results of single-note ornament detection are presented in Table 4 separately for 'cut' and 'strike'. The achieved detection performance is significantly higher than that presented in previous flute studies using similar data [9]. The performance for 'cut' is close to the overall onset detection performance as presented in Table 3. The performance for 'strike' is considerably lower than for 'cut'. This has also been observed in previous research and may be due to the nature the 'strike' is created. There was 5 substitutions of

Onset detection method with best values of the parameters
sig-energy (spectral):
– frame length of 1024 samples (23.2 ms)
– frame shift of 896 samples (20.3 ms)
– threshold set as fixed at 2% of the maximum of the normalised detection function
– minimum distance between peaks set to 10 ms
sig-energy (temporal):
– half-Hanning window of 35 ms
– threshold set as fixed at 15% of the maximum of the normalised detection function
– minimum distance between peaks set to 20 ms
F_0:
– frame length of 1024 samples (23.2 ms)
– frame shift of 128 samples (2.9 ms)
– median filter of length 9 frames
– parameter Θ set to 6 frames (17 ms)
– parameter α_{F_0} set to 10 Hz

Table 2. Parameters of each onset detection method and their best values obtained based on the development set.

Algorithm	Evaluation performance (%)		
	Precision	Recall	F-measure
sig-energy (spectral)	94.9	85.0	89.7
sig-energy (temporal)	87.9	88.6	88.3
F_0	89.1	92.9	91.0

Table 3. Results of onset detection obtained by each of the employed method.

cut for strike and 1 substitution of strike for cut. These errors were contributed by slight inaccuracies in onset detection and F_0 misestimation.

	Single-note Ornament Detection		
	Precision (%)	Recall (%)	F-measure (%)
Cut	88.4	86.4	87.4
Strike	63.8	68.2	65.9

Table 4. Results of single-note ornament detection obtained by employing the F_0-based onset detection method.

4.5 Results of multi-note ornament detection

Experiments for multi-note ornament detection were performed by analysing all the possible sequence patterns resulting from the detected segments – this consisted here of 3020 sequence pattern candidates. The results of multi-note ornament detection are presented in Table 5 separately for 'roll', 'crann' and 'shake'. These results include also the short versions for 'roll' and 'crann'. It can be seen that

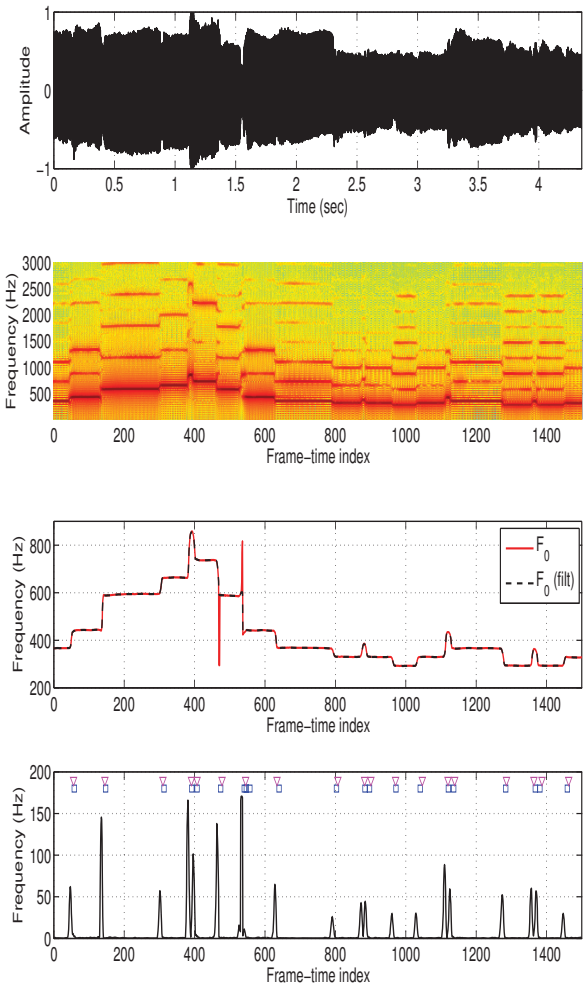


Figure 3. An extract from the tune 'The Lonesome Jig', depicting (from top to bottom) the waveform, spectrogram, F_0 estimation (unfiltered (red) and filtered (dashed black)) and the detection function with indicated detected onsets (blue □) and true label (magenta ▽).

the performance for 'shake' is considerably lower than that for 'roll' and 'crann'. This is due to the short sequence pattern of 'shake', consisting of only 4 parts, which makes it more likely to be accidentally match with other note sequence. We have also analysed the performance separately for the short and normal versions of the 'roll' and 'crann' ornaments. This showed that the F-measure performance for 'roll' was approximately 17% better than for 'short-roll'. This trend was not observed for 'short-crann', which may be due its longer note sequence.

5. CONCLUSION AND FUTURE WORK

In this paper, we presented work on detection of single- and multi-note ornaments in Irish traditional flute music. We employed three different methods for onset detection and customised their parameter values to detecting soft onsets of possibly very short notes. The method based on the fundamental frequency (F_0) achieved around 91% onset detection performance in terms of the F-measure and

	Multi-note Ornament Detection		
	Precision (%)	Recall (%)	F-measure (%)
Roll	87.5	67.0	75.9
Crann	86.7	68.4	76.5
Shake	50.0	50.0	50.0

Table 5. Results of multi-note ornament detection obtained by employing the F_0-based onset detection method.

outperformed slightly the other two energy-based methods. The F_0-based method was then used for evaluating the ornament detection performance. The discrimination between notes and single-note ornaments was based on the duration of segments defined by the adjacent detected onsets. The F_0 information of the current and the following segment was used to distinguish between 'cut' and 'strike' single-note ornaments. The achieved F-measure performance for 'cut' was over 87%, while for 'strike' over 65%. The multi-note ornament detection system was based on analysing the properties of a sequence of detected segments. This included the sequential pattern of segmental F_0's, the duration of each segment, and the relationship of the segmental F_0's among the segments. The average F-measure performance over all types of multi-note ornaments was over 67%.

There are several points we are currently considering to extend this work. First, we plan to analyse the errors made by each of the onset detection methods and accordingly explore whether their combination could lead to detection performance improvements. This would also include exploration of the use of other onset detection methods, including other F_0 estimation algorithms and possible incorporation of the sinusoidal detection method we presented in [12]. Second, we will explore a compensation for variations in tempo across the recordings. Finally, we plan to employ probabilistic rules for detection of multi-note ornaments which should allow for better handling of the variations due to player's style.

Acknowledgement

This work was supported by a project under the 'Transforming Musicology' programme funded by Arts and Humanities Research Council (UK).

6. REFERENCES

[1] J. P. Bello, L. Daudet, S. Abdallah, Ch. Duxbury, M. Davies, and M. B. Sandler. A tutorial on onset detection in music signals. *IEEE Trans. on Speech and Audio Processing*, pages 1–13, 2005.

[2] G. Boenn. Automated quantisation and transcription of musical ornaments from audio recordings. In *Proc. of the Int. Computer Music Conf. (ICMC)*, pages 236–239, Copenhagen, Denmark, Aug. 2007.

[3] B. Breathnach. *Folk music and dances of Ireland*. Ossian, London, 1996.

[4] C. Carson. *Last night's fun: in and out of time with Irish music*. North Point Press, New York, 1997.

[5] M. Casey and T. Crawford. Automatic location and measurement of ornaments in audio recording. In *Proc. of the 5th Int. Conf. on Music Information Retrieval (ISMIR)*, pages 311–317, Barcelona, Spain, 2004.

[6] N. Collins. Using a pitch detector for onset detection. In *Proc. of the 5th Int. Conf. on Music Information Retrieval (ISMIR)*, pages 100–106, Spain, 2005.

[7] A. de Cheveigne and H. Kawahara. Yin, a fundamental frequency estimator for speech and music. *Journal of the Acoustical Society of America*, 111(4):1917–1930, April 2002.

[8] S. Dixon. Onset detection revisited. In *Proc. of the 9th Int. Conf. on Digital Audio Effects (DAFx)*, pages 133–137, Montreal, Canada, Sep. 2006.

[9] M. Gainza and E. Coyle. Automating ornamentation transcription. In *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Honolulu, Hawaii, 2007.

[10] M. Gainza, E. Coyle, and B. Lawlor. Single-note ornaments transcription for the Irish tin whistle based on onset detection. In *Proc. of the Digital Audio Effects (DAFX)*, Naples, Italy, 2004.

[11] A. Holzapfel, Y. Stylianou, A. C. Gedik, and B. Bozkurt. Three dimensions of pitched instrument onset detection. *IEEE Trans. on Audio, Speech, and Language Processing*, 18(6):1517–1527, Aug. 2010.

[12] P. Jančovič and M. Köküer. Detection of sinusoidal signals in noise by probabilistic modelling of the spectral magnitude shape and phase continuity. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 517–520, Prague, Czech Republic, May 2011.

[13] A. Klapuri. Sound onset detection by applying psychoacoustic knowledge. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 3089–3092, March 1999.

[14] M. Köküer, I. Ali-MacLachlan, P. Jančovič, and C. Athwal. Automated detection of single-note ornaments in Irish traditional flute playing. In *Int. Workshop on Folk Music Analysis*, Istanbul, Turkey, June 2014.

[15] G. Larsen. *The Essential Guide to Irish Flute and Tin Whistle*. Mel Bay Publications, Pacific, Missouri, USA, 2003.

[16] M. Puiggros, E. Gómez, R. Ramirez, X. Serra, and R. Bresin. Automatic characterization of ornamentation from bassoon recordings for expressive synthesis. In *9th Int. Conf. on Music Perception and Cognition*, Bologna, Italy, 2006.

[17] E. D. Scheirer. Tempo and beat analysis of acoustic musical signals. *Journal of the Acoustical Society of America*, 103(1):588–601, 1998.

THE KIKI-BOUBA CHALLENGE: ALGORITHMIC COMPOSITION FOR CONTENT-BASED MIR RESEARCH & DEVELOPMENT

Bob L. Sturm
Audio Analysis Lab, Aalborg University, Denmark
bst@create.aau.dk

Nick Collins
Dept. Music, Durham University, UK
nick.collins@durham.ac.uk

ABSTRACT

We propose the "Kiki-Bouba Challenge" (KBC) for the research and development of content-based music information retrieval (MIR) systems. This challenge is unencumbered by several problems typically encountered in MIR research: insufficient data, restrictive copyrights, imperfect ground truth, a lack of specific criteria for classes (e.g., genre), a lack of explicit problem definition, and irreproducibility. KBC provides a limitless amount of free data, a perfect ground truth, and well-specifiable and meaningful characteristics defining each class. These ideal conditions are made possible by open source algorithmic composition — a hitherto under-exploited resource for MIR.

1. INTRODUCTION

Before attempting to solve a complex problem, one should approach it by first demonstrably solving simpler, well-defined, and more restricted forms, and *only then* increase the complexity. However, there are key problems of research in content-based music information retrieval (MIR) [8] where this has yet to be done. For example, much of the enormous amount of research that attempts to address the problem of music genre recognition (MGR) [26] has started with genre in the "real world" [30]. The same is seen for research in music mood recognition [28, 29, 37], and music autotagging [6]. On top of this, the problem of describing music using genre, mood, or tags in general, has rarely, if ever, been explicitly defined [32].

In lieu of an explicit definition of the problem, the most common approach in much of this research is to implicitly define it via datasets of real music paired with "ground truth." The problem then becomes reproducing as much of the "ground truth" as possible by pairing feature extraction and machine learning algorithms, and comparing the resulting numbers to those of other systems (including humans). Thousands of numerical results and publications have so far been produced, but it now appears as if most of it has tenuous relevance for *content-based* MIR [3, 27, 30, 31, 34]. The crux of the argument is that the lack of scientific validity in evaluation in much of this work [3, 27, 30] has led to the development of many MIR systems that appear as if they are "listening" to the music when they are actually just exploiting confounded characteristics in a test dataset [31]. Thus, in order to develop MIR systems that address the goal of "making music, or information about music, easier to find" [8] in the real-world, there is a need to first demonstrably solve simple, well-defined and restricted problems.

Toward this end, this paper presents the "Kiki-Bouba Challenge" (KBC), which is essentially a simplification of the problem of MGR. On a higher level, we propose KBC to refocus the goals in content-based MIR. We devise KBC such that solving it is unencumbered by six significant problems facing content-based MIR research and development: 1) the lack of formal definition of retrieving information in recorded music; 2) the large amount of data necessary to ensure representativeness and generalization for machine learning; 3) the problem of obtaining "ground truth"; 4) the stifling affect of intellectual property (e.g., music copyright) on collecting and sharing recorded music; 5) the lack of validity of standard evaluation approaches of systems; and 6) a lack of reproducible research. KBC employs algorithmic composition to generate a limitless amount of music from two categories, named *Kiki* and *Bouba*. Music from each category are thereby free from copyright, are based in well-defined programs, and have a perfect ground truth. Solving KBC represents a veritable contribution of content-based MIR research and development, and promises avenues for solving parallel problems in less restricted and real-world domains.

Instead of being merely the reproduction of a "ground truth" of some dataset, the MIR "flagship application" of MGR [4] — and that which KBC simplifies — has as its principal goals the *imitation of the human ability to organize, recognize, distinguish between, and imitate genres used by music* [28]. To "imitate the human ability" is not necessarily to replicate the physiological processes humans use to hear, process and describe a piece of music, but merely to describe as humans do a piece of music *according to its content*, e.g., using such musically meaningful attributes as rhythm, instrumentation, harmonic progression, or formal structure. Solving the problem of MGR means creating an artificial system that can work with music like humans, but unencumbered by human limitations.

The concept of genre [12, 13, 16] is notoriously difficult to define such that it can be addressed by algorithms [23]. Researchers building MGR systems have by and large posed

the problem, implicitly or explicitly, from an Aristotelean viewpoint, i.e., "genre" is a categorization of music just as "species" is a categorization of living things, e.g., [5,33]. [1] The problem then is to automatically learn the characteristics that place a piece of music on one branch of a taxonomy, distinguish it from a piece of music on a different branch, and avoid contradiction in the process [7]. Researchers have combined signal processing and machine learning with datasets of real music recordings in hopes that the resulting system can discover Aristotelean criteria by which music can be categorized according to genre. The majority of the resulting work, however, documents how much "ground truth" an algorithm replicates in benchmark datasets [26], but rarely illuminates the criteria a system has learned and is using to categorize music [27]. The former quantity is meaningless when the latter is senseless.

In the next section, we discuss the use of algorithmic music composition for data generation. Then we present KBC in its most general form. We follow this with a concrete and specific realisation of KBC, available at the relevant webpage: `http://composerprogrammer.com/kikibouba.html`. We present an unacceptable solution to KBC, and discuss aspects of an acceptable solution. We conclude this paper with a discussion of KBC, and how it relates to content-based MIR in the "real world."

2. ALGORITHMIC MUSIC COMPOSITION FOR GENERATING DATA

Algorithmic composition [1,9,19,21,22,25,36] has a long history back to mainframe computer experiments in the mid 1950s, predating by a decade MIR's first explicit paper [17]. Ames and Domino [2] differentiate *empirical style modeling* (of historic musical styles) and *active style synthesis* (of novel musical style). In the practical work of this article we concentrate more on the latter, but there is a rich set of techniques for basing generation of music on models trained on existing musical data. Many musical models deployed to capture regularities in data sets are generative, in that a model trained from a corpus can generalise to production of new examples in that style [11].

Though anticipated by some authors, it is surprising how few studies in computer music have utilised algorithmic composition to create the ground truth. Although [24] present a four category taxonomy of algorithmic composition, they do not explicitly discuss the option of using algorithmic composition to produce data sets. The closest category is where "theories of a musical style are implemented as computer programs" [24], essentially empirical style modeling as above.

Sample CD data, especially meta-data on splices, have also rarely been used. But the advantage of algorithmic composition techniques are the sheer volume of data which can potentially be generated, and appropriately handled should be free of the copyright issues that plague databases of music recordings and hinder research access.

We believe that algorithmic generation of datasets within a framework of open source software has the following potential benefits to MIR and computer music analysis:

- Limitless data set generation, with perfect ground truth (the originating program is fully accessible, and can be devised to log all necessary elements of the ground truth during generation. Random seeds can be used to recover program runs exactly as necessary)
- A fully controlled musical working space, where all assumptions and representational decisions are clear
- Copyright free as long as license free samples or pure synthesis methods are utilised, under appropriate software licensing
- Established data sets can be distributed free of the originating software once accepted by the community, though their origins remain open to investigation by any interested researcher

The greatest issue with dependence on algorithmic generation of music is the ecological validity of the music being generated. A skeptic may question the provenance of the music, especially with respect to the established cultural and economically proven quality of existing human driven recorded music production. Nonetheless, humans are intimately involved in devising algorithmic composition programs. We believe that there is place for expert judgement here, where experts in algorithmic composition can become involved in the process of MIR evaluation. The present paper serves as one humble example; but ultimately, a saving grace of any such position is that the generation code is fully available, and thus accessible to reproduction and evaluation by others.

3. THE KIKI-BOUBA CHALLENGE

We now present KBC in its most general form: *develop a system that can organize, recognize, distinguish between, and imitate Aristotelean categories of "music."* We define these in the subsections below, after we specify the domain.

3.1 Domain

The music universe of KBC is populated by "music" belonging to either one of two categories, *Kiki* and *Bouba*. [2] In KBC, music from either category is algorithmically composed such that there is available a limitless number of recordings of music from both categories, and which are entirely unencumbered by copyrights. A music recording from this universe therefore embeds music from *Kiki* and not from *Bouba*, or vice versa, for several reasons that are neither ambiguous nor disputable, and which can be completely garnered from the music recording. The ground truth of a dataset of recordings of music from the music universe then is absolute. Note that a music recording need not be an audio recording, but can be a notated score, or other kind of representation. Now, given that this is ideal

[1] This of course belies the profound issues that biologists face in recognizing "speciation" events [10].

[2] Shapes named "Kiki" and "Bouba" (the two are spiky and rounded, respectively) were originally introduced in gestalt psychology to investigate cross-cultural associations of visual form and language [18, 20]. Our example realization of KBC involves two distinctive artificial musical "genres" meant to illustrate in sonic terms a similar opposition.

Attribute	*Kiki*	*Bouba*
Form	Alternating accelerando rises and crazy section ("freak out")	Steady chorale
Rhythm	Accelerando and complex "free" rhythm, fast	Limited set of rhythmic durations, slow
Pitch	Modulo octave tuning system	Recursive subdivision tuning system
Dynamics	Fade ins and outs during accelerando and close of "freak out" sections	Single dynamic
Voicing	All voices in most of the time	Arch form envelope of voice density, starting and ending with single voice
Timbre	Percussive sounds alongside fast attack and decay bright pitched sounds. Second rise has an additional siren sound.	Slow attack and decay sounds with initial portamento and vibrato, with an accompanying dull thud
Harmony	Accidental coincidences only, no overall precepts	System of tonality, with a harmonic sequence built from relatively few possible chords
Texture	More homophonic in accelerando, heterogenous with independent voices in "freak out" sections	Homophonic, homogenous
Expression	Ensemble timing loose on accelerando, independent during "freak out" sections	Details of vibrato, portamento and "nervousness" (chance of sounding on a given chord) differ for each voice in the texture
Space	Little or no reverb	Very reverberant

Table 1. Musical attributes of our realization of *Kiki* and *Bouba*.

for toolboxes of algorithms in an Aristotelean world, we pose the following tasks.

3.2 The discrimination task (unsupervised learning)

Given an unlabelled collection of music recordings from the music universe, build a system that determines there exist two categories in this music universe, *and* high-level (content) criteria that discriminate them. In machine learning, this can be seen as unsupervised learning, but ensuring discrimination is caused by content and not criteria that are irrelevant to the task.

3.3 The identification task (supervised learning)

Given a labelled collection of music recordings from the music universe, build a system that can learn to identify, using high-level (content) criteria, recordings of music (either from this music universe or from others) as being from *Kiki*, *Bouba*, or from neither. In machine learning, this can be seen as supervised learning, but ensuring identification is caused by content and not criteria that are irrelevant to the task.

3.4 The recognition task (retrieval)

Given a labelled collection of music recordings from this music universe, build a system that can recognize content in *real world music recordings* as being similar to contents in music from *Kiki*, *Bouba*, both, or neither. In information retrieval, this can be seen as relevance ranking.

3.5 The composition task (generation)

Given a labelled collection of music recordings from this music universe, build a system that composes music having content similar to music from *Kiki*, and/or music from *Bouba*. The rules that the system uses to create the music must themselves be meaningful. For example, a music analyst would find the program that generates the music to provide a high-level breakdown of the characteristics of a category. In one sense, this challenge is a necessary precursor to those above, in that a human composer must design the ground truth of the music universe. The production of a dataset of music recordings with algorithmic composition necessitates creation in real musical terms. The machine challenge here is to backwards engineer, or to learn in short, the compositional ability to work in the pre-established music universe. However, backwards engineering the compositional mechanisms of such a system, as an expert human musician can potentially do when encountering a musical style unfamiliar to them, is itself an important challenge of high-level musical understanding.

4. AN EXAMPLE REALIZATION OF KBC

We now present an example realization of KBC. We specify *Kiki* and *Bouba* via computer programs for algorithmic composition, which we use to create unlimited recordings of music from *Kiki* and *Bouba*, each varying subtly in the fine details (we discuss the practical range of this variation further below). Our computer program is written in the SuperCollider audio programming language [35], with SuperCollider used here in non-realtime mode for fast synthesis of music recordings (which in this case are monophonic digital audio files). We measure the speed of generation of music recordings to be around 60×real-time, so that one piece of around one minute can be created every second by our code. With this we easily created a multi-gigabyte dataset of ten hours, and could very easily create far more.

As *Kiki* and *Bouba* are designed here by humans, they are not independent of "real" music, even though they are fully specified via open source code. [3] Table 3 outlines properties of music from *Kiki* and *Bouba* with respect to some high and low level musical properties. This conveys a sense of why *Kiki* and *Bouba* are well-differentiated in musically meaningful ways. Figure 1 further attempts to illustrate the formal structure of the two styles, again as a demonstration of their distinctiveness. Although the musical description is not as simple as the visual manifestation of the original shapes of "kiki" and "bouba" [18,20], it was designed to avoid too much overlap of musical characteristics. Each output piece is around 40-60 seconds, since

[3] We make available this source code, as well as a few representative sound examples at the accompanying webpage: `http://composerprogrammer.com/kikibouba.html`.

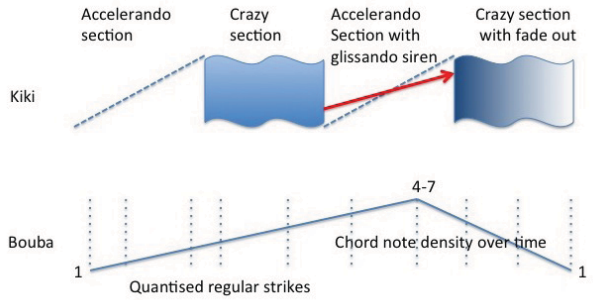


Figure 1. Comparative musical forms of our realization of music from *Kiki* and *Bouba* (labeled).

the actual length of sections is itself generative. It is beyond the scope of this article to discuss every detail of the code and the variability of output allowed, but this gives some idea. To anthropomorphise and allow a little literary conceit, our realization envisages music from *Kiki* to be ecstatic, chaotic and ritualistic, characterised by alternating build-ups (accelerando rises) and cathartic "freak-outs." Our realization envisages music from *Bouba* as an abstract choral funeral march, steady and affected.

4.1 An unacceptable solution

A typical approach to attempt to address an identification task is by computing a variety of low-level and short-time features from music recordings, modelling collections of these by probability distributions (bags of frames), and specifying criteria for classification, such as maximum likelihood. To this end, we use supervised learning to build a single nearest neighbor classifier trained with features computed from a dataset consisting of 250 recordings of music from *Kiki* and 250 from *Bouba*. As features, we first compute the number of zero crossings for 46.3 ms Hann-windowed audio frames, overlapped 50% across the entire recording. We then compute the mean and variance of the number of zero crossings from texture windows of 129 consecutive frames. Finally, we normalize the feature dimensions in the training dataset observations, and use the same normalization parameters to transform input observations. Figure 2 shows a scatter plot of these training dataset observations. To classify an input music recording as being of music from *Kiki* or *Bouba*, we use majority vote from the nearest neighbor classification of the first 10 consecutive texture windows.

We test the system using a stratified test dataset of 500 music recordings from *Kiki* or *Bouba*. For each input, we compare the system output to the ground truth. Our system produces a classification error of 0.00! It has thus successfully labeled all observations in the test dataset with the correct answer. However, this system is not a solution to the identification task of KBC, let alone the three other KBC tasks, *simply because it is not using high-level criteria (content).* Of course, the statistics of low-level zero crossings across short-time frames has *something* to do with content [15], but this relationship is quite far removed and ambiguous. In other words, people listen to and describe music in terms related to key, tempo and timbre, but not zero crossings. Statistics of zero crossings are

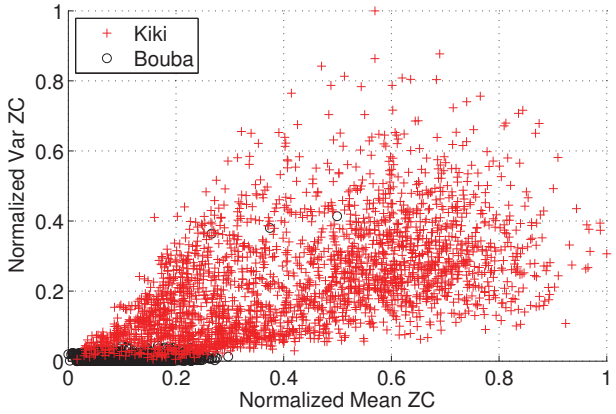


Figure 2. Scatter plot of features extracted from recordings of music from *Kiki* and *Bouba*.

not meaningful musical information for solving any task of KBC. That this feature contributes to the perfect figure of merit of this system, it does not illuminate what makes music *Kiki*, and what makes music *Bouba*.

4.2 An acceptable solution

As of the current time, we have yet to find any acceptable solution to our realization of KBC, or any of its tasks — which motivates this challenge. (Furthermore, as discussed below, the goal of KBC is not "a solution" but "solving.") We can, however, describe aspects of solutions acceptable for our specific realization of KBC. An acceptable solution to the discrimination task determines that in a set of music recordings from the music universe, there exist two different kinds of music, which are discriminable by high-level content, some of which are listed in Table 3, and shown in Fig. 1. An acceptable solution to the identification task determines for any given music recording whether its high-level contents are or are not consistent with all the musical attributes of *Kiki* or *Bouba*. An acceptable solution to the recognition task might recognize as *Bouba* characteristics the slow plodding rhythm, wailing timbre, and homophonic texture of some jazz funeral music. It might recognize as *Kiki* characteristics the glissando siren of some rave music, or the complex, unpredictable and ametrical rhythm of some free improvisation. It would recognize as not characteristic of either *Kiki* or *Bouba* the form of 12-bar blues. Finally, an acceptable solution to the composition task generates music that mimics particular characteristics of music from *Kiki* and *Bouba*.

5. DISCUSSION

In essence, KBC is a general exercise, of which we have provided one realization. KBC simplifies MGR — and music description in general — to the degree that many problems typically encountered in MIR research are not an issue, i.e., lack of data, copyright restrictions, cost and inaccuracy of ground truth, poor problem definition, and evaluations that lack validity with respect to meaningful musical understanding by machine. While most research in MGR searches for an Aristotelean categorization of real music (or the reverse engineering of the categorization used to create benchmark music datasets like GTZAN [30, 33]),

it sustains most of the complexity inherent to the problem of MGR. KBC simplifies it to be Aristotelean and well-defined. Essentially, KBC defines categories of music as well-specified and open-source programs, which comports with an Aristotelean conception of music genre. This allows us to benefit from algorithmic composition since we can generate from these programs any quantity of data, free of copyright, and with a perfect ground truth and specified classification criteria.

It can be speculated that KBC is too much of a simplification of MGR, that defining music using programs has little "ecological validity," and thus that a solution to KBC will be of little use for music in the "real world." To the first claim, the tasks of KBC are much more complex than reproducing ground truth labels of datasets by any means — the implicit goal of the majority of work addressing MGR [27, 30] — *because solving the tasks requires machine listening*, i.e., "intelligent, automated processing of music" [8]. To the second claim, our realizations of music from *Kiki* and *Bouba* actually originate in higher-level musical processes defined by humans trained and practiced in music composition. Fundamentally, "algorithmic music" and "non-algorithmic music" is a false dichotomy; but this is not to say all algorithms create equally "valid" music. One non-sensical realization of KBC is defining music from *Kiki* and *Bouba* as 50 ms long compositions, each consisting of a single sine, but with frequencies separated by 1 Hz between the two categories. To the final claim, we emphasize an important distinction between "a solution to KBC" and "solving KBC." We are not claiming that, e.g., a system that has learned to discriminate between music from *Kiki* and *Bouba* will be useful for discriminating between "real" music using any two "real" genres. The system (the actual finished product and black box [29]) will likely be useless. Rather, *solving KBC* is the goal because this requires developing a system that demonstrates a capacity to listen to acoustic signals in ways that consider high level (musical) characteristics.

If one desires more complexity than KBC offers, one can conceive of a music universe with more than two categories, and/or various mixings of "base" categories, e.g., giving rise to cross-genres *Bouki* and *Kiba* (the code at our link already has the capacity to generate these hybrid forms). However, we contend the best strategy is to first demonstrably solve the simplest problems before tackling ones of increased difficulty. If the components of a proposed MGR system result in a system that does not solve KBC, then why should they be expected to result in a system that can discriminate between, or identify, or recognize, or compose music using "real" genres of music from a limited amount of data having a ground truth output by a complex culturally negotiated system that cannot be as unambiguously specified as *Kiki* and *Bouba*?

6. CONCLUSION

Simply described, content-based MIR research and development aims to design and deploy artificial systems that are useful for retrieving, using or making music content. The enormous number of published works [6, 14, 26, 38], not to mention the participation during the past ten years of MIREX, [4] show many researchers are striving to build machine listening systems that imitate the human ability to listen to, search for, and describe music. Examples of such research include music genre recognition, music mood recognition, music retrieval by similarity, cover song identification, and various aspects of music analysis, such as rhythmic and harmonic analysis, melody extraction, and segmentation. These pursuits, however, are hindered by several serious problems: a limited amount of data, the sharing of which is restricted by copyright; the problematic nature of obtaining "ground truth," and explicitly defining its relationship to music content; and a lack of validity in the evaluation of content-based MIR systems with respect to the task they are supposedly addressing. We are thus left to ask: *Have the simplest problems been demonstrably solved yet?*

In this paper, we show how algorithmic music composition facilitates limitless amounts of data, with perfect ground truth and no restricting copyright, thus holding appreciable potential for MIR research and development We propose the "Kiki-Bouba Challenge" (KBC) as a simplification of the problem of MGR, and produce an example realization of it facilitated by algorithmic composition. We do not present an acceptable solution to our realization of KBC, but discuss aspects of such a solution. We also illustrate an unacceptable solution, which fails to reveal anything relating to musical meaning even though it still perfectly labels a test dataset. We emphasize, *the goal of KBC is not the system itself, but in solving the challenge.* Solving KBC changes the incentive of research and development in content-based MIR from one of developing systems obtaining high figures of merit by any means, to one of developing systems obtaining high figures of merit by *relevant* means.

7. ACKNOWLEDGMENTS

We wish to acknowledge the anonymous reviewers who helped significantly in the revision of this paper. The work of BLS was supported in part by Independent Postdoc Grant 11-105218 from Det Frie Forskningsråd.

8. REFERENCES

[1] C. Ames. Automated composition in retrospect: 1956-1986. *Leonardo*, 20(2):169–185, 1987.

[2] C. Ames and M. Domino. Cybernetic Composer: an overview. In M. Balaban, K. Ebcioğlu, and O. Laske, editors, *Understanding Music with AI: Perspectives on Music Cognition*, pages 186–205. The AAAI Press/MIT Press, Menlo Park, CA, 1992.

[3] J.-J. Aucouturier and E. Bigand. Seven problems that keep MIR from attracting the interest of cognition and neuroscience. *J. Intell. Info. Systems*, 41(3):483–497, 2013.

[4] http://www.music-ir.org/mirex

[4] J.-J. Aucouturier and E. Pampalk. Introduction – from genres to tags: A little epistemology of music information retrieval research. *J. New Music Research*, 37(2):87–92, 2008.

[5] J. G. A. Barbedo and A. Lopes. Automatic genre classification of musical signals. *EURASIP J. Adv. Sig. Process.*, 2007:064960, 2007.

[6] T. Bertin-Mahieux, D. Eck, and M. Mandel. Automatic tagging of audio: The state-of-the-art. In W. Wang, editor, *Machine Audition: Principles, Algorithms and Systems*. IGI Publishing, 2010.

[7] G. C. Bowker and S. L. Star. *Sorting things out: Classification and its consequences*. The MIT Press, 1999.

[8] M. Casey, R. Veltkamp, M. Goto, M. Leman, C. Rhodes, and M. Slaney. Content-based music information retrieval: Current directions and future challenges. *Proc. IEEE*, 96(4):668–696, Apr. 2008.

[9] D. Cope, editor. *Virtual Music : Computer Synthesis of Musical Style*. MIT Press, Cambridge, MA, 2001.

[10] D. C. Dennett. *Darwin's Dangerous Idea: Evolution and the Meanings of Life*. Simon and Schuster, 1996.

[11] S. Dubnov, G. Assayag, O. Lartillot, and G. Bejerano. Using machine learning methods for musical style modelling. *Computer*, pages 73–80, October 2003.

[12] F. Fabbri. A theory of musical genres: Two applications. In *Proc. Int. Conf. Popular Music Studies*, 1980.

[13] J. Frow. *Genre*. Routledge, New York, NY, USA, 2005.

[14] Z. Fu, G. Lu, K. M. Ting, and D. Zhang. A survey of audio-based music classification and annotation. *IEEE Trans. Multimedia*, 13(2):303–319, Apr. 2011.

[15] F. Gouyon, F. Pachet, and O. Delrue. On the use of zero-crossing rate for an application of classification of percussive sounds. In *Proc. DAFx*, 2000.

[16] Scott Johnson. The counterpoint of species. In J. Zorn, editor, *Arcana: Musicians on Music*, pages 18–58. Granary Books, Inc., New York, NY, 2000.

[17] M. Kassler. Towards musical information retrieval. *Perspectives of New Music*, 4(2):59–67, 1966.

[18] W. Köhler. *Gestalt Psychology*. Liveright, New York, 1929.

[19] J. McCormack and M. d'Inverno. *Computers and Creativity*. Springer-Verlag, Berlin Heidelberg, 2012.

[20] E. Milan, O. Iborra, M. J. de Cordoba, V. Juarez-Ramos, M. A. Rodríguez Artacho, and J. L. Rubio. The kiki-bouba effect a case of personification and ideaesthesia. *J. Consciousness Studies*, 20(1-2):1–2, 2013.

[21] E. R. Miranda, editor. *Readings in Music and Artificial Intelligence*. Harwood Academic Publishers, Amsterdam, 2000.

[22] G. Nierhaus. *Algorithmic Composition: Paradigms of Automated Music Generation*. Springer-Verlag/Wien, New York, NY, 2009.

[23] F. Pachet and D. Cazaly. A taxonomy of musical genres. In *Proc. Content-based Multimedia Information Access Conference*, Paris, France, Apr. 2000.

[24] M. Pearce, D. Meredith, and G. Wiggins. Motivations and methodologies for automation of the compositional process. *Musicae Scientiae*, 6(2):119–147, 2002.

[25] C. Roads. Research in music and artificial intelligence. *Computing Surveys*, 17(2), June 1985.

[26] B. L. Sturm. A survey of evaluation in music genre recognition. In *Proc. Adaptive Multimedia Retrieval*, Oct. 2012.

[27] B. L. Sturm. Classification accuracy is not enough: On the evaluation of music genre recognition systems. *J. Intell. Info. Systems*, 41(3):371–406, 2013.

[28] B. L. Sturm. Evaluating music emotion recognition: Lessons from music genre recognition? In *Proc. ICME*, 2013.

[29] B. L. Sturm. Making explicit the formalism underlying evaluation in music information retrieval research: A look at the MIREX automatic mood classification task. In *Post-proc. Computer Music Modeling and Research*, 2014.

[30] B. L. Sturm. The state of the art ten years after a state of the art: Future research in music information retrieval. *J. New Music Research*, 43(2):147–172, 2014.

[31] B. L. Sturm. A simple method to determine if a music information retrieval system is a "horse". *IEEE Trans. Multimedia*, 2014 (in press).

[32] B. L. Sturm, R. Bardeli, T. Langlois, and V. Emiya. Formalizing the problem of music description. In *Proc. ISMIR*, 2014.

[33] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Trans. Speech Audio Process.*, 10(5):293–302, July 2002.

[34] J. Urbano, M. Schedl, and X. Serra. Evaluation in music information retrieval. *J. Intell. Info. Systems*, 41(3):345–369, Dec. 2013.

[35] S. Wilson, D. Cottle, and N. Collins, editors. *The SuperCollider Book*. MIT Press, Cambridge, MA, 2011.

[36] I. Xenakis. *Formalized Music*. Pendragon Press, Stuyvesant, NY, 1992.

[37] Y.-H. Yang, D. Bogdanov, P. Herrera, and M. Sordo. Music retagging using label propagation and robust principal component analysis. In *Proc. Int. Conf. Companion on World Wide Web*, pages 869–876, 2012.

[38] Y.-H. Yang and H. H. Chen. *Music Emotion Recognition*. CRC Press, 2011.

Poster Session 1

This Page Intentionally Left Blank

TRANSFER LEARNING BY SUPERVISED PRE-TRAINING FOR AUDIO-BASED MUSIC CLASSIFICATION

Aäron van den Oord, Sander Dieleman, Benjamin Schrauwen
Electronics and Information Systems department, Ghent University
{aaron.vandenoord, sander.dieleman, benjamin.schrauwen}@ugent.be

ABSTRACT

Very few large-scale music research datasets are publicly available. There is an increasing need for such datasets, because the shift from physical to digital distribution in the music industry has given the listener access to a large body of music, which needs to be cataloged efficiently and be easily browsable. Additionally, deep learning and feature learning techniques are becoming increasingly popular for music information retrieval applications, and they typically require large amounts of training data to work well. In this paper, we propose to exploit an available large-scale music dataset, the Million Song Dataset (MSD), for classification tasks on other datasets, by reusing models trained on the MSD for feature extraction. This transfer learning approach, which we refer to as *supervised pre-training*, was previously shown to be very effective for computer vision problems. We show that features learned from MSD audio fragments in a supervised manner, using tag labels and user listening data, consistently outperform features learned in an unsupervised manner in this setting, provided that the learned feature extractor is of limited complexity. We evaluate our approach on the GTZAN, 1517-Artists, Unique and Magnatagatune datasets.

1. INTRODUCTION

With the exception of the Million Song Dataset (MSD) [3], public large-scale music datasets that are suitable for research are hard to come by. Among other reasons, this is because unwieldy file sizes and copyright regulations complicate the distribution of large collections of music data. This is unfortunate, because some recent developments have created an increased need for such datasets.

On the one hand, content-based music information retrieval (MIR) is finding more applications in the music industry, in a large part due to the shift from physical to digital distribution. Nowadays, online music stores and streaming services make a large body of music readily available to the listener, and content-based MIR can facilitate cataloging and browsing these music collections, for example by automatically tagging songs with relevant terms, or by creating personalized recommendations for the user. To develop and evaluate such applications, large music datasets are needed.

On the other hand, the recent rise in popularity of feature learning and deep learning techniques in the domains of computer vision, speech recognition and natural language processing has caught the attention of MIR researchers, who have adopted them as well [13]. Large amounts of training data are typically required for a feature learning approach to work well.

Although the initial draw of deep learning was the ability to incorporate large amounts of unlabeled data into the models using an unsupervised learning stage called *unsupervised pre-training* [1], modern industrial applications of deep learning typically rely on purely supervised learning instead. This means that large amounts of labeled data are required, and labels are usually quite costly to obtain.

Given the scarcity of large-scale music datasets, it makes sense to try and leverage whatever data is available, even if it is not immediately usable for the task we are trying to perform. We can use a *transfer learning* approach to achieve this: given a target task to be performed on a small dataset, we can train a model for a different, but related task on another dataset, and then use the learned knowledge to obtain a better model for the target task.

In image classification, impressive results have recently been attained on various datasets by reusing deep convolutional neural networks trained on a large-scale classification problem: ImageNet classification. The ImageNet dataset contains roughly 1.2 million images, divided into 1,000 categories [5]. The trained network can be used to extract features from a new dataset, by computing the activations of the topmost hidden layer and using them as features. Two recently released software packages, *OverFeat* and *DeCAF*, provide the parameters of a number of pre-trained networks, which can be used to extract the corresponding features [7, 20]. This approach has been shown to be very competitive for various computer vision tasks, sometimes surpassing the state of the art [18, 26].

Inspired by this approach, we propose to train feature extractors on the MSD for two large-scale audio-based song classification tasks, and leverage them to perform other classification tasks on different datasets. We show that this approach to transfer learning, which we will refer to as *supervised pre-training* following Girshick et al. [9],

consistently improves results on the tasks we evaluated.

The rest of this paper is structured as follows: in Section 2, we give an overview of the datasets we used for training and evaluation. In Section 3 we describe our proposed approach and briefly discuss how it relates to transfer learning. Our experiments and results are described in Section 4. Finally, we draw conclusions and point out some directions for future work in Section 5.

2. DATASETS

The **Million Song Dataset** [3] is a collection of metadata and audio features for one million songs. Although raw audio data is not provided, we were able to obtain 30 second preview clips for almost all songs from 7digital.com. A number of other datasets that are linked to the MSD are also available. These include the **Taste Profile Subset** [15], which contains listening data from 1 million users for a subset of about 380,000 songs in the form of play counts, and the **last.fm dataset**, which provides tags for about 500,000 songs. We will use the combination of these three datasets to define two *source tasks*: user listening preference prediction and tag prediction from audio.

We will evaluate four *target tasks* on different datasets:

- genre classification on the **GTZAN dataset** [22], which contains 1,000 audio clips, divided into 10 genres.
- genre classification on the **Unique** dataset [21], which contains 3,115 audio clips, divided into 14 genres.
- genre classification on the **1517-artists** dataset [21], which contains 3,180 full songs, divided into 19 genres.
- tag prediction on the **Magnatagatune** dataset [14], which contains 25,863 audio clips, annotated with 188 tags.

3. PROPOSED APPROACH

3.1 Overview

There are many ways to transfer learned knowledge between tasks. Pan and Yang [17] give a comprehensive overview of the transfer learning framework, and of the relevant literature. In their taxonomy, our proposed supervised pre-training approach is a form of *inductive transfer learning* with *feature representation transfer*: target labels are available for both the source and target tasks, and the feature representation learned on the source task is reused for the target task.

In the context of MIR, transfer learning has been explored by embedding audio features and labels from various datasets into a shared latent space with linear transformations [10]. The same shared embedding approach has previously been applied to MIR tasks in a multi-task learning setting [24]. We refer to these papers for a discussion of some other work in this area of research.

For supervised pre-training, it is essential to have a source task that requires a very rich feature representation, so as to ensure that the information content of this representation is likely to be useful for other tasks. For computer vision problems, ImageNet classification is one such task, since it involves a wide range of categories. In this paper, we will evaluate two source tasks using the MSD: tag prediction and user listening preference prediction from audio. The goal of tag prediction is to automatically determine which of a large set of tags are associated with a given song. User listening preference prediction involves predicting whether users have listened to a given song or not.

Both tasks differ from typical classification tasks in a number of ways:

- Tag prediction is a *multi-label classification* task: each song can be associated with multiple tags, so the classes are not disjoint. The same goes for user listening preference prediction, where we attempt to predict for each user whether they have listened to a song. The listening preferences of different users are not disjoint either, and one song is typically listened to by multiple users.
- There are large numbers of tags and users; orders of magnitude larger than the 1,000 categories of ImageNet.
- The data is weakly labeled: if a song is not associated with a particular tag, the tag may still be applicable to the song. In the same way, if a user has not listened to a song, they may still enjoy it (i.e. it would be a good recommendation). In other words, some positive labels are missing.
- The labels are redundant: a lot of tags are correlated, or have the same meaning. For example, songs tagged with *disco* are more likely to also be tagged with *80's*. The same goes for users: many of them have similar listening preferences.
- The labels are very sparse: most tags only apply to a small subset of songs, and most users have only listened to a small subset of songs.

We will tackle some of the problems created by these differences by first performing dimensionality reduction in the label space using *weighted matrix factorization* (WMF, see Section 3.2), and then training models to predict the reduced label representations instead.

We will first use the spherical K-means algorithm (see Section 3.3) to learn low-level features from audio spectrograms, and use them as input for the supervised models that we will train to perform the source tasks. Feature learning using K-means is very fast compared to other unsupervised feature learning methods, and yields competitive results. It has recently gained popularity for content-based MIR applications [6, 19, 25].

In summary, our workflow will be as follows: we will first learn low-level features from audio spectrograms, and apply dimensionality reduction to the target labels. We will train supervised models to predict the reduced label representations from the extracted low-level audio features. These models can then be used to perform the source tasks. Next, we will use the trained models to extract higher-level features from other datasets, and use those features to train shallow classifiers for different but related target tasks. We will compare the higher-level features obtained from different model architectures and different source tasks by

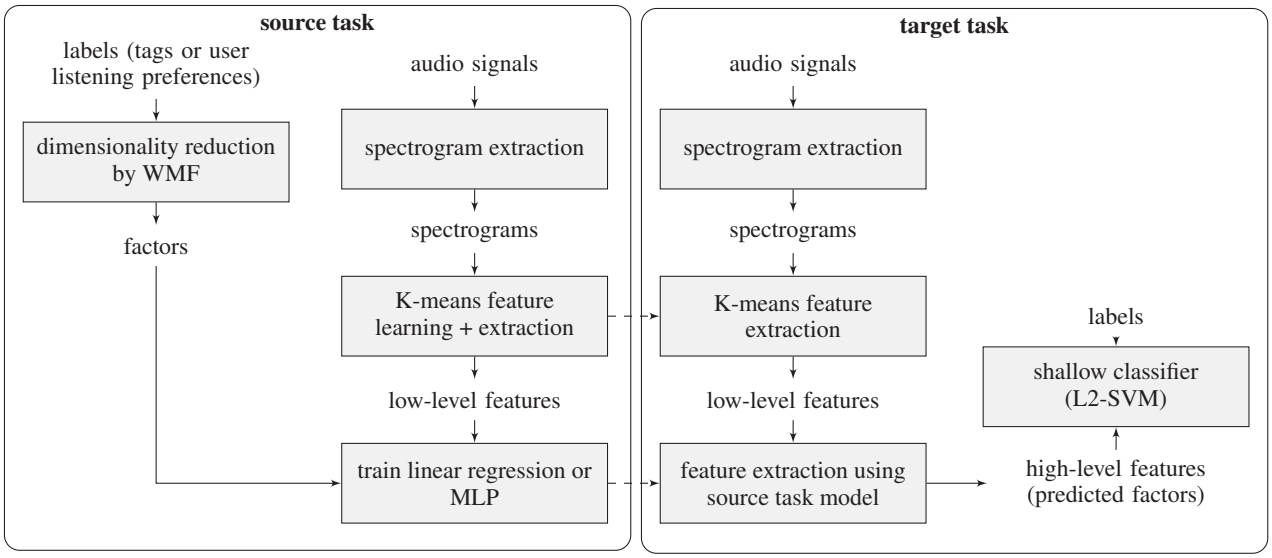


Figure 1: Schematic overview of the workflow we will use for our supervised pre-training approach. Dashed arrows indicate transfer of the learned feature extractors from the source task to the target task.

evaluating their performance on these target tasks. This workflow is visualized in Figure 1. The key learning steps are detailed in the following subsections.

3.2 Dimensionality reduction in the label space

To deal with large numbers of overlapping labels, we first consider the matrix of labels for all examples, and perform weighted matrix factorization (WMF) on it [12]. Given a binary $m \times n$-matrix A (m examples and n labels), WMF will find an $m \times f$-matrix U and an $n \times f$-matrix V, so that $A \approx UV^T$. The hyperparameter f controls the rank of the resulting approximation. This approximation is found by optimizing the following weighted objective function:

$$J(U,V) = C \circ (A - UV^T)^2 + \lambda(||U||_F^2 + ||V||_F^2),$$

where C is a $m \times n$ *confidence matrix*, $\circ$ represents elementwise multiplication, the squaring is elementwise as well, and λ is a regularization parameter. If the confidence values in C are chosen to be 1 for all zeroes in A, an efficient alternating least squares (ALS) method exists to optimize $J(U,V)$, provided that A is sparse. For details, we refer to Hu et al. [12].

After optimization, each row of U can be interpreted as a reduced representation of the m labels associated with the corresponding example, which captures the latent factors that affect its classification. We can then train a model to predict these f factors instead, which is much easier than predicting m labels directly (typically $f \ll m$). We have previously used a similar approach to do content-based music recommendation with a convolutional neural network [23]. In that paper, we showed that these factors capture a lot of relevant information and can also be used for tag prediction. We use the same settings and hyperparameter values for the WMF algorithm in this work.

Our choice for WMF over other dimensionality reduction methods, such as PCA, is motivated by the particular structure of the label space described earlier. WMF allows for the sparsity and redundancy of the labels to be exploited, and we can take into account that the data is weakly labeled by choosing C so that positive signals are weighed more than negative signals.

The original label matrix for the tag prediction task has 173,203 columns, since we included all tags from the last.fm dataset that occur more than once. The matrix for the user listening preference prediction task has 1,129,318 columns, corresponding to all users in the Taste Profile Subset. By applying WMF, we obtain reduced representations with 400 factors for both tasks. These factors will be treated as ground truth target values in the supervised learning phase.

3.3 Unsupervised learning of low-level features

We learn a low-level feature representation from spectrograms in an unsupervised manner, to use as input for the supervised pre-training stage. First, we extract log-scaled mel-spectrograms from single channel audio signals, with a window size of 1024 samples and a hop size of 512. Conversion to the mel scale reduces the number of frequency components to 128. We then use the spherical K-means algorithm (as suggested by Coates et al. [4]) to learn 2048 bases from randomly sampled PCA-whitened windows of 4 consecutive spectrogram frames. This is similar to the feature learning approach proposed by Dieleman et al. [6].

To extract features, we divide the spectrograms into overlapping windows of 4 frames, and compute the dot product of each base with each PCA-whitened window. We then aggregate the feature values across time by computing the maximal value for each base across groups of consecutive windows corresponding to about 2 seconds of audio. Finally, we take the mean of these values across the entire audio clip to arrive at a 2048-dimensional feature representation for each example. This two-stage temporal pooling approach turns out to work well in practice.

3.4 Supervised learning of high-level features

For both source tasks, we train three different model architectures to predict the reduced label representations from the low-level audio features: a linear regression model, a multi-layer perceptron (MLP) with a hidden layer with 1000 rectified linear units (ReLUs) [16], and an MLP with two such hidden layers. The MLPs are trained using stochastic gradient descent (SGD) to mimize the mean squared error (MSE) of the predictions, and dropout regularization [11]. The training procedure was implemented using Theano [2].

We trained all these models on a subset of the MSD, consisting of 373,855 tracks for which we were able to obtain audio samples, and for which listening data is available in the Taste Profile Subset. We used 308,443 tracks for training, 18,684 for validation and 46,728 for testing. For the tag prediction task, the set of tracks was further reduced to 253,588 tracks, including only those for which tag data is available in the last.fm dataset. For this task, we used 209,218 tracks for training, 12,763 for validation and 31,607 for testing.

The trained models can be used to extract high-level features simply by computing predictions for the reduced label representations and using those as features, yielding feature vectors with 400 values. For the MLPs, we can alternatively compute the activations of the topmost hidden layer, yielding feature vectors with 1000 values instead. The latter approach is closer to the original interpretation of supervised pre-training as described in Section 1, but since the trained models attempt to predict latent factor representations, the former approach is viable as well. We will compare both.

To evaluate the models on the source tasks, we compute the predicted factors U' and obtain predictions for each class by computing $A' = U'V^T$. This matrix can then be used to compute performance metrics.

3.5 Evaluation of the features for target tasks

To evaluate the high-level features for the target tasks outlined in Section 2, we train linear L2-norm support vector machines (L2-SVMs) for all tasks with liblinear [8], using the features as input. Although using more powerful classifiers could probably improve our results, the use of a shallow, linear classifier helps to assess the quality of the input features.

4. EXPERIMENTS AND RESULTS

4.1 Source tasks

To assess whether the models trained for the source tasks are able to make sensible predictions, we evaluate them by computing the normalized mean squared error (NMSE) [1] of the latent factor predictions, as well as the area under the ROC curve (AUC) and the mean average precision (mAP) of the class predictions [2]. They are reported in Table 1. Note that the latter two metrics are computed on a separate test set, but the former is computed on the validation set that we also used to optimize the hyperparameters for the dimensionality reduction of the labels. This is because the ground truth latent factors, which are necessary to compute the NMSE, are not available for the test set.

User listening preference prediction			
Model	**NMSE**	**AUC**	**mAP**
Linear regression	0.986	0.750	0.0076
MLP (1 hidden layer)	0.971	0.760	0.0149
MLP (2 hidden layers)	0.961	0.746	0.0186

Tag prediction			
Model	**NMSE**	**AUC**	**mAP**
Linear regression	0.965	0.823	0.0099
MLP (1 hidden layer)	0.939	0.841	0.0179
MLP (2 hidden layers)	0.924	0.837	0.0179

Table 1: Results for the source tasks. For all three models, we report the normalized mean squared error (NMSE) on the validation set, and the area under the ROC curve (AUC) and the mean average precision (mAP) on a separate test set.

It is clear that using a more complex model (i.e. an MLP) results in better predictions of the latent factors in the least-squares sense, as indicated by the lower NMSE values. However, when using the AUC metric, this does not always seem to translate into better performance for the task at hand: MLPs with only a single hidden layer perform best for both tasks in this respect. The mAP metric seems to follow the NMSE on the validation set more closely.

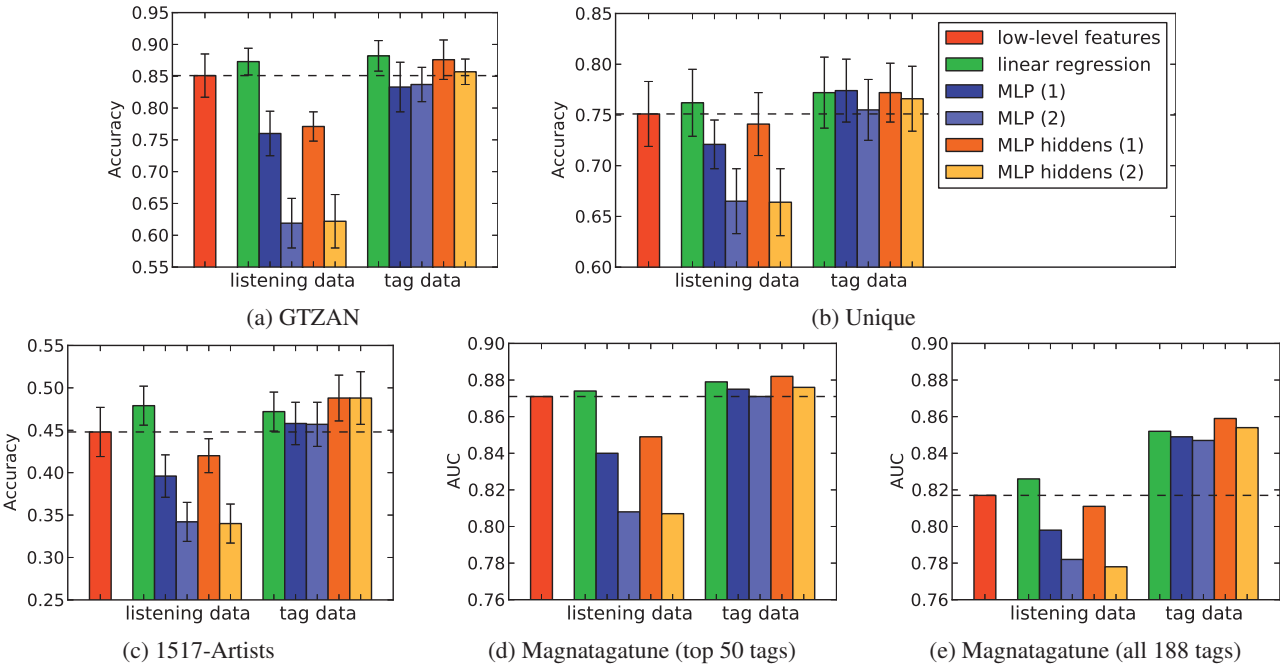
Although the NMSE values are relatively high, the class prediction metrics indicate that the predicted factors still yield acceptable results for the source tasks. In our preliminary experiments we also observed that using fewer factors tends to result in lower NMSE values. In other words, as we add more factors, they become less predictable. This implies that the most important latent factors extracted from the labels are also the most predictable from audio.

4.2 Target tasks

We report the L2-SVM classification performance of the different feature sets across all target tasks in Figure 2. For the GTZAN, Unique and 1517-Artists datasets, we report the average cross-validation classification accuracy across 10 folds. Error bars indicate the standard deviations across folds. We optimize the SVM regularization parameter using nested cross-validation with 5 folds. Magnatagatune comes divided into 16 parts; we use the first 11 for training and the next 2 for validation. After hyperparameter optimization, we retrain the SVMs on the first 13 parts, and the last 3 are used for testing. We report the AUC aver-

[1] The NMSE is the MSE divided by the variance of the target values across the dataset.

[2] The class predictions are obtained by multiplying the factor predictions with the matrix V^T, as explained in the previous section.



(a) GTZAN (b) Unique

(c) 1517-Artists (d) Magnatagatune (top 50 tags) (e) Magnatagatune (all 188 tags)

Figure 2: Target task performance of the different feature sets. The dashed line represents the performance of the low-level features. From left to right, the five bars in the bar groups represent high-level features extracted with linear regression, an MLP with 1 hidden layer, an MLP with 2 hidden layers, the hidden layer of a 1-layer MLP, and the topmost hidden layer of a 2-layer MLP respectively. Error bars for the first three classification tasks indicate the standard deviation across cross-validation folds. For Magnatagatune, no error bars are given because no cross-validation was performed.

aged across tags for the 50 most frequently occuring tags (Figure 2d), and for all 188 tags (Figure 2e).

The single bar on the left of each graph shows the performance achieved when training an L2-SVM directly on the low-level features learned using spherical K-means. The two groups of five bars show the performance of the high-level features trained in a supervised manner for the user listening preference prediction task and the tag prediction task respectively.

Across all tasks, using the high-level features results in improved performance over the low-level features. This effect is especially pronounced for Magnatagatune, when predicting all 188 tags from the high-level features learned on the tag prediction source task. This makes sense, as some of the Magnatagatune tags are quite rare, and features learned on this closely related source task must contain at least some relevant information for these tags.

Comparing the performance of different source task models for user listening preference prediction, model complexity seems to play a big role. Across all datasets, features learned with linear regression perform much better than MLPs, despite the fact that the MLPs perform better for the source task. Clearly the MLPs are able to achieve a better fit for the source task, but in the context of transfer learning, this is actually a form of overfitting, as the features generalize less well to the target tasks – they are too specialized for the source task. This effect is not observed when the source task is tag prediction, because this task is much more closely related to the target tasks. As a result, a better fit for the source task is more likely to result in better generalization across tasks.

For MLPs, there is a limited difference in performance between using the predictions or the topmost hidden layer activations as features. Sometimes the latter approach works a bit better, presumably because the feature vectors are larger (1000 values instead of 400) and sparser.

On GTZAN, we are able to achieve a classification accuracy of 0.882 ± 0.024 using the high-level features obtained from a linear regression model for the tag prediction task, which is competitive with the state of the art. If we use the low-level features directly, we achieve an accuracy of 0.851 ± 0.034. This is particularly interesting because the L2-SVM classifier is linear, and the features obtained from the linear regression model are essentially linear combinations of the low-level features.

5. CONCLUSION AND FUTURE WORK

We have proposed a method to perform supervised feature learning on the Million Song Dataset (MSD), by training models for large-scale tag prediction and user listening preference prediction. We have shown that features learned in this fashion work well for other audio classification tasks on different datasets, consistently outperforming a purely unsupervised feature learning approach.

This transfer learning approach works particularly well when the source task is tag prediction, i.e. when the source task and the target task are closely related. Acceptable results are also obtained when the source task is user listening preference prediction, although it is important to restrict the complexity of the model in this case. Otherwise, the features become too specialized for the source task,

which hampers generalization to other tasks and datasets.

In future work, we would like to investigate whether we can achieve transfer from more complex models trained on the user listening preference prediction task, and other tasks that are less closely related to the target tasks. Since a lot of training data is available for this task, using more powerful models than linear regression to learn features is desirable, especially considering the complexity of models used for supervised pre-training in the computer vision domain. This will require a different regularization strategy that takes into account generalization to other tasks and datasets, and not just to new examples within the same task, as it seems that these two do not always correlate. We will also look into whether using different dimensionality reduction techniques instead of WMF can lead to representations that enable better transfer to new tasks.

6. REFERENCES

[1] Yoshua Bengio. Learning deep architectures for AI. Technical report, Dept. IRO, Université de Montreal, 2007.

[2] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010.

[3] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proceedings of the 11th International Conference on Music Information Retrieval (ISMIR)*, 2011.

[4] Adam Coates and Andrew Y. Ng. Learning feature representations with k-means. *Neural Networks: Tricks of the Trade, Reloaded*, 2012.

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[6] Sander Dieleman and Benjamin Schrauwen. Multiscale approaches to music audio feature learning. In *Proceedings of the 14th International Conference on Music Information Retrieval (ISMIR)*, 2013.

[7] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. *CoRR*, abs/1310.1531, 2013.

[8] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[9] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013.

[10] Philippe Hamel, Matthew EP Davies, Kazuyoshi Yoshii, and Masataka Goto. Transfer learning in MIR: sharing learned latent representations for music audio classification and similarity. In *ISMIR 2013*, 2013.

[11] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. Technical report, University of Toronto, 2012.

[12] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, 2008.

[13] Eric J. Humphrey, Juan P. Bello, and Yann LeCun. Moving beyond feature design: Deep architectures and automatic feature learning in music informatics. In *Proceedings of the 13th International Conference on Music Information Retrieval (ISMIR)*, 2012.

[14] Edith Law and Luis von Ahn. Input-agreement: a new mechanism for collecting data using human computation games. In *Proceedings of the 27th international conference on Human factors in computing systems*, 2009.

[15] Brian McFee, Thierry Bertin-Mahieux, Daniel P.W. Ellis, and Gert R.G. Lanckriet. The million song dataset challenge. In *Proceedings of the 21st international conference companion on World Wide Web*, 2012.

[16] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010.

[17] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(10):1345–1359, 2010.

[18] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. *CoRR*, abs/1403.6382, 2014.

[19] Jan Schlüter and Christian Osendorfer. Music Similarity Estimation with the Mean-Covariance Restricted Boltzmann Machine. In *Proceedings of the 10th International Conference on Machine Learning and Applications (ICMLA)*, 2011.

[20] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *CoRR*, abs/1312.6229, 2013.

[21] Klaus Seyerlehner, Gerhard Widmer, and Tim Pohle. Fusing block-level features for music similarity estimation. In *Proc. of the 13th Int. Conference on Digital Audio Effects (DAFx-10)*, pages 225–232, 2010.

[22] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10:293–302, 2002.

[23] Aäron van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In *Advances in Neural Information Processing Systems 26*, 2013.

[24] Jason Weston, Samy Bengio, and Philippe Hamel. Large-scale music annotation and retrieval: Learning to rank in joint semantic spaces. *Journal of New Music Research*, 2011.

[25] J. Wülfing and M. Riedmiller. Unsupervised learning of local features for music classification. In *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR)*, 2012.

[26] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013.

ESTIMATING MUSICAL TIME INFORMATION FROM PERFORMED MIDI FILES

Harald Grohganz, Michael Clausen
Bonn University
{grohganz,clausen}@cs.uni-bonn.de

Meinard Müller
International Audio Laboratories Erlangen
meinard.mueller@audiolabs-erlangen.de

ABSTRACT

Even though originally developed for exchanging control commands between electronic instruments, MIDI has been used as quasi standard for encoding and storing score-related parameters. MIDI allows for representing musical time information as specified by sheet music as well as physical time information that reflects performance aspects. However, in many of the available MIDI files the musical beat and tempo information is set to a preset value with no relation to the actual music content. In this paper, we introduce a procedure to determine the musical beat grid from a given performed MIDI file. As one main contribution, we show how the global estimate of the time signature can be used to correct local errors in the pulse grid estimation. Different to MIDI quantization, where one tries to map MIDI note onsets onto a given musical pulse grid, our goal is to actually estimate such a grid. In this sense, our procedure can be used in combination with existing MIDI quantization procedures to convert performed MIDI files into semantically enriched score-like MIDI files.

1. INTRODUCTION

MIDI (Music Instrument Digital Interface) is used as a standard protocol for controlling and synchronizing electronic instruments and synthesizers [10]. Even though MIDI has not originally been developed to be used as a symbolic music format and imposes many limitation of what can be actually represented [11, 13], the importance of MIDI results from its widespread usage over the last three decades and the abundance of MIDI data freely available on the web. An important feature of the MIDI format is that it can handle musical as well as physical onset times and note durations. In particular, the header of a MIDI file specifies the number of basic time units (referred to as ticks) per quarter note. Physical timing is then given by means of additional tempo messages that determine the number of microseconds per quarter note. On the one hand, disregarding the tempo messages makes it possible to generate a mechanical version of constant tempo, which closely relates to the musical time axis (given in beats) of a score. On the other hand, by including the tempo messages, one may generate a performed version with a physical time axis (given in seconds). However, many of the available MIDI files do not follow this convention. For example, MIDI files are often generated by freely performing a piece of music on a MIDI instrument without explicitly specifying the tempo. As a result, neither the ticks-per-quarter-note parameter nor the tempo messages are set in a musically meaningful way. Instead, these parameters are given by presets, which makes it possible to derive the physical but not the musical time information.

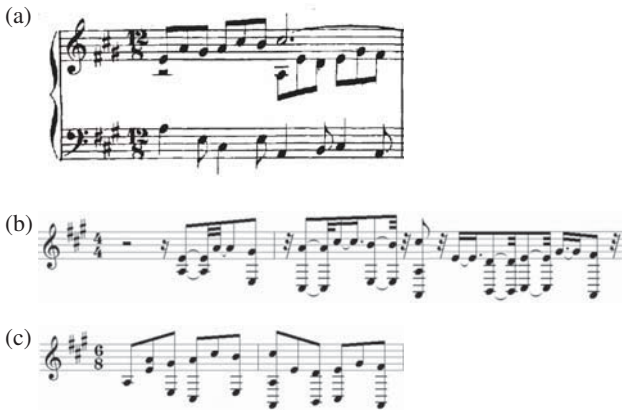


Figure 1. The first measure of the prelude BWV 888 by J. S. Bach. **(a)** Original score. **(b)** Score from P-MIDI of a performed version without musical pulse grid. **(c)** Score from S-MIDI based on an estimated musical pulse grid.

In the following, we distinguish between two types of MIDI files. When the musical beat and tempo messages are set correctly in a MIDI file, then a musical time axis as specified by a score can be derived. In this case, we speak of a *score-informed MIDI file* or simply S-MIDI. When the actual tempo and beat positions are not known (using some presets), we speak of a *performed MIDI file* or simply P-MIDI. This paper deals with the general problem of converting a P-MIDI into a reasonable approximation of an S-MIDI file. The main step is to estimate a musically informed beat or pulse grid from which one can derive the musical time axis. The general problem of estimating beat- and rhythm-related information from music representation (including MIDI and audio representations) is a difficult problem [1, 7]. Typically approaches are based on Hidden

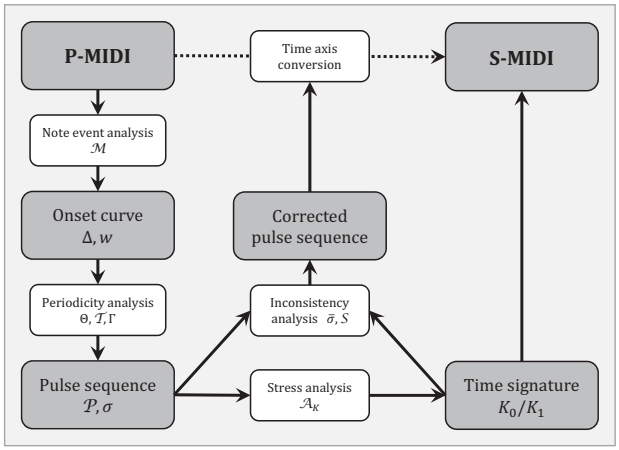


Figure 2. Overview of algorithmic pipeline.

Markov Models [12] and dynamic programming [6, 15]. Even when knowing the note onset positions explicitly (as is the case for MIDI files), finding beats and measure is by far not trivial—in particular when dealing with performed MIDIs having local tempo fluctuations. In [2], an approach based on salience profiles of MIDI notes is used for estimating the time signature and measure positions. Based on a trained dictionary of rhythmical patterns, a more general approach for detecting beat, measure, and rhythmic information is described in [14]. Note that the extraction of such musical time information from MIDI files is required before software for MIDI quantization and score generation can be applied in a meaningful way. This is demonstrated by Figure 1, which shows the original score, the score generated from a P-MIDI, and a score generated from an estimated S-MIDI.

In this paper, we introduce a procedure for estimating the musical beat grid as well as the time signature from a given P-MIDI file, which can then be converted into an approximation of an S-MIDI file. [1] The main idea is to adapt a beat tracking procedure originally developed for audio representations to estimate a first pulse grid. Despite of local errors, this information suffices to derive an estimate of a global time signature. This information, in turn, is then used to correct the local pulse irregularities. In Section 2, we describe the algorithmic details of our proposed method. Then, in Section 3, we evaluate our method and discuss a number of explicit examples to illustrate benefits and limitations. We conclude the paper with Section 4 with possible applications and an outlook on future work. Further related work is discussed in the respective sections.

2. ALGORITHMIC PIPELINE

In this section, we describe our procedure for converting P-MIDI files into (approximations of) S-MIDI files by mapping the physical time axis of the P-MIDI to an appropriate musical time axis. As shown in Figure 2, we extract an onset curve from the P-MIDI, and perform periodicity analysis by adapting a pulse tracking method to obtain a sequence of pulse candidates (Section 2.1). In Section 2.2 we introduce a method to estimate the global time signature by analyzing the stress distribution of the pulses. This information is used for detecting and resolving inconsistencies in the pulse sequence.

[1] Our implementation in Java with GUI is available at `http://midi.sechsachtel.de`

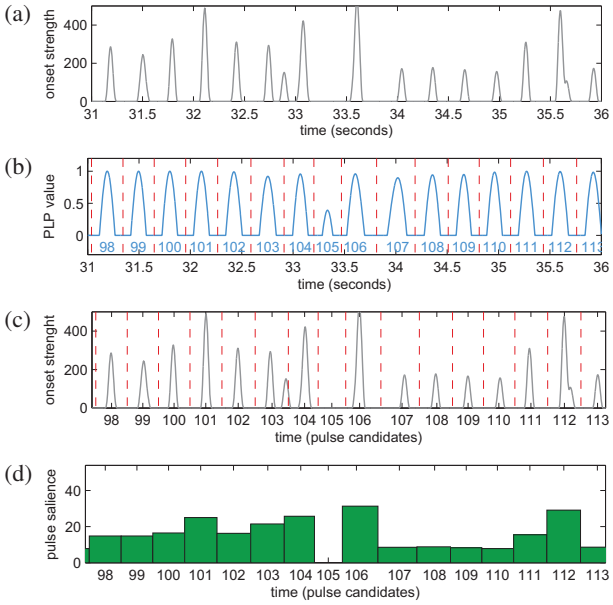


Figure 3. Computation of pulse salience for a 5-second excerpt of BWV 888: **(a)** MIDI onset curve Δ, **(b)** PLP curve Γ with pulse region boundaries b, **(c)** Onset curve Δ with boundaries b, **(d)** Pulse salience sequence σ.

In the following we use the notation $[a : n : b] := (\lceil a \rceil + n\mathbb{N}_0) \cap [a, b]$, where $[a, b] := \{t \in \mathbb{R} \mid a \leq t \leq b\}$ for $a, b \in \mathbb{R}$ and $n \in \mathbb{N}$. If $n = 1$, we use the notation $[a : b] := [a : 1 : b]$.

2.1 Pulse Detection

For pulse tracking, we build upon the method introduced by [9] which detects the local predominant periodicity in onset curves, and generates a pulse curve indicating the most likely positions for a pulse-grid. The peaks of this curve are then interpreted as pulse candidates. Although this method was originally developed for audio data like other beat tracking methods (see, e. g., [4,6]), it also works for onset curves derived from MIDI files.

We assume that the MIDI file is already converted to a physical time axis $[0, T]$, where T denotes the end of the last MIDI note, and we have a *MIDI note list* for a suitable finite index set $I \subset \mathbb{N}$:

$$\mathcal{M} := (t_i, d_i, p_i, v_i)_{i \in I},$$

where $t_i \in [0, T)$ describes the start time of the i^{th} MIDI note, d_i its duration (also in seconds), $p_i \in [0 : 127]$ its pitch, and $v_i \in [0 : 127]$ its note onset velocity. Based on these notes, we define for a weighting parameter $w = (w_1, w_2, w_3) \in \mathbb{R}^3$, a MIDI onset curve

$$\Delta_w(t) := \textstyle\sum_{i \in I} (w_1 + w_2 \cdot d_i + w_3 \cdot v_i) \cdot h(t - t_i),$$

for $t \in [0, T]$, with h describing a Hann window centered

at 0 of length 50 ms, cf. Figure 3a. Thus the components of the parameter w corresponds to weights of the presence of an onset, the duration, and the velocity, respectively. In our procedure, we fix $w := \left(1, 20, \frac{50}{128}\right)$ to balance the components of each MIDI note, so it will be omitted in the notation. Experiments have shown that the method is robust to slight changes of these values.

Using a short-time Fourier transform, we compute from Δ a *tempogram* $\mathcal{T} : [0, T] \times \Theta \to \mathbb{C}$ for a given set Θ of considered BPM values as explained in [9], using parameters for smoothness (window length) and time granularity (step size). First, we compute a coarse tempogram $\mathcal{T}^{\text{coarse}}$ using the tempo set $\Theta = [40 : 4 : 240]$, window length 8 sec, and step size 1 sec. The dominant global tempo T_0 is derived by summing up the absolute values of $\mathcal{T}^{\text{coarse}}$ row-wise and by detecting the maximum. Next, we compute a second tempogram $\mathcal{T}^{\text{fine}}$ based on the new set $\Theta = \left[\frac{1}{\sqrt{2}} \cdot T_0, \sqrt{2} \cdot T_0\right] \cap \mathbb{N}$, which is the *tempo octave* around T_0. For this tempogram, the window length is set to $5 \cdot \frac{60}{T_0}$ sec, and we use a finer step size of 0.2 sec. Choosing the BPM range in such a manner prevents unexpected jumps between multiples of the detected tempo; the window length corresponds to five expected pulses based on the assumption that a stable tempo remains almost constant for at least five beats.

Following [9], we estimate the predominant tempo for each time position from the tempogram $\mathcal{T}^{\text{fine}}$, and use this information to derive sinusoidal kernels which best describe local periodicity of the underlying onset curve Δ. These kernels are combined to a *predominant local pulse* (PLP) curve $\Gamma : [0, T] \to [0, 1]$, which indicates positions of pulses on the physical time axis, see Figure 3b. The points in time corresponding to the local maxima of Γ form a *pulse candidate* sequence $\mathcal{P} = (\mathcal{P}_1, \ldots, \mathcal{P}_N)$, which is suitable to estimate the beats in a first approximation. But this sequence may contain additional pulses (not describing a musical beat) or missing pulses. Thus we introduce a post-processing method in the next section which detects and corrects these errors.

2.2 Optimizing the pulse sequence

The main idea of the method described in this section relies on finding a global time signature and using it for resolving inconsistencies of the detected pulse sequence. Here, we assume that the measure type of the considered musical piece is not changed throughout the piece. The time signature can be estimated by periodicity analysis of pulse stress using short-time autocorrelation. In a second step, we compare the relative position of each pulse candidate to a measure grid induced by the time signature and detect deviations to correct *isolated* erroneous pulses. Finally, the pulses are interpreted as a new musical time axis, and the tick position of all MIDI events are mapped to this axis.

Now we describe our optimization procedure in more detail. First, we accumulate the onset strength for the n^{th} pulse candidate by defining its *pulse salience*

$$\sigma(n) := \int_{b(n-1)}^{b(n)} \Delta(t)\, dt \quad (n \in [1 : N]), \qquad (1)$$

where the pulse region boundaries are given by $b(n) = \frac{1}{2} \cdot (\mathcal{P}_n + \mathcal{P}_{n+1})$ for $1 \leq n < N$, $b(0) = 0$, and $b(N) = T$. The boundaries b are illustrated in Figures 3b and 3c, and for the salience values σ see Figures 3d and 4a.

Our next goal is to compute an estimation of the time signature K_0/K_1. To this end we perform a salience analysis via autocorrelation. However, to ensure that errors in $\mathcal{P}$ and σ have only a local influence, we use short-time autocorrelation. For a fixed window size $K > 12$ ($K = 32$ in our implementation), we consider the $K \times N$ matrix

$$\mathcal{A}(k, n) := |\, I_k \,|^{-1} \sum_{i \in I_k} \sigma(n + i) \cdot \sigma(n + i + k),$$

where $I_k := [0 : k : K - k - 1]$ and $\sigma(n) := 0$ for $n \in \mathbb{Z} \setminus [1 : N]$. Thus $\mathcal{A}(k, n)$ quantifies the plausibility of period length k around the n^{th} pulse candidate. Our predominant salience period K_0, the nominator of the estimated time signature, is obtained by row-wise summation and maximum picking of parts of $\mathcal{A}$:

$$K_0 := \operatorname*{arg\,max}_{k \in [3:12]} \textstyle\sum_{n=1}^{N} \mathcal{A}(k, n).$$

For robustness and musical reasons we have excluded the cases $k < 3$ and $k > 12$, respectively. (Excluding the case $k = 2$ is not a serious problem as we can use, e. g., $4/8$ as surrogate for $2/4$.) The relevant rows of the matrix $\mathcal{A}$ are illustrated in Figure 4b where $K_0 = 6$. The denominator K_1 is not necessary for further computation. It is chosen accordingly to the main tempo T_0 to ensure a value between 70 and 140 quarter notes per minute.

With the help of K_0 we are now able to perform the inconsistency analysis. For now, we primarily consider the case where all detected pulse candidates are actually correct beats. In this idealized scenario, the restriction of $\mathcal{P}$ to the n^{th} K_0-congruence class $[n : K_0 : N]$, $n \in [1 : K_0]$, describes the n^{th} position within the measures in a semantically meaningful way. In particular, the first class ($n = 1$) corresponds to all downbeat positions if the considered piece does not start with an upbeat. An analogous decomposition applied to σ leads to salience patterns of each position in the measure. Due to rhythmic variations, we expect that the first class of σ *mostly* shows the highest salience value. To enhance robustness, σ is smoothed locally within the K_0-congruence classes

$$\bar{\sigma}(n) := \sigma(n) + \sum_{k=1}^{\lfloor K/K_0 \rfloor} \sigma(n \pm k \cdot K_0),$$

as illustrated in Figure 4c. Since the restriction to a congruence class is reminiscent of a comb, we call $\bar{\sigma}$ the K_0-*combed* version of σ.

Erroneously detected pulse candidates disturb the assignment of all downbeats to a specific class. In such cases, the class containing highest salience values changes at some points of time. To make this visible, a $K_0 \times N$ matrix $\mathcal{S}$ is defined which shows the local salience distribution of the congruence classes. More precisely, we define

$$S(k, n) := \bar{\sigma}(k) \cdot \delta(k \equiv_{K_0} n),$$

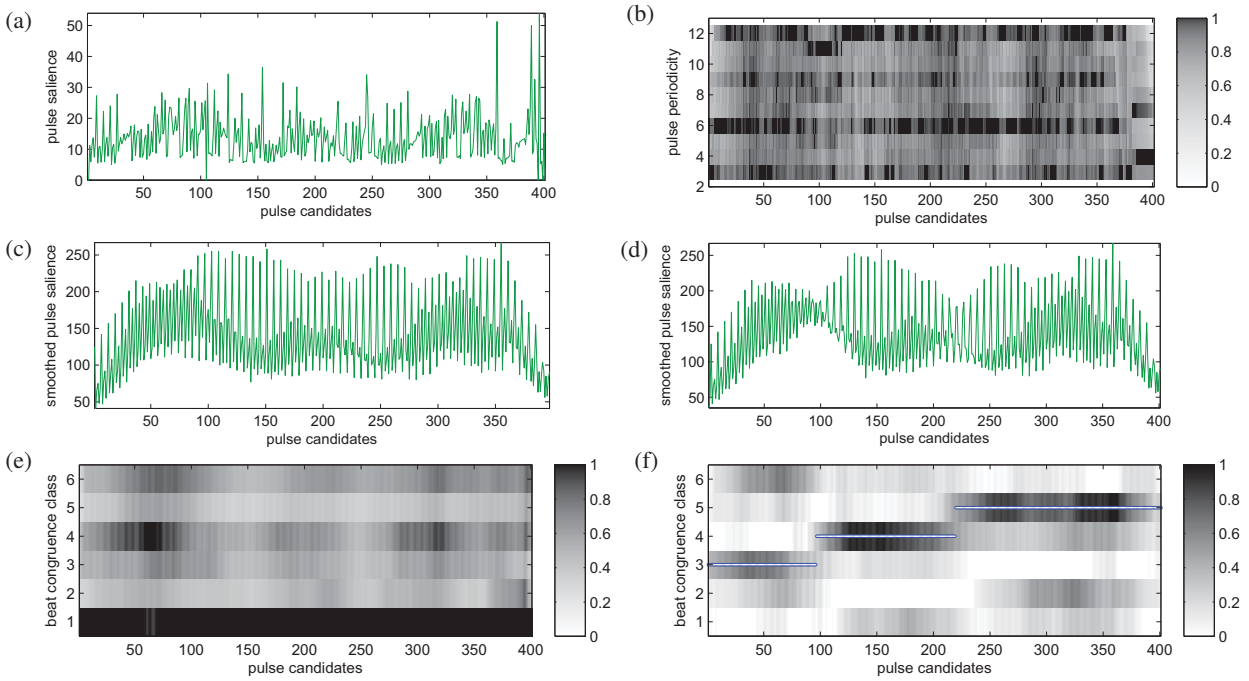


Figure 4. Step-by-step illustration how to detect inconsistencies in the stress sequence for BWV 888: **(a)** Pulse salience sequence σ as in Figure 3d. **(b)** Excerpt of short-time autocorrelation matrix $\mathcal{A}$ of σ showing maximal energy in 6^{th} row. **(c)** 6-combed salience sequence $\bar{\sigma}$ if all pulse candidates are correctly detected. **(d)** 6-combed salience sequence $\bar{\sigma}$ if two pulses are additionally inserted. **(e)** Stressgram showing maximal salience in 1^{st} congruence class. **(f)** Stressgram showing two stress changes and path of highest salience.

where $\delta(A) := 1$ if statement A holds and 0 else. Smoothing S along the temporal axis using a Hann window of length $2 \cdot K_0$ yields a so-called *stressgram* $\mathcal{S}$. Such stressgrams are visualized for the ideal scenario (Figure 4e) as well as under presence of two additional pulses (Figure 4f).

Now we discuss this last case in more detail. First, the estimation of K_0 is only locally disturbed which does not lead to a change of the estimated time signature. However, the decomposition into K_0-congruence classes does no longer coincide semantically with the position in the measures, since all pulses after the additional one are shifted by one beat position. In the stressgram $\mathcal{S}$ this is indicated by changes of the rows showing high salience. To enhance robustness, we switch to a more global point of view by computing a path of highest energy through $\mathcal{S}$ using dynamic programming. Each point in this path shows the congruence class with the highest coincidence of representing the downbeats at a specific time. More precisely, if the downbeats are in the class having index i, then a change to index $i+1$ near the additional pulse can be noticed, see Figure 4f. The case of a missing pulse is similar, here the row index of the maximal salience changes to $i-1$.

These detected irregularities can now be solved by choosing either falsely added pulse candidates or finding positions to insert an apparently missing pulse. For lack of space, we only sketch our correction procedure. To remove a candidate, one can delete the pulse having lowest salience σ or lowest PLP score (for this, replace Δ by Γ in Equation 1). For adding an additional pulse, one may look for two adjacent relatively low values of the PLP curve.

Finally, this corrected pulse sequence defines a beat grid in the P-MIDI file, which allows to detect a sequence of tick positions corresponding to beats. By mapping them to equally distributed new tick positions, adding appropriate tempo change MIDI messages, and performing linear interpolation between the beat positions, the previous time axis of the P-MIDI is replaced by a musical time axis. In case of an upbeat, additional beats are added to the beginning of the piece such that the first K_0-congruence class corresponds to the estimated downbeats. Lastly, the time signature K_0/K_1 is added to the new MIDI file.

3. EXPERIMENTS AND DISCUSSION

Evaluating the output of a beat tracking procedure is a non-trivial task due to the vague definition of beat times as described in [5]. Particularly determining the beat granularity, i. e., the decision between similar time signatures like $6/8$ and $3/4$, or multiples such as $4/4$ or $8/4$, appears as an ill-posed and negligible problem. Even for humans, beat and measure tracking can be challenging especially in the presence of rhythmic variations and expressive timing. Our evaluation is inspired by [14], where among others the visual impression of the computed score is considered, and by [5], where comparison to a ground truth annotation by a human and listening tests for a perceptual evaluation are suggested.

Because of its modeling, our procedure is not suitable for all kinds of P-MIDI files. The PLP approach described in Section 2.1 has some constraints like a stable rhythm or an almost stable tempo for a certain amount of time (in our

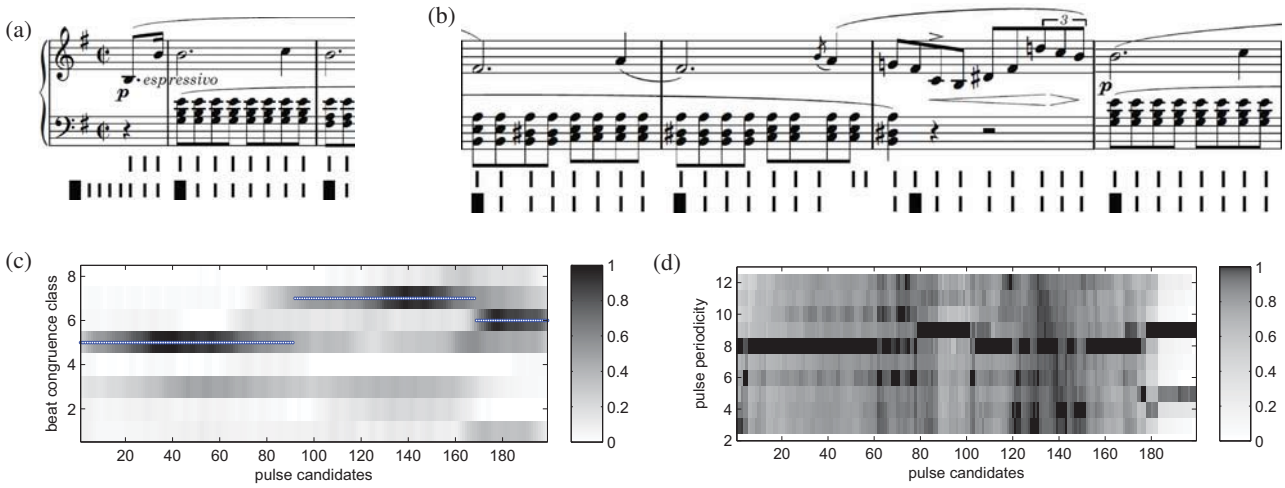


Figure 5. Prelude No. 4 from Chopin (Op. 28). The score excerpts show the detected pulses (upper row) and post-processed measure grid (lower row). Downbeats are indicated by bold lines. **(a)** Correctly detected upbeat. **(b)** Joint correction of two subsequent errors. **(c)** Stressgram with path of highest salience. **(d)** Short-time autocorrelation matrix.

implementation, this time window is roughly five seconds). In addition, tempo octave confusions are not considered here. For the optimization step described in Section 2.2, a global time signature is required. Furthermore, downbeats must be detectable by their length or stress. In the following, we discuss in detail two typical examples of P-MIDI files, and then perform an automatic analysis on a small test set of artificially distorted MIDI files.

3.1 Qualitative Evaluation

The performance of the proposed procedure for Bach's Prelude BWV 888 is already indicated in Figures 1, 3, and 4. Two insertions of additional pulses caused by ritardandi are corrected well, and also the erroneous rests at the beginning are eliminated. The estimated time signature $6/8$ is perceptually similar to the notated time signature $12/8$.

Our second example is the Prelude No. 4 from Chopin's romantic Piano Preludes. Figure 5 shows two score excerpts together with the detected pulse candidates and the estimated measure structure as well as the corresponding stressgram and the short-time autocorrelation matrix for the whole piece. [2] Prelude No. 4 contains some long notes at downbeat positions leading to a good measure tracking result. Because of their strong presence, an eighth note pulse grid was detected, see Figure 5d. This piece of music starts with an upbeat of a quarter note. Since the MIDI format does not support upbeat information directly, our method adds enough additional pulses such that the first pulse lies in the congruence class of the downbeats as shown in Figure 5a.

Noticeable tempo changes together with short appoggiatura and a triplet around pulse No. 90 causes the PLP procedure to detect two additional pulses in consecutive measures (Fig. 5b). In the stressgram this is indicated by a jump of the salience path across two classes (Fig. 5c). Note that this error has only little influence on the estimation of the time signature (Fig. 5d). As indicated by the stressgram, two pulses are removed in this region during our post-processing. Although the deletion of a correctly detected pulse in the 2nd measure of Figure 5b leads to a wrong downbeat position in the subsequent measure, the global measure grid is restored in the 4th measure. This shows how our method optimizes the measure grid without having to correct each single pulse error.

3.2 Automatic Evaluation

Furthermore, we evaluated our method on score-like MIDI files which have been automatically disturbed by adding additional tempo change events. A similar approach was used in [8] to show that smooth tempo changes are detected well by the PLP method.

In particular, the goal of our procedure is to recognize measure positions correctly, therefore we use standard precision (P), recall (R), and F-measure (F) on the set of the MIDI notes. A note is considered relevant if it starts at a downbeat position in the S-MIDI file, and it is "retrieved" if it was mapped by our method to a downbeat position of the distorted MIDI file. Since no quantization step is included, we allow a tolerance of $\pm 5\%$ of each measure as its downbeat position.

By neglecting the musical time axis and using only the physical time position (in milliseconds) of all MIDI events, we simulate a performed MIDI of an S-MIDI file. The systematic distortion is done by adding a tempo change of $\pm 20\%$ around the normal tempo each 10 seconds.

As test set, we consider the Fifteen Fugues by Beethoven from IMSLP [3]. Here, the note durations are sufficient for a good estimation of the PLP pulses. Adding note velocity information from real performed MIDI files suggests an further improvement of the results.

[2] The examples are recordings on a MIDI piano taken from Saarland Music Data (`http://www.mpi-inf.mpg.de/resources/SMD/`), and the scores are picked from Mutopia (`http://www.mutopiaproject.org/`).

[3] Petrucci Music Library, `http://imslp.org/wiki/15_Fugues_(Beethoven,_Ludwig_van)`

Piece	Full method			PLP only			# Corrections		
	F	P	R	F	P	R	add	delete	upbeat
No. 1	0.477	0.587	0.402	0.372	0.509	0.293	0	2	0
No. 2	0.978	1	0.956	0.397	0.56	0.308	1	0	1
No. 3	0.656	0.663	0.649	0.144	0.196	0.113	1	0	0
No. 4	0.945	0.984	0.909	0.738	0.804	0.682	2	0	0
No. 5	0.966	0.971	0.962	0	0	0	0	0	2
No. 6	0.996	1	0.993	0.996	1	0.993	0	0	0
No. 7	0.826	0.832	0.82	0.324	0.386	0.28	1	4	1
No. 8	0.953	0.985	0.923	0.821	0.945	0.725	0	1	0
No. 9	0.896	0.916	0.876	0.787	0.855	0.73	1	0	1
No. 10	0.581	0.579	0.582	0.008	0.013	0.005	2	2	1
No. 11	1	1	1	1	1	1	0	0	0
No. 12	0.994	1	0.988	0.792	0.842	0.748	3	1	0
No. 13	0.656	0.884	0.522	0.245	0.393	0.178	0	2	2
No. 14	0.975	0.995	0.957	0.432	0.75	0.303	1	3	0
No. 15	0.692	0.98	0.535	0.633	0.992	0.465	0	0	4
Mean	0.839	0.892	0.805	0.513	0.616	0.455	0.8	1	0.8

Table 1. Evaluation results for 15 Fugues from Beethoven for full method and PLP-based beat tracking only

The results for the automatic evaluation are shown in Table 1. We evaluated both the original pulse sequence derived from the PLP pulse tracking method introduced in Section 2.1, and the post-processed version. In both cases, the detected time signature was used to locate the downbeats. For all pieces except No. 11, the annotated $2/2$ signature was mostly detected as $4/4$, sometimes as $8/4$. Compared to the results of the PLP pulse tracker, which is not designed for detecting downbeats, the results for some pieces were improved significantly by our method. For example, in Fugue No. 7 our post-processing method added one pulse and removed four other pulses. At the beginning, another single pulse was inserted to prevent upbeat shifts. These changes lead to an increase of the F-measure from 0.324 to 0.826, which has major consequences, e. g., on the amount of additional work for a human importing this MIDI file into a score notation software to optimize the score manually.

4. CONCLUSION

We presented a bottom-up method to derive a musically meaningful time axis for performed MIDI files, and converting them into semantically enriched score-like MIDI files. Our proposed procedure optimized an estimated pulse sequence by insertion of missing pulses as well as removal of spurious pulses to derive an overall consistent measure grid.

Since the output of the presented method is another MIDI file, our procedure can be used in combination with any MIDI quantization software by using it for preprocessing performed MIDI files having no musically meaningful time information. Essentially, the physical time axis remains unchanged, so it can be used further in combination with rhythm transcription approaches. Deriving a musical time axis without quantization is also meaningful for real-time interaction with MIDI synthesizers, e. g., as a variation of [3]. Because of its generality, our procedure can be simply extended by including other rhythmic or harmonic aspects.

Acknowledgments: This work has been supported by the German Research Foundation (DFG CL 64/8-1, DFG MU 2686/5-1). The International Audio Laboratories Erlangen are a joint institution of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and Fraunhofer IIS.

5. REFERENCES

[1] E. Benetos, S. Dixon, D. Giannoulis, H. Kirchhoff, and A. Klapuri. Automatic music transcription: challenges and future directions. *Journal of Intelligent Information Systems*, 41(3):407–434, 2013.

[2] E. Cambouropoulos. From MIDI to traditional musical notation. In *Proc. of the AAAI Workshop on Artificial Intelligence and Music*, vol. 30, 2000.

[3] R. B. Dannenberg and C. Raphael. Music score alignment and computer accompaniment. *Communications of the ACM, Special Issue: Music information retrieval*, 49(8):38–43, 2006.

[4] M. E. P. Davies and M. D. Plumbley. Context-dependent beat tracking of musical audio. *IEEE Transact. on Audio, Speech and Language Processing*, 15(3):1009–1020, 2007.

[5] S. Dixon. Automatic extraction of tempo and beat from expressive performances. *Journal of New Music Research*, 30:39–58, 2001.

[6] D. P. W. Ellis. Beat tracking by dynamic programming. *Journal of New Music Research*, 36(1):51–60, 2007.

[7] F. Gouyon and S. Dixon. A review of automatic rhythm description systems. *Computer Music Journal*, 29:34–54, 2005.

[8] P. Grosche and M. Müller. A mid-level representation for capturing dominant tempo and pulse information in music recordings. In *Proc. of the Intern. Conf. on Music Information Retrieval (ISMIR)*, pp. 189–194, 2009.

[9] P. Grosche and M. Müller. Extracting predominant local pulse information from music recordings. *IEEE Transact. on Audio, Speech, and Language Processing*, 19(6):1688–1701, 2011.

[10] D. M. Huber. *The MIDI manual*. Focal Press, 3rd edition, 2006.

[11] F. R. Moore. The dysfunctions of MIDI. *Computer Music Journal*, 12(1):19–28, 1988.

[12] C. Raphael. Automated rhythm transcription. In *Proc. of the Intern. Conf. on Music Information Retrieval (ISMIR)*, 2001.

[13] E. Selfridge-Field, editor. *Beyond MIDI: the handbook of musical codes*. MIT Press, Cambridge, MA, USA, 1997.

[14] H. Takeda, T. Nishimoto, and S. Sagayama. Rhythm and tempo analysis toward automatic music transcription. In *IEEE Intern. Conf. on Acoustics, Speech and Signal Processing*, vol. 4, pp. IV–1317, 2007.

[15] A. C. Yang, E. Chew, and A. Volk. A dynamic programming approach to adaptive tatum assignment for rhythm transcription. In *IEEE Intern. Symposium on Multimedia*, 2005.

ESTIMATION OF THE DIRECTION OF STROKES AND ARPEGGIOS

Isabel Barbancho[1], George Tzanetakis[2], Lorenzo J. Tardón[1], Peter F. Driessen[2], Ana M. Barbancho[1]
[1]Universidad de Málaga, ATIC Research Group, ETSI Telecomunicación,
Dpt. Ingeniería de Comunicaciones, 29071 Málaga, Spain
[2] University of Victoria, Department of Computer Science, Victoria, Canada
ibp@ic.uma.es, gtzan@cs.uvic.ca, lorenzo@ic.uma.es,
peter@ece.uvic.ca, abp@ic.uma.es

ABSTRACT

Whenever a chord is played in a musical instrument, the notes are not commonly played at the same time. Actually, in some instruments, it is impossible to trigger multiple notes simultaneously. In others, the player can consciously select the order of the sequence of notes to play to create a chord. In either case, the notes in the chord can be played very fast, and they can be played from the lowest to the highest pitch note (upstroke) or from the highest to the lowest pitch note (downstroke).

In this paper, we describe a system to automatically estimate the direction of strokes and arpeggios from audio recordings. The proposed system is based on the analysis of the spectrogram to identify meaningful changes. In addition to the estimation of the up or down stroke direction, the proposed method provides information about the number of notes that constitute the chord, as well as the chord playing speed. The system has been tested with four different instruments: guitar, piano, autoharp and organ.

1. INTRODUCTION

The design and development of music transcription systems has been an open research topic since the first attempts made by Moorer in 1977 [15]. Since then, many authors have worked in different aspects of the transcription problem [12], [17]. A common task in this context is automatic chord transcription [13], [1], [3], [7], [14], but also, other aspects beyond the mere detection of the notes played are nowadays considered, shifting the focus of the research to different pieces of information related to the way in which these notes are played, i.e. musical expressiveness [18], [4], [7], [11].

A chord can be defined as a specific set of notes that sound at the same time. Often, when a chord is played, not all the notes in the chord start at the same time. Because of the mechanics of actuation of some instruments like the guitar, the mandolin, and the autoharp [20], it is hard to excite different strings at the same time. Instead the performer typically actuates them sequentially in a stroke. A stroke is a single motion across the strings of the instrument. The stroke can have two different directions: UP, when the hand moves from the lowest to the highest note, and DOWN, when the hand moves from the highest to the lowest note. A strum is made up of several strokes combined in a rhythmic pattern. In other instruments like the piano or the organ, all the notes that belong to a certain chord can be played at the same time. However, the musician can still choose to play the chord in arpeggio mode, i.e., one note after another. Again, the arpeggio direction can be up or down.

In this paper, we propose a new chord related analysis task focused on the identification of the stroke or arpeggio direction (up or down) in chords. Because the movement can be fast it is not feasible to look for onsets [6] to detect each note individually. Therefore, a different approach will be proposed. In addition to the detection of the stroke direction, our proposed method also detects the speed with which the chord has been played as well as the number of notes. The estimation of the number of notes played in a chord is a problem that has not been typically addressed, although some references can be found related to the estimation of the number of instruments in polyphonic music [16], which constitutes a related but different problem. Regarding the chord playing speed, to the best our knowledge there are no published works to identify this parameter except when specific hardware is used for the task [19], [9]. The paper is organized as follows: in Section 2, the proposed system model is explained. Section 3 presents some experimental results and Section 4 draws some conclusions.

2. STROKE AND ARPEGGIO ANALYSIS

The main goal of this work is the analysis of audio excerpts to detect if a chord has been played from lower to higher notes (UP) or vice versa (DOWN). The movement to play a chord may be quite fast and all the information about the movement is contained at the very beginning of the chord waveform. After all the strings of the chord have been played, it is no longer possible to know whether the

movement was up or down because the resulting sound contains all the component pitches. This means that any feature that may provide information about how the spectrum varies when the chord is being played has to be calculated at the very beginning of the chord. We will consider that the time needed to complete a stroke varies from 1 s (relatively slow) to less than 0.2 s, when the chord is played fast.

Let x denote the samples of the played chord under study. In order to calculate a spectrogram, the samples x are divided into segments $x_m = [x_m[1], ..., x_m[M]]^T$, where M is the selected window size for the spectrogram calculation. Let PSD_m denote the Power Spectral Density of each segment x_m and L_m the logarithm of the PSD_m i.e $L_m = 10\log_{10}(PSD_m)$. In Fig. 1, the log spectrogram of an 'F Major' guitar chord played from the lowest to the highest string is shown (up stroke). The exact fret position employed to play this chord is $frets = [2, 2, 3, 4, 4, 2]$ where the vector $frets$ represents the frets pressed to play the chord from string 1 (highest string) to string 6 (lowest string). This chord has been generated synthetically to control the exact delay between each note in the chord (in this case the delay is $\tau = 4ms$). The guitar samples have been extracted from the RWC database [10]. As it can be observed in Fig. 1, the information about the stroke direction is not directly visible in the spectrogram. Therefore, in order to detect the stroke direction, the spectrogram needs to be further analysed.

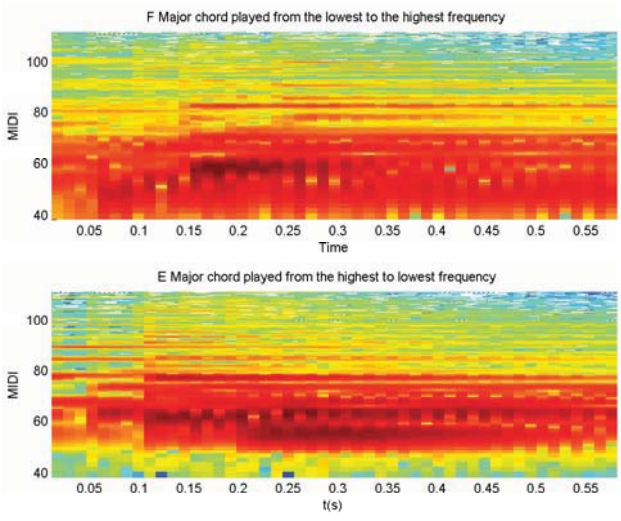


Figure 1. Spectrogram of an F Major chord UP stroke in a classic guitar (upper figure) and an E Major chord DOWN stroke. Audio file is sampled with $f_s = 44100$ Hz. For the spectrogram, the selected parameters are window size $M = 1024$, $overlapp = 512$ with a Hamming window. The DFT size is $K = 4096$. For convenience, the MIDI numbers are shown in the y-axis instead of the frequency bins: MIDI $= 69 + 12\log_2(f/440)$.

2.1 Detection of new spectral components

Whenever a new note is played, it is expected that new spectral components corresponding to the new note will be added to the existing components of the previous note (if any). In auditory scene analysis [8] this is termed the 'old+new heuristic'. The main idea is to take advantage of this heuristic by detecting whether the current spectrum contains new components or, conversely, whether it simply retains the components from the previous spectrum. As we are frequently dealing with sounds that decay quickly our model of sustained notes will also contain a decay component. In order to detect 'old+new' changes we minimize the following equation:

$$\epsilon[m] = \min_{\alpha[m]} \left[\sum_{k=1}^{K} |L_m[k] - \alpha[m] L_{m-1}[k]| \right] \quad (1)$$

The goal is to find a local $\alpha[m]$ (decay factor) that minimizes $\epsilon[m]$ for two consecutive windows m and $m-1$. The minimization is carried out by means of the unconstrained nonlinear minimization Nelder-Mead method [21]. The idea is to remove from window m all the spectral components that were also present in window $m-1$ with a gain adjustment so that any new spectral component becomes more clearly visible. Thus, if there are no new played notes in window m with respect to window $m-1$, $\epsilon[m]$ will be small, otherwise $\epsilon[m]$ will become larger because of the presence of the new note.

In Fig. 2 (a) and (b), the normalized evolutions of $\alpha[m]$ and $\epsilon[m]$ respectively are displayed for the F Major UP chord shown in Fig.1 (a). The vertical lines represent the instants when new notes appear in the chord. When a new note is played in the chord, $\alpha[m]$ attains a minimum and $\epsilon[m]$ a maximum. In order to automatically detect the instants when the new notes appear, the following variables are defined:

$$\epsilon'[m] = \begin{cases} \epsilon[m] - \epsilon[m-1] & \text{if } \epsilon[m] - \epsilon[m-1] > 0.5 \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

$$\alpha'[m] = \alpha[m] - \alpha[m-1] \quad (3)$$

$$f_{\alpha,\epsilon}[m] = \frac{\epsilon'[m]}{max(\epsilon')} \cdot \left| \frac{\alpha'[m]}{max(\alpha')} \right| \quad (4)$$

Fig. 2 (c) shows the behaviour of $f_{\alpha,\epsilon}$, where if becomes easy to identify the presence of new notes. In addition, it is also possible to estimate the number of notes played in the chord (in this case 6), as well as the stroke speed.

2.2 Estimation of number of notes and stroke speed

After a measure that highlights the presence of new notes has been defined, the next step is to find the peaks of $f_{\alpha,\epsilon}$. Each sample of the function $f_{\alpha,\epsilon}(m)$ is compared against $f_{\alpha,\epsilon}(m-1)$ and $f_{\alpha,\epsilon}(m+1)$. If $f_{\alpha,\epsilon}(m)$ is larger than both neighbors (local maximum) and $f_{\alpha,\epsilon}(m) > 0.1$, then a candidate local peak is detected. Finally, if there are two

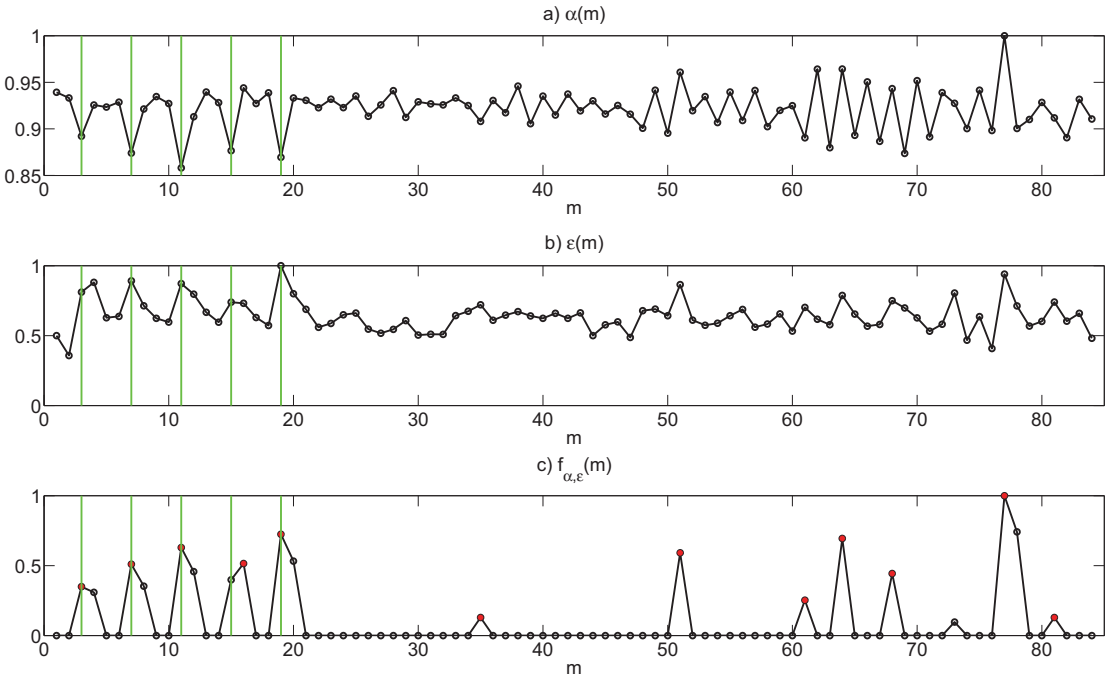


Figure 2. F Major chord UP stroke in classic guitar: (a) Evolution of $\alpha[m]$ that minimizes equation (1) , (b) Evolution of the error $\epsilon[m]$ as defined in equation (1) and (c) Evolution of $f_{\alpha,\epsilon}[m]$ (equation (4)) where the presence of new notes becomes apparent.

peaks less than two points apart, the smallest one is not considered. Once these selected peaks have been localized, the final step is to determine which ones belong to played notes so that the number of played notes can be estimated together with the speed of the stroke. The key observation is that the time difference between the note onsets that belong to the same stroke or arpeggio will be approximately constant. The reason is that, because of human physiology, in most cases the stroke is performed with fixed speed.

Let f_{locs} stand for a function that contains the positions where the selected peaks of $f_{\alpha,\epsilon}$ are located. The objective is to detect sets of approximately equispaced peaks which will correspond to the played notes in a chord or arpeggio. Then, the number of played notes NPN_e will be estimated as follows:

$$\text{NPN}_e = n_{neq} + 2 \tag{5}$$

where n_{neq} represents the minimum value of n such that the distance between the positions of peaks contained in f_{locs} is no longer kept approximately constant. n_{neq} is defined as:

$$n_{neq} = \underset{n}{\text{argmin}} \left(|f''_{locs}(n)| > 3 \right) \tag{6}$$

where $f''_{locs}(n)$ stands for the second order difference of $f_{locs}(n)$.

Finally, the stroke speed estimate in notes per second is given by:

$$V = \frac{f_{locs}(\text{NPN}_e - 3) \cdot (windowsize - overlapp)}{f_s \cdot \text{NPN}} \tag{7}$$

Once the location of every new note is estimated using the method described, the feature to detected the stroke direction is computed.

2.3 Feature to detect stroke direction

In Fig. 3, the details of the windows in which the spectral changes occur are depicted for the two guitar chords that are being analysed. The stroke direction can be guessed from those figures, but we still need to derive a meaningful computational feature that can be used for automatic classification.

In order to reduce the amount of information to be processed by the classifier that will decide the stroke direction, a meaningful feature must be considered. Thus, the spectral centroid in each of the windows in which the changes have been detected is calculated.

The spectral centroid is the centre of gravity of the spectrum itself [22], [24] and, in our case, it is estimated in each of the windows x_m where the change has been detected. This feature will be denoted SPC_m (Spectral Centroid of window m) and it is calculated as follows:

$$SPC_m = \left(\sum_{k=1}^{K} f_m(k) PSD_m(k) \Big/ \sum_{k=1}^{K} PSD_m(k) \right) \tag{8}$$

where PSD_m is the power spectral density of the window x_m and f_m is the corresponding frequency vector.

Note that we will use SPCs-KD when the SPCs are estimated with the delays known beforehand and SPCs-ED when the delays are estimated according to the procedure described in section 2.1.

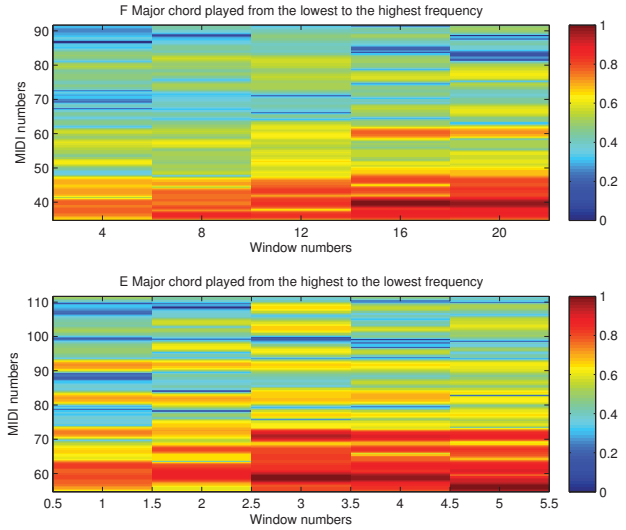


Figure 3. Windows of the spectrogram of the UP F Major chord and the DOWN E Major chord in which new notes appear.

Fig. 4 illustrates the behaviour of the SPC in the selected windows in which a change of the spectral content is detected for the UP F Major chord and the DOWN E Major chord shown in the previous illustrations.

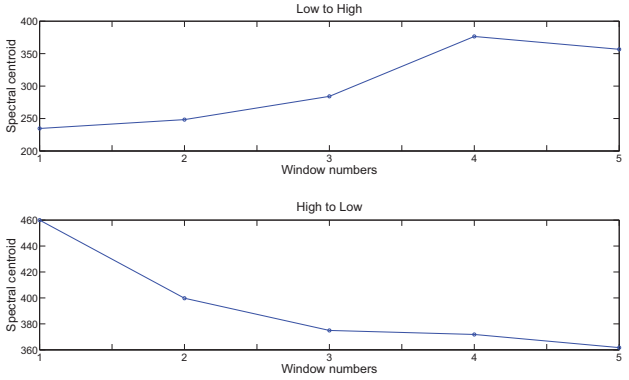


Figure 4. Spectral centroid evolution for the UP F Major chord and the DOWN E Major chord in the windows of the spectrogram in which the changes happen.

3. CLASSIFICATION RESULTS OF UP AND DOWN STROKES

The proposed scheme has been tested with four different instruments: guitar, piano, organ and autoharp. The guitar and organ samples have been extracted from the RWC database [10], the piano recordings have been extracted from [2] and the autoharp recordings have been specifically made by the research team.

A subset of the chords used in the experiment contains chords artificially assembled so that all the information regarding the number of notes played and the delay is available for assessing the proposed system performance. All audio file are sampled with $f_s = 44100$ Hz. The delay between consecutive notes in a chord ranges between 1000 samples (11 ms) and 5000 samples (55 ms).

With the guitar and the autoharp, the natural way of playing a chord is by adding the sound of one string after another. The guitar is a well known instrument [5], but the autoharp is not. The autoharp is an American instrument invented in 1881 by Charles F. Zimmerman. It was very popular in Canada and the USA for teaching music fundamentals because it is easy to play and introduces in a very intuitive way harmony concepts. Briefly, the instrument has 36 strings and the musician can select which ones can vibrate by pressing buttons corresponding to different chords. The buttons in the autoharp mute the strings corresponding to the notes that do not belong to the chord to be played. Then, the musician actuates the strings by strumming with the other hand. In the guitar, the decay of each string is exponential and very fast. In the case of the autoharp, due to the resonance box, the decay of the sound is slower. In the piano and in the organ, the musician can play the chords arpeggiated. In the piano the decay is also exponential but in the organ the sound of a note is sustained and decays slowly.

In Tables 1 and 1, the UP and DOWN classification results are summarized for the artificially assembled chords. In all the cases, 100 chords have been used for training (50 UP and 50 DOWN) and a total of 500 chords equally distributed among UP and DOWN have been used to evaluate the classification performance. The chord recordings used for training and testing purposes are separate and different.

The performance of the proposed feature is compared against a baseline that makes use of MFCCs (Mel Frequency Cepstral Coefficients) calculated as explained in [22]. More specifically, 15 coefficients are considered with the first one, corresponding to the DC component, removed.

A Fisher Linear Discriminant and a linear Support Vector Machine (SVM) classifier [23] have been evaluated.

Looking at Tables 1 and 2, we observe that the results of the proposed method and feature are satisfactory. In almost all the cases, the performance of the proposed scheme is better than the one achieved by the baseline based on MFCCs.

The error in the determination of the number of played notes is estimated as follows:

$$Error_{NPN} = A\left(\frac{|NPN_e - NPN_r|}{NPN_r}\right) \cdot 100 \quad (9)$$

where $A()$ is the averaging operator, NPN_e stands for the estimated Number of Played Notes in (5) and NPN$_r$ represents the the actual number of notes.

The error in the estimated delay between consecutive notes is evaluated as follows:

$$Error_W = A\left(\frac{|W_e - W_r|}{NPN_e \cdot W_d}\right) \cdot 100 \quad (10)$$

where W_e represents the windows in which a significant spectral change has been found, W_r stands for the windows

Instrument	stroke	Fisher		
		SPCs-KD	SPCs-ED	MFCCs
Guitar	up	93.88	78.88	72.22
	down	96.11	97.22	60.55
	overall	95.00	88.05	66.38
Piano	up	91.95	79.31	77.85
	down	97.81	84.36	81.42
	overall	94.88	81.83	79.64
Organ	up	90.00	89.16	78.33
	down	90.00	86.66	56.66
	overall	90.00	87.91	67.50
Autoharp	up	100	94.44	97.91
	down	100	86.80	79.86
	overall	100	90.62	88.88

Table 1. Success Rate (%) of UP and DOWN stroke classification using a Fisher linear classifier [23]. The features used by the classifier are: SPCs-KD (Spectral Centroid of selected Windows with known-delay), SPCs-ED (Spectral Centroid of selected Windows with estimated delay) and MFCCs (15 Mel Frequency Cepstral Coefficients).

Instrument	stroke	SVM		
		SPCs-KD	SPCs-ED	MFCCs
Guitar	up	91.57	87.77	58.44
	down	95.01	95.55	98.78
	overall	93.29	91.66	78.61
Piano	up	90.12	81.22	77.25
	down	96.45	82.84	83.63
	overall	93.28	82.03	80.44
Organ	up	89.16	90.52	90.83
	down	88.66	87.98	51.66
	overall	88.91	89.25	71.25
Autoharp	up	99.30	90.97	91.27
	down	97.91	95.14	90.89
	overall	98.61	93.05	91.08

Table 2. Success Rate (%) of UP and DOWN stroke classification using a linear SVM classifier [23]. The features used by the classifier are: SPCs-KD (Spectral Centroid of selected Windows with known-delay), SPCs-ED (Spectral Centroid of selected Windows with estimated delay) and MFCCs (15 Mel Frequency Cepstral Coefficients).

where the changes actually happen and W_d is number of windows between two consecutive W_r windows. Table 3 shows the obtained results.

The proposed method for the estimation of the number of notes and delays can be improved. This is a first approach to solve this problem. Our main goal has been to detect the up or down stroke direction which is useful to complete the transcription of the performance of certain instruments, specifically the autoharp. The performance attained in the detection of the stroke direction is satisfactory according to the results shown.

It is important to note, that even though $Error_W$ seems to be quite high, this error is in most of cases positive, i.e., the change is detected one or two windows after the first window that actually contains the change. This issue is not critical for the feature employed by the classifier because it is possible to observe the difference in the estimation of the SPC_m in (8).

Instrument	stroke	$Error_{NPN}$	$Error_W$
Guitar	up	37.65	10.49
	down	33.33	15.92
	overall	35.49	13.20
Piano	up	30.72	28.38
	down	33.65	18.10
	overall	32.18	23.24
Organ	up	24.54	29.72
	down	36.52	26.12
	overall	30.53	27.92
Autoharp	up	53.06	10.46
	down	42.88	13.96
	overall	47.97	12.21

Table 3. Error (%) in the estimation of the number of notes played and in the estimation of the delay between consecutive played notes in chords.

Instrument	stroke	Fisher	
		SPCs-ED	MFCCs
Autoharp	up	65.21	43.47
	down	86.44	94.91
	overall	75.10	69.19
		SVM	
		SPCs-ED	MFCCs
	up	73.77	62.84
	down	89.83	81.52
	overall	77.52	72.18

Table 4. Success Rate (%) of UP and DOWN stroke classification for real autoharp chords.

Finally, Table 4 presents the results obtained for real chords played in an autoharp. We have used 100 chords for training and 230 chords for testing. The 330 autoharp chords recorded are equaly distributed between UP and DOWN chords and in different tessituras. It can be observed that the proposed feature outperforms the baseline proposed based on the usage of MFCCs.

4. CONCLUSIONS

In this paper, a new feature to detect the up or down direction of strokes and arpeggios has been presented. The developed method also provides information about the number of played notes and the stroke speed.

The system have been tested with four different instruments: classic guitar, piano, autoharp and organ and it has been shown how the new proposed feature outperforms the baseline defined for this task. The baseline makes use of the well known MFCCs as classification features so that the baseline scheme can be easily replicated. The Matlab files used to generate the data-set for piano, guitar and organ, the audio files of the autoharp and the ground truth are available upon request for reproducible research.

5. ACKNOWLEDGMENTS

This work has been funded by the Ministerio de Economía y Competitividad of the Spanish Government under Project No. TIN2013-47276-C6-2-R, by the Junta de Andalucía under Project No. P11-TIC-7154 and by the Ministerio de Educación, Cultura y Deporte through the Programa Nacional de Movilidad de Recursos Humanos del Plan Nacional de I-D+i 2008- 2011. Universidad de Málaga. Campus de Excelencia Internacional Andalucía Tech.

6. REFERENCES

[1] A. M. Barbancho, I. Barbancho, B. Soto, and L.J. Tardón. Transcription of piano recordings. *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 377–380, 2011.

[2] A. M. Barbancho, I. Barbancho, L. J. Tardón, and E. Molina. *Database of Piano Chords. An Engineering View of Harmony*. SpringerBriefs in Electrical and Computer Engineering, 2013.

[3] A. M. Barbancho, A. Klapuri, L.J. Tardón, and I. Barbancho. Automatic transcription of guitar chords and fingering from audio. *IEEE Transactions on Audio, Speech, and Language Processing*, 20:915–921, March 2012.

[4] I. Barbancho, C. de la Bandera, A. M. Barbancho, and L. J. Tardón. Transcription and expressiveness detection system for violin music. *Proceedings of the IEEE conference on Acoustics, Speech, and Signal Proc. (ICASSP)*, pages 189–192, March 2009.

[5] I. Barbancho, L.J Tardon, S. Sammartino, and A.M. Barbancho. Inharmonicity-based method for the automatic generation of guitar tablature. *Audio, Speech, and Language Processing, IEEE Transactions on*, 20(6):1857–1868, 2012.

[6] J.P. Bello, L. Daudet, and M.B. Sandler. A tutorial on onset detection in music signals. *IEEE Trans. on Audio, Speech and Language Processing*, 14:1035–1047, September 2005.

[7] E. Benetos, S. Dixon, D. Giannoulis, and H. Kirchhoff. Automatic music transcription: Breaking the glass ceiling. *ISMIR*, 2012.

[8] Albert S Bregman. *Auditory scene analysis: The perceptual organization of sound*. MIT press, 1994.

[9] D. Chadefaux, J.-L. Le Carrou, B. Fabre, and L. Daudet. Experimentally based description of harp plucking. *The Journal of the Acoustical Society of America*, 131:844, 2012.

[10] M. Goto. Development of the RWC music database. In *18th Int. Con. on Acoustics*, volume I, pages 553–556, 2004.

[11] A. Kirke and E. R. Miranda. An overview of computer systems for expressive music performance. In *Guide to Computing for Expressive Music Performance*, pages 1–47. Springer, 2013.

[12] A. Klapuri and T. Virtanen. Automatic music transcription. In *Handbook of Signal Processing in Acoustics*, pages 277–303. Springer, 2009.

[13] A.P. Klapuri. Multiple fundamental frequency estimation based on harmonicity and spectral smoothness. *IEEE Trans. on Speech and Audio Processing*, 11:804–816, Nov. 2003.

[14] K. Lee and M. Slaney. Acoustic chord transcription and key extraction from audio using key-dependent hmms trained on synthesized audio. *Audio, Speech, and Language Processing, IEEE Transactions on*, 16(2):291–301, 2008.

[15] J. A. Moorer. On the transcription of musical sound by computer. *Computer Music Journal*, pages 32–38, 1977.

[16] M.Schoeffler, F.R. Stoter, H.Bayerlein, B.Edler, and J.Herre. An experiment about estimating the number of instruments in polyphonic music: a comparison between internet and laboratory results. *ISMIR*, 2013.

[17] M. Müller, D. P. W. Ellis, A. Klapuri, and G. Richard. Signal processing for music analysis. *IEEE Journal of Selected Topics in Signal Processing*, 5:1088–1110, October 2011.

[18] T. H. Ozaslan, E Guaus, E. Palacios, and J. L. Arcos. Identifying attack articulations in classical guitar. In *Computer Music Modeling and Retrieval. Exploring Music Contents. Lecture Notes in Computer Science*, pages 219–241. Springer-Verlag, 2011.

[19] J.A. Paradiso, L.S Pardue, K.-Y. Hsiao, and A.Y. Benbasat. Electromagnetic tagging for electronic music interfaces. *Journal of New Music Research*, 32(4):395–409, 2003.

[20] M. Peterson. *Mel Bays Complete Method for Autoharp or Chromaharp*. Mel Bay Publications, 1979.

[21] W.H. Press, S. A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes: The Art of Scientific Computing. Third Edition*. Cambridge University Press, 2007.

[22] L.J. Tardón, S.Sammartino, and I.Barbancho. Design of an efficient music-speech discriminator. *Journal of the Acoustical Society of America*, 1:271–279, January 2010.

[23] S. Theodoridis and K. Koutroumbas. *Pattern Recognition, 4th Edition*. Academic Press, 2008.

[24] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Trans. on Audio, Speech and Language Processing*, 10:293–302, 2002.

PREDICTING EXPRESSIVE DYNAMICS IN PIANO PERFORMANCES USING NEURAL NETWORKS

Sam van Herwaarden
Austrian Research Institute for AI
samvherwaarden@gmail.com

Maarten Grachten
Austrian Research Institute for AI
http://www.ofai.at/~maarten.grachten

W. Bas de Haas
Utrecht University
w.b.dehaas@uu.nl

ABSTRACT

This paper presents a model for predicting expressive accentuation in piano performances with neural networks. Using Restricted Boltzmann Machines (RBMs), features are learned from performance data, after which these features are used to predict performed loudness. During feature learning, data describing more than 6000 musical pieces is used; when training for prediction, two datasets are used, both recorded on a Bösendorfer piano (accurately measuring note on- and offset times and velocity values), but describing different compositions performed by different pianists. The resulting model is tested by predicting note velocity for unseen performances. Our approach differs from earlier work in a number of ways: (1) an additional input representation based on a local history of velocity values is used, (2) the RBMs are trained to result in a network with sparse activations, (3) network connectivity is increased by adding skip-connections, and (4) more data is used for training. These modifications result in a network performing better than the state-of-the-art on the same data and more descriptive features, which can be used for rendering performances, or for gaining insight into which aspects of a musical piece influence its performance.

1. INTRODUCTION

Music is not performed exactly the way it is described in score: a performance in which notes occur on a regular temporal grid and all notes are played equally loud is often considered dull. Depending on the instrument, performers have different parameters they use for modulating expression in their music [14]: time (timing, tempo), pitch, loudness and timbre. For some of these parameters composers add annotations to musical score describing how they should be varied, but for a large part performers are expected render the score according to tacit knowledge, and personal judgment. This allows performers to imbue on a performance their personal style, but this is not to say that music performance is arbitrary—it is often clear which interpretations are (not) musically appropriate.

This article describes a number of modifications to the method for modeling expressive dynamics proposed by Grachten & Krebs [7], and is based on the MSc thesis work described in [17]. We show that, with an additional input representation and a different set-up of the machine learning approach, we achieve a statistically significant improvement on the prediction accuracy achieved in [7], with more descriptive features. Our achieved performance is also comparable with the work in [8]. In the following sections we first summarize previous work in this area, followed by an overview of the used machine learning architecture. We then describe the experiments, the results and the relevance of the findings.

2. PREVIOUS WORK

Two important aspects of music that affect the way it is to be performed are the musical structure, and the emotion that the performance should convey [13]. The last decades different methods for analyzing the structural properties of a piece of music have been proposed (e.g. [12, 15]), where the analysis tends to stress the relationship between structure on a local level (elements of pitch and rhythm) and their effect on the melodic expectancy of a listener. Emotional charge conveyed by a piece is more abstract and variable: trained musicians can play the same piece conveying different emotions, and in fact these emotions can be identified by listeners [5].

Because musical structure can be studied through inspection of the musical score, computational models of musical expression tend to focus on this. A number of different computational models of expression have been developed earlier, studying different expressive parameters (e.g. [1, 4]). Many models are rule-based, where the rules describing how expression should be applied are often hand-designed. Other models still focus on rules, but automatically extract them from performance data (e.g. [11, 18]). A performance model can also be based on the score annotations for the relevant parameter provided by the composer, as in [8] which uses information on note pitch, loudness annotations and other hand-crafted features.

Some recent studies model regularities in musical sequences using unsupervised techniques [2, 16], in the context of musical sequence prediction. Grachten & Krebs [7] apply unsupervised learning techniques to learn features from a simple input representation based on a piano roll

representation of the symbolic score, in the context of predicting musical expression. The resulting learned features then describe common patterns occurring in the input data, which can be related to concepts from music theory and used for prediction of expressive dynamics. By using a simple input representation and network, the model remains relatively transparent with regard to its inner workings. It is shown that Restricted Boltzmann Machines (RBMs) learn the most effective model, and in this paper, we build on that approach.

An RBM is a type of artificial neural network, particularly suitable for unsupervised learning from binary input data. During training it learns a set of features that can efficiently encode the input data. The features are used to transform the input data non-linearly, which can be useful for further (supervised) learning. For a detailed explanation of RBMs the reader is referred to for example [9, 10].

3. ARCHITECTURE

Figure 1 illustrates the setup of the network we use. As input the network sees the music data in two different representations: the score-based note-centered representation first developed by [7] and the new loudness-based velocity-history representation. The input data is transformed through a series of hidden unit activations (RBM feature activations) in $\mathbf{L_1}$, $\mathbf{L_2}$ and $\mathbf{L_3}$. These feature activations are then used to estimate the output (normalized velocity). As is typical with neural networks, the model is blind to the meaningful ordering of the input nodes (we could change the ordering without affecting the results).

The set-up is different from that in [7] in a number of ways: (1) an additional input representation based on a local history of velocity values is used, (2) the RBMs are trained for sparse activations, (3) network connectivity is increased with skip-connections (i.e. $\mathbf{w_1}$ and $\mathbf{w_2}$ in Figure 1 can be used simultaneously), and (4) more data is used for training. The following sections cover these changes in more detail. First, we describe the data available for developing the model. We then describe the way these data are presented to our model as input and output, and finally the process of training and evaluating our model.

3.1 Available data

Data from a number of sources is used for the experiments in this paper. We have *score* data, which describes musical score in a piano-roll fashion, and we have *performance* data, based on recordings from a computer-controlled Bösendorfer piano. For the performance data, accurate note on- and offsets are available as well as velocity values, and these values have been linked to corresponding score data. For all available performance data, score data is also available, the converse does not hold.

A number of (MIDI) score datasets is used: the JSB Chorales, [1] some MuseScore pieces, [2] the Mutopia database, [3] the Nottingham database, [4] the Piano-midi archive [5] and the Voluntocracy dataset [6]. These datasets are used during unsupervised learning with the note-centered representation only. The performance datasets we use have been developed at the Austrian Research Institute for AI (OFAI). One dataset contains performance data of all Chopin's piano music played by Nikita Magaloff [3] (~ 300.000 notes in 155 pieces), the other contains all Mozart piano sonatas, performed by Roland Batik [18] (~ 100.000 notes in 128 pieces). These datasets have been used both for unsupervised and supervised learning.

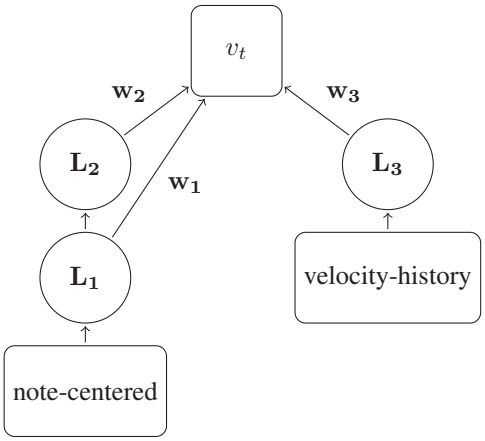


Figure 1: The used architecture. The rounded squares correspond to in- and outputs, the circles to layers of hidden units trained as Restricted Boltzmann Machines. $\mathbf{w_1}$ through $\mathbf{w_3}$ are the weights used to predict v_t based on the hidden unit activations in hidden layers $\mathbf{L_1}$ through $\mathbf{L_3}$. $\mathbf{w_1}$ through $\mathbf{w_3}$ are determined with a least-squares fit.

3.2 Note-centered representation

Score data is input into the network in one form in the *note-centered representation*, which is based on a piano-roll representation. For every note in a musical score, an input sample is generated with this note in the center, as illustrated in Figure 2b. The horizontal axis corresponds to score time and covers a span of 3 beats before the onset of the central note to 3 beats after the onset. Each beat is further divided into 8 equal units of time (effectively each column in the input corresponds to a 32nd note), and longer notes are wider. The vertical axis corresponds to relative pitch compared to the central note, and covers a span of -55 to $+55$ semi-tones. To allow the representation to distinguish between separate notes of the same pitch played consecutively, and a single long note at that pitch, note durations are represented as their score duration minus 32nd note duration (this was also done in [7]).

This approach is the same as the *duration coding* approach used in [7] with two exceptions: they experimented with time-spans of 1, 2 and 4 beats (with very small dif-

[1] www.jsbchorales.net
[2] www.musescore.org
[3] www.mutopiaproject.org
[4] www.chezfred.org.uk/University/music/database.htm
[5] www.piano-midi.de
[6] www.voluntocracy.org

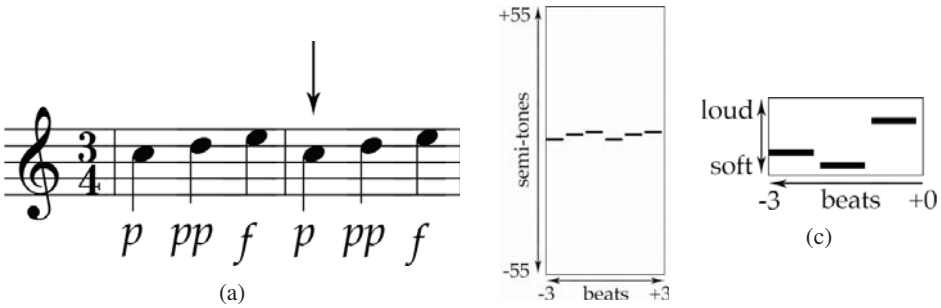


Figure 2: A short piece of score, and resulting network input for the note indicated by the arrow: (a) shows the score, where the annotations should be interpreted as *performed* loudness, not as annotated loudness directives, (b) shows the note-centered representation and (c) the velocity-history representation.

ferences in results between the 2 and 4 beats experiments), and used a pitch range of -87 to $+87$ semi-tones (so that always the entire piano keyboard is covered). In practice, the large pitch range is likely unnecessary and only increases the length of the network input vector (note combinations with such intervals are very rare and do not noticeably affect the learned features).

This choice of representation makes our system insensitive to absolute pitch: if all input notes are transposed by a few semi-tones in the same direction, the generated input samples will be identical. This also allows the system to learn about harmony based on relative pitch: for example certain chords will typically be represented in the same way regardless of their root tone. No additional information on absolute note pitch was included, to keep the model simple.

3.3 Velocity-history representation

When analyzing expressive parameters in existing performances, it is interesting to not only take into account direct harmonic and rhythmic structure around a note as is done with the note-centered representation, but also effects in continuity of musical phrases: for example, in many cases note loudness increases or decreases gradually over a number of notes. The precise accentuation of a note is than affected by the accentuation of preceding notes.

Our *velocity-history representation* is designed to encode this kind of information. Figure 2c illustrates this representation. Conceptually, it is similar to the note-centered representation, with a few differences: the vertical axis now represents relative velocity (normalized with respect to the mean μ and standard deviation σ of the velocity in a piece, where the range from $\mu - 2\sigma$ to $\mu + 2\sigma$ is quantized into 12 discrete values), and the horizontal axis corresponds to the time preceding the current note (ranging from note onset -3 beats to note onset $+0$).

The velocity-history representation uses information from an actual performance during prediction. In a sense, the system is asked to predict the continuation of a musical phrase: given that the last notes were played in a certain way, how will the next note be played? When using this representation, experiments with our model aim to *explain* how a note is performed in an existing performance, rather than *predict* it for a new piece of bare score (an actual performance needs to be available).

3.4 Velocity normalization

Since we use semi-supervised learning, at some point we need target values accompanying our input representations. We have exactly one sample for each note, and we are studying dynamics, so the logical parameter to base these target values on is note velocity. However, the different pieces described in our data have fairly diverse characteristics when it comes to dynamics. Some pieces are performed louder on average, or have stronger variations in dynamics. In this study we have chosen to focus on local effects within a single piece, and not so much on differences between pieces. For this reason we normalize our velocity target values so they have zero-mean and unit standard-deviation within a piece (we use these values both for supervised learning and for generating the velocity-history representation). This is slightly different from the normalization used in [7], where normalization was only used to obtain zero-mean within a piece.

3.5 Training and evaluation

The process of developing and testing the network can be separated into three phases: unsupervised learning, supervised learning and performance evaluation. We will now describe these in more detail.

3.5.1 Unsupervised learning

During unsupervised learning, we train only hidden layers $\mathbf{L_1}$ through $\mathbf{L_3}$. The layers are trained as RBMs on the full set of score data in the note-centered and velocity-history representations, where $\mathbf{L_1}$ and $\mathbf{L_3}$ are trained on the input representations directly, and $\mathbf{L_2}$ is trained on the feature activations in $\mathbf{L_1}$.

In the note-centered representation samples consist of 5280 binary input values. $\mathbf{L_1}$ is trained with 512 hidden units (ensuring a significant bottleneck in the network), and $\mathbf{L_2}$ contains fewer hidden units again: 200 units. In the velocity-history representation samples consist of 288 input values, these are encoded in 120 hidden units in $\mathbf{L_3}$.

We enforce sparse coding in the network, using the method proposed in [6], which allows us to not only control the average activation of hidden units in the network, but also the actual distribution of activations: we can force the RBM to represent each sample as a number of highly active features, improving inspectability.

3.5.2 Supervised learning

For supervised learning we use a simple approach: given the transformation of an input sample by $\mathbf{L_1}$ to $\mathbf{L_3}$, we fit the hidden unit activations in these layers to the corresponding v_t (normalized velocity) values using least-squares. Exploratory experiments suggested that more advanced techniques do not yield much better results. Thus, $\mathbf{w_1}$ through $\mathbf{w_3}$ simply define a linear transformation from the features activations to a prediction of the normalized velocity.

3.5.3 Performance evaluation

To evaluate the performance of our model we use a leave-one-out approach: we cycle through all the pieces in the performance data, where every time a particular piece is left out during supervised learning, after which the trained network is used to predict the expressive dynamics of the left-out piece. The quality of the prediction is then quantified using the R^2 measure (coefficient of determination). As mentioned before, the full set of data is used during unsupervised learning – because the objective function optimized during this phase has no relation to the velocity targets, we believe that this is an acceptable approach. As the final score after cycling through the whole dataset in this fashion, we use the weighted average R^2, where the number of notes in a piece is used as its weight.

4. EXPERIMENTS

In our experiments we vary two parameters: network connectivity, and training/testing datasets. Other experiments were also done but are not described in this paper, for these the interested reader is referred to [17].

4.1 Network connectivity

Different parts of our model describe information concerning different aspects of the input data. The note-centered representation corresponds to rhythmic and harmonic structure of the score surrounding a note, while the velocity-history representation relates more closely to expressive phrases. This distinction continues through the layers of feature activations. To get an impression of how strongly the expressive variation in velocity data corresponds to these different aspects, we experimented with the different layers in isolation and together. We will refer to the network configurations by the layers that were used during training and prediction, i.e. $\mathbf{L_{1,2}}$ means both of the layers on top of the note-centered representation were used, and $\mathbf{L_3}$ was not. Another way to see this would be that $\mathbf{w_3}$ is constrained to be a matrix of only 0's.

	no vel. inf.			**with vel. inf.**	
	$\mathbf{L_1}$	$\mathbf{L_{1,2}}$	$\mathbf{L_2}$	$\mathbf{L_3}$	$\mathbf{L_{1,2,3}}$
M. $\rightarrow$ **M.**	.202	<u>.207</u>	.191	.315	.470
B. $\rightarrow$ **B.**	.366	.376	.357	.236	.532
B. $\rightarrow$ **M.**	.132	.126	.125	.286	.386
M. $\rightarrow$ **B.**	.291	.295	.283	.209	.457
All $\rightarrow$ **M.**	.198	.203	.186	.313	.466
All $\rightarrow$ **B.**	.341	.350	.329	.222	.503

Table 1: $\bar{R}^2$ scores obtained on the test data. $X \rightarrow Y$ indicates the model was trained on X and tested on Y, where **M.** is the Magaloff and **B.** the Batik dataset. Experiments with velocity information (vel. inf.) use the velocity-history representation as input. We use the underlined result for comparison with previous work ([7] and [8]).

4.2 Training datasets

Experimenting with different sets of training data is interesting for several reasons. One is that from a musicological perspective, the structure of music of different styles can be quite different. As an extreme example, a system trained on Jazz music would not be expected to reliably predict performances of piano music by Bach. Another reason is that we can use combinations of datasets to test the validity of our model: if a model trained on music from one set of recordings, still performs well on another set of recordings, this can give us some confidence that our model has learned something about music in a general sense, and not just about the particular dataset.

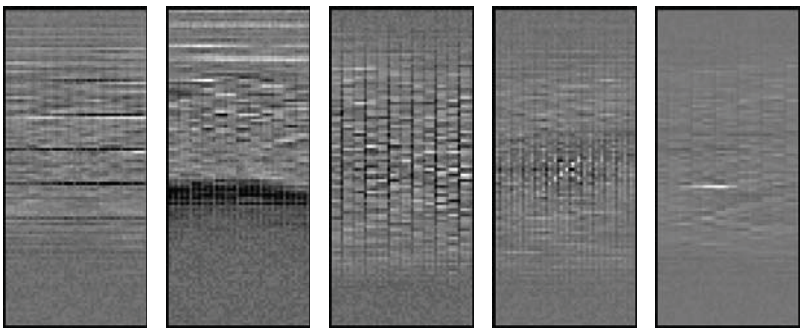
As mentioned before, we use two datasets: one describing performances of Chopin music and the other Mozart music. In all cases, during testing we kept the datasets separate. However, we varied the set of data used for training: we trained on the same dataset as used for testing, we trained on one dataset and tested on the other, and we tested a model trained on all data.

5. RESULTS

Table 1 lists the results obtained with our model. The model is more successful explaining the variance in the Batik (Mozart) data than in the Magaloff (Chopin) data – one possible explanation for this is that Chopin's music (from the Romantic period) has much more extreme variations in expression than Mozart's music (from the Classical period). It seems reasonable that a performance with more dynamic variation is harder to predict.

When comparing the different architectures, most information used by our model is encoded in $\mathbf{L_1}$ and $\mathbf{L_3}$. $\mathbf{L_2}$ has less predictive value than $\mathbf{L_1}$, and the score only improves by a little bit when these two layers are used together (suggesting there is a large amount of overlap in the information they encode). $\mathbf{L_3}$, which is based on the velocity-history representation (which was not used in [7]) clearly contains a lot of information.

Interestingly, $\mathbf{L_3}$ contains most relevant information for



(a) (b) (c) (d) (e)

Figure 3: Some hand-selected features from $\mathbf{L_1}$ that are representative for the types of patterns learned from the note-centered representation (see Figure 2b). Dark values correspond to negative weights, light values to positive weights.

the Magaloff data, and $\mathbf{L_1}$ for the Batik data. This could be due to the difference between music from the Romantic period and that from the Classical period: $\mathbf{L_1}$ contains more information about harmony, whereas $\mathbf{L_3}$ contains more information about the expressive 'flow' of the piece.

Training on a single dataset has a positive effect on the prediction scores. This is likely due to the fact that the datasets are of a different nature in terms of musical style, and if we would want to predict performance parameters for a Mozart piece, training on Chopin music will not provide our model with the relevant 'know-how'. This is also illustrated by the cross-training experiments, where we trained on one dataset and tested on the other: a drop in performance of around 0.08 in all cases is observed. Still, also a relatively large amount of the predictive capability remains, providing some confidence that our model generalizes over different datasets to some extent.

Because the velocity-history representation requires detailed performance data for predictions, we use the results from our $\mathbf{L_{1,2}}$ experiments when comparing our results to earlier work which does not use performance data. In [7] the best obtained $\bar{R}^2$ score on the Magaloff data is .139, using a single dense RBM layer with 1000 hidden units (similar to our $\mathbf{L_1}$ model). Our $\mathbf{L_{1,2}}$ model achieved an $\bar{R}^2$ of .207 on the same dataset. To keep statistical testing simple, we tested the statistical significance of the difference in *unweighted* average R^2 of our model and the model in [7] using a Wilcoxon signed rank test. We chose the Wilcoxon test because the underlying distribution of the R^2 data is unknown. We found that the unweighted average R^2 of .199 of our $\mathbf{L_{1,2}}$ model is significantly different from the unweighted average R^2 of .121 of the model in [7] ($W = 11111, p < 2.2 \cdot 10^{-16}$). In [8], the maximal obtained prediction accuracy on the Magaloff dataset is an $\bar{R}^2$ of .188. This model uses information our models have no access to, most importantly dynamic score annotations. Nevertheless, with an $\bar{R}^2$ of .207 our $\mathbf{L_1 L_2}$ model again seems more successful even though it does not take such annotations into account. [7] When we do use performance data, the difference becomes more pronounced: our $\mathbf{L_{1,2,3}}$ model obtains an $\bar{R}^2$ of .470 on the Magaloff data.

Something interesting to mention here is that in [17] we also experimented with limiting training data to a particular genre (i.e. training only on Nocturnes). These experiments suggested that the velocity-history representation encodes some genre-specific information, however due to space constraints we do not cover these results further here.

6. DISCUSSION

We discuss two properties of our model: the features that were learned from the musical data, and the performance achieved during prediction. Figure 3 illustrates a number of hand-selected features that have been learned from the note-centered representation, which were chosen to give an impression of the variety of learned features. Compared to the features learned by [7], there is a larger variety of features, where features represent sharper patterns.

6.1 Learned features

Figure 3 illustrates some of the learned features. The displayed features were selected so as to give the reader an impression of the diversity of the learned features . From a musicological perspective, it is interesting to see that there seem to be some remarkable patterns relating the features to music theory. The features learned from the velocity-history representation are harder to interpret musicologically, these are not further discussed in this paper.

Figure 3a shows clear horizontal banding, where interestingly the bands are exactly 12 rows apart – this corresponds to octaves. The feature in some locations displays a strong contrast between pitches one semi-tone apart, which is related to dissonance.

A common pattern is illustrated in Figure 3b, with a dark (inhibitive) band above or below a lighter region. This type of feature is also described by Grachten & Krebs [7], who argue this can be regarded as an accompaniment versus melody detector: the illustrated feature is strongly inhibited by notes in a sample that are below the central note, meaning that the feature activates more readily for bass notes. The opposite type of feature, with inhibitive regions above and excitatory regions below the central note (not

[7] To perform the statistical test, detailed results from [7] were kindly provided by the authors. For the work in [8] these results were unfortunately unavailable, meaning we could not perform the same statistical analysis with this result.

shown here), is active with a high probability for melody notes, where surrounding notes have lower pitch.

Another common pattern is the vertical banding illustrated by Figure 3c. There is some variation in the offset of the vertical bands from the edges (their phase) and how close they are together (their period). These features can convey information on the pace in the current part of the piece (predominantly short or long notes) and the temporal position of the note with respect to the beat.

A few features also display diagonal banding as illustrated by Figure 3d, although these are relatively rare. Still, we hypothesize that with these our model can deduce whether the central note is in an ascending or descending sequence.

A final common pattern is that in Figure 3e, with a sharp white band corresponding to a note at a certain relative pitch and time from the central note. It seems reasonable to suggest that these can be related to particular melodic steps – changes from one note to another with a particular relative pitch and timing.

6.2 Model performance

The performance of our model is an improvement compared to earlier work, particularly when the goal is to *explain* the structure of an existing performance rather than predict a performance for a new piece of score – in the former situation the velocity-history representation can be used to good effect. Still, when considering a purely predictive context (using no velocity information), an R^2 of around 0.2 leaves room for improvement. There is of course a practical limit in terms of what score can be obtained: even the same pianist might not play a piece in exactly the same way on different occasions, meaning that an R^2 close to 1.0 cannot be expected. A factor that limits our model is that it considers score structure at a local level only – structure at larger timescales is not considered, nor are loudness annotations, which of course also convey a lot of information about how loudly a particular piece of score is to be played. These omissions are opportunities for further work: including these components could improve performance further, for example loudness annotations could be included similarly to what was done in [8].

7. CONCLUSIONS

We showed that neural networks trained on relatively raw representations of musical score and musical performances can be used to predict expressive dynamics in piano performances. This was done before in [7], but we changed the learning architecture (using sparse RBMs and skip-connections), and developed a new input representation, resulting in better predictions and clearer features. We also showed that our model generalizes well to datasets on which it was not trained.

8. ACKNOWLEDGEMENTS

The research described in this article was sponsored by the Austrian Science Fund (FWF) under project Z159 (Wittgenstein Award), and by the European Commission under the projects Lrn2Cre8 (grant agreement no. 610859), and PHENICX (grant agreement no. 601166). The research was part of an MSc project resulting in a thesis [17], the contents of which overlap to some extent with those in this paper. W. Bas de Haas is supported by the Netherlands Organization for Scientific Research, through the NWO-VIDI-grant 276-35-001.

9. REFERENCES

[1] E. Bisesi and R. Parncutt. An accent-based approach to automatic rendering of piano performance: Preliminary auditory evaluation. *Archives of Acoustics*, 36(2):283–296, 2010.

[2] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *Proceedings of the Twenty-nine International Conference on Machine Learning*. ACM, 2012.

[3] S. Flossmann, W. Goebl, M. Grachten, B. Niedermayer, and G. Widmer. The magaloff project: An interim report. *Journal of New Music Research*, 39(4):363–377, 2010.

[4] A. Friberg, L. Fryden, L. Bodin, and J. Sundberg. Performance rules for computer-controlled contemporary keyboard music. *Computer Music Journal*, 15(2):49–55, 1991.

[5] A. Gabrielsson and P.N. Juslin. Emotional expression in music performance: Between the performer's intention and the listener's experience. *Psychology of music*, 24(1):68–91, 1996.

[6] H. Goh, N. Thome, and M. Cord. Biasing restricted boltzmann machines to manipulate latent selectivity and sparsity. In *NIPS workshop on deep learning and unsupervised feature learning*, 2010.

[7] M. Grachten and F. Krebs. An assessment of learned score features for modeling expressive dynamics in music. *Transactions on multimedia: Special issue on music data mining*, 16(5):1–8, 2014.

[8] M. Grachten and G. Widmer. Explaining musical expression as a mixture of basis functions. In *Proceedings of the 8th Sound and Music Computing Conference (SMC 2011)*, 2011.

[9] G.E. Hinton. A practical guide to training restricted boltzmann machines. *Momentum*, 9(1):926, 2010.

[10] G.E. Hinton and T.J. Sejnowski. Learning and relearning in boltzmann machines. *MIT Press, Cambridge, Mass*, 1:282–317, 1986.

[11] H. Katayose and S. Inokuchi. Learning performance rules in a music interpretation system. *Computers and the Humanities*, 27(1):31–40, 1993.

[12] F. Lerdahl and R.S. Jackendoff. *A generative theory of tonal music*. The MIT Press, 1983.

[13] C. Palmer. Music performance. *Annual review of psychology*, 48(1):115–138, 1997.

[14] R. Parncutt. Accents and expression in piano performance. *Perspektiven und Methoden einer Systemischen Musikwissenschaft*, pages 163–185, 2003.

[15] H. Schenker. Five graphic music analyses, 1969.

[16] A. Spiliopoulou and A. Storkey. Comparing probabilistic models for melodic sequences. In *Proceedings of the 2011 European conference on Machine learning and knowledge discovery in databases - Volume Part III*, ECML PKDD'11, pages 289–304, Berlin, Heidelberg, 2011. Springer-Verlag.

[17] S. van Herwaarden. Teaching neural networks to play the piano. Master's thesis, Utrecht University, 2014.

[18] G. Widmer. Large-scale induction of expressive performance rules: First quantitative results. In *Proceedings of the International Computer Music Conference (ICMC'2000)*, pages 344–347, 2000.

AN RNN-BASED MUSIC LANGUAGE MODEL FOR IMPROVING AUTOMATIC MUSIC TRANSCRIPTION

Siddharth Sigtia†, Emmanouil Benetos‡, Srikanth Cherla‡, Tillman Weyde‡, Artur S. d'Avila Garcez‡, and Simon Dixon†
† Centre for Digital Music, Queen Mary University of London
‡ Department of Computer Science, City University London
† { s.s.sigtia, s.e.dixon}@qmul.ac.uk
‡ { emmanouil.benetos.1,srikanth.cherla.1,t.e.weyde,a.garcez}@city.ac.uk

ABSTRACT

In this paper, we investigate the use of Music Language Models (MLMs) for improving Automatic Music Transcription performance. The MLMs are trained on sequences of symbolic polyphonic music from the Nottingham dataset. We train Recurrent Neural Network (RNN)-based models, as they are capable of capturing complex temporal structure present in symbolic music data. Similar to the function of language models in automatic speech recognition, we use the MLMs to generate a prior probability for the occurrence of a sequence. The acoustic AMT model is based on probabilistic latent component analysis, and prior information from the MLM is incorporated into the transcription framework using Dirichlet priors. We test our hybrid models on a dataset of multiple-instrument polyphonic music and report a significant 3% improvement in terms of F-measure, when compared to using an acoustic-only model.

1. INTRODUCTION

Automatic Music Transcription (AMT) involves automatically generating a symbolic representation of an acoustic musical signal [4]. AMT is considered to be a fundamental topic in the field of music information retrieval (MIR) and has numerous applications in related fields in music technology, such as interactive music applications and computational musicology. The majority of recent transcription papers utilise and expand *spectrogram factorisation* techniques, such as non-negative matrix factorisation (NMF) [18] and its probabilistic counterpart, probabilistic latent component analysis (PLCA) [25]. Spectrogram factorisation techniques decompose an input spectrogram of the audio signal into a product of spectral templates (that typically correspond to musical notes) and component activations (that indicate whether each note is active at a given time frame). Spectrogram factorisation-based AMT systems include the work by Bertin et al. [7], who proposed a Bayesian framework for NMF, which considers each pitch as a model of Gaussian components in harmonic positions. Benetos and Dixon [3] proposed a convolutive model based on PLCA, which supports the transcription of multiple-instrument music and supports tuning changes and frequency modulations (modelled as shifts across log-frequency).

In terms of connectionist approaches for AMT, Nam et al. [20] proposed a method where features suitable for transcribing music are learned using a deep belief network consisting of stacked restricted Boltzmann machines (RBMs). The model performed classification using support vector machines and was applied to piano music. Böck and Schedl used recurrent neural networks (RNNs) with Long Short-Term Memory units for performing polyphonic piano transcription [8], with the system being particularly good at recognising note onsets.

There is no doubt that a reliable acoustic model is important for generating accurate symbolic transcriptions of a given music signal. However, since music exhibits a fair amount of structural regularity much like language, it is natural for one to think of the possibility of improving transcription accuracy using a *music language model* (MLM) in a manner akin to the use of a language model to improve the performance of a speech recognizer [21]. In [9], the predictions of a polyphonic MLM were used to this end, which was further developed in [10], where an input/output extension of the RNN-RBM was proposed that learned to map input sequences to output sequences in the context of AMT. Both in [9] and [10], evaluations were performed using synthesized MIDI data. In [22], Raczyński et al. utilise chord and key information for improving an NMF-based AMT system in a post-processing step. A major advantage of using a hybrid acoustic + language model system is that the two models can be trained independently using data from different sources. This is particularly useful since annotated audio data is scarce while it is relatively easy to find MIDI data for training robust language models.

In the present work, we integrate a MLM with an AMT system, in order to improve transcription performance. Specifically, we make use of the predictions made by a Recurrent Neural Network (RNN) and a RNN-Neural Autore-

gressive Distribution Estimator (RNN-NADE) based polyphonic MLM proposed in [9] to refine the transcriptions of a PLCA-based AMT system [2, 3]. Information from the MLM is incorporated into the PLCA-based acoustic model as a prior for pitch activations during parameter estimation. It is observed that combining the two models in this way boosts transcription accuracy by +3% on the Bach10 dataset of multiple-instrument polyphonic music [13], compared to using the acoustic AMT system only.

The outline of this paper is as follows. The PLCA-based transcription system is presented in Section 2. The RNN-based polyphonic music prediction system that is used as a music language model is described in Section 3. The combination of the two aforementioned systems is presented in Section 4. The employed dataset, evaluation metrics, and experimental results are shown in Section 5; finally, conclusions are drawn in Section 6.

2. AUTOMATIC MUSIC TRANSCRIPTION SYSTEM

For combining acoustic and music language information in an AMT context, we employ the model of [3], which supports the transcription of multiple-instrument polyphonic music and also supports pitch deviations and frequency modulations. The model of [3] is based on PLCA, which is a latent variable analysis method which has been used for decomposing spectrograms. For computational efficiency purposes, we employ the fast implementation from [2], which utilized pre-extracted note templates that are also pre-shifted across log-frequency, in order to account for frequency modulations or tuning changes. In addition, as was shown in [24], PLCA-based models can utilise priors for estimating unknown model parameters, which will be useful in this paper for informing the acoustic transcription system with symbolic information.

The transcription model takes as input a normalised log-frequency spectrogram $V_{\omega,t}$ (ω is the log-frequency index and t is the time index) and approximates it as a bivariate probability distribution $P(\omega, t)$. $P(\omega, t)$ is decomposed into a series of log-frequency spectral templates per pitch, instrument, and log-frequency shifting (which indicates deviation with respect to the ideal tuning), as well as probability distributions for pitch, instrument, and tuning.

The model is formulated as:

$$P(\omega, t) = P(t) \sum_{p,f,s} P(\omega|s, p, f) P_t(f|p) P_t(s|p) P_t(p) \tag{1}$$

where p denotes pitch, s denotes the musical instrument source, and f denotes log-frequency shifting. $P(t)$ is the energy of the log-spectrogram, which is a known quantity. $P(\omega|s, p, f)$ denotes pre-extracted log-spectral templates per pitch p and instrument s, which are also pre-shifted across log-frequency. The pre-shifting operation is made in order to account for pitch deviations, without needing to formulate a convolutive model across log-frequency. $P_t(f|p)$ is the time-varying log-frequency shifting distribution per pitch, $P_t(s|p)$ is the time-varying source contribution per pitch, and finally, $P_t(p)$ is the pitch activation, which essentially is the resulting music transcription. As a time-frequency representation in the log-frequency domain we use the constant-Q transform (CQT) with a log-spectral resolution of 60 bins/octave [23].

The unknown model parameters ($P_t(f|p)$, $P_t(s|p)$, and $P_t(p)$) can be iteratively estimated using the expectation-maximisation (EM) algorithm [12]. For the *Expectation* step, the following posterior is computed:

$$P_t(p, f, s|\omega) = \frac{P(\omega|s, p, f) P_t(f|p) P_t(s|p) P_t(p)}{\sum_{p,f,s} P(\omega|s, p, f) P_t(f|p) P_t(s|p) P_t(p)} \tag{2}$$

For the *Maximization* step (without using any priors) unknown model parameters are updated using the posterior computed from the Expectation step:

$$P_t(f|p) \propto \sum_{\omega,s} P_t(p, f, s|\omega) V_{\omega,t} \tag{3}$$

$$P_t(s|p) \propto \sum_{\omega,f} P_t(p, f, s|\omega) V_{\omega,t} \tag{4}$$

$$P_t(p) \propto \sum_{\omega,f,s} P_t(p, f, s|\omega) V_{\omega,t} \tag{5}$$

We consider the sound state templates to be fixed, so no update rule for $P(\omega|s, p, f)$ is applied. Using fixed templates, 20-30 iterations using the update rules presented in the present section are sufficient for convergence. The output of the system is a pitch activation which is scaled by the energy of the log-spectrogram:

$$P(p, t) = P(t) P_t(p) \tag{6}$$

After performing 5-sample median filtering for note smoothing, thresholding is performed on $P(p, t)$ followed by minimum note duration pruning set to 40ms in order to convert $P(p, t)$ into a binary piano-roll representation, which is the output of the transcription system, and is also used for evaluation purposes.

3. POLYPHONIC MUSIC PREDICTION SYSTEM

Taking inspiration from speech recognition, it has been shown that a good statistical model of symbolic music can help the transcription process [11]. However there are 2 main reasons for the use of MLMs in AMT not being more common.

1. Training models that capture the temporal structure and complexity of symbolic polyphonic music is not an easy task. In speech recognition, often simple language models like n-grams work extremely well. However, music has a more complex structure and simple statistical models like n-grams and HMMs fail to model these characteristics accurately even for music with simple structure [9].

2. There is no consensus on how to incorporate this prior information within the transcription system. However, recently there have been some successful attempts at using this prior information to improve the accuracy on AMT tasks [9, 10].

In this section we discuss the details of the music prediction system and the models used. In the next section we discuss how we incorporate the predictions from these models in a PLCA-based music transcription system.

3.1 Recurrent Neural Networks

A recurrent neural network (RNN) is a powerful model for time-series data which can account for long-term temporal dependencies, over multiple time-scales when trained effectively. Given a sequence of inputs $v_1, v_2, \ldots, v_T$ each in $\mathbb{R}^n$, the network computes a sequence of hidden states $\hat{h}_1, \hat{h}_2, \ldots, \hat{h}_T$ each in $\mathbb{R}^m$, and a sequence of predictions $\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_T$ each in $\mathbb{R}^k$ by iterating the equations

$$\hat{h}_t = e(W_{\hat{h}x} v_t + W_{\hat{h}\hat{h}} \hat{h}_{t-1} + b_{\hat{h}}) \quad (7)$$

$$\hat{y}_t = g(W_{y\hat{h}}) \quad (8)$$

where $W_{y\hat{h}}$, $W_{\hat{h}x}$, $W_{\hat{h}\hat{h}}$ are the weight matrices, $b_{\hat{h}}$ is the bias and e and g are activation functions which are typically non-linear and applied element-wise.

An RNN can be trained using the gradient-based Back-Propagation Through Time algorithm [27] using the exactly computable error gradients in the network. However, 1^{st} order gradient methods fail to correctly train RNNs for many real-world problems. This difficulty has been associated with what is known as the *vanishing/exploding gradients* phenomenon [6], where the errors exhibit exponential decay/growth as they are back-propagated through time. years [15, 16, 19].

However, recent work in the field of neural networks and deep learning has led to several improvements in gradient based optimization methods that make training of RNNs possible. Most notably, the Hessian Free (HF) optimization algorithm has been used to train RNNs successfully on several real world datasets, including symbolic polyphonic music data [19]. Apart from second order methods like HF, several modifications to first-order gradient based methods exist that currently form the state of the art in training RNNs [5].

3.2 Recurrent Neural Network-based models

One of the drawbacks of using RNNs to predict polyphonic symbolic music is that any output of the network, $\hat{y}_i$ at time step t, is conditionally independent of $\hat{y}_j, \forall j \neq i$ given the sequence of input vectors $v_1, v_2, \ldots, v_T$. This is a severe constraint when used for modelling polyphonic music, where notes often appear in very correlated patterns within a frame. In order to overcome this limitation, models derived from RNNs have been proposed which are better at modelling high-dimensional sequences [9, 26].

The first RNN-based model that tried to model high-dimensional sequences is the Recurrent Temporal Restricted Boltzmann Machine (RTRBM) [26]. This model was extended to the more general RNN-RBM model, where the hidden states for the RBM and RNN were not constrained to be the same. For our prediction system, we make use of a variant of the RNN-RBM, called the RNN-NADE. The only difference is that the conditional distributions at each step are modelled by a Neural Autoregressive Distribution Estimator (NADE) [17] as opposed to an RBM. As discussed in the next section, to combine the predictions with the transcription system, we need individual pitch activation probabilities at each time-step. Obtaining these probabilities from an RBM is intractable as it requires summing over all possible hidden states. However the NADE is a tractable distribution estimator and we can easily obtain these probabilities from the NADE. The NADE models the probability of occurrence of a vector p as:

$$P(p) = \prod_{i=1}^{D} P(p_i | \mathbf{p}_{<\mathbf{i}}) \quad (9)$$

where $p \in \mathbb{R}^D$,p_i is the pitch activation and $\mathbf{p}_{<\mathbf{i}}$ is the vector containing all the pitch activations p_j such that $j < i$.

In our system we utilise each of the conditional probabilities $p(p_i|\mathbf{p}_{<\mathbf{i}})$ as probabilities of the pitch activations. Although the pitch activation probabilities are only conditioned on $\mathbf{p}_{<\mathbf{i}}$, we hypothesize that this will be a better model than the RNN, where the pitch activation probabilities are completely independent. Another motivation for using the NADE is that the gradients can be computed exactly, and therefore we can employ HF optimization for training the RNN-NADE.

4. COMBINING TRANSCRIPTION AND PREDICTION

In this section, we describe the process for combining the acoustic model with the music language model for deriving an improved transcription. Firstly, the input music signal is transcribed using the process described in Section 2. The resulting piano-roll representation of the transcription system is considered to be a sequence $p_1, p_2, \ldots, p_T$ that is placed as input to the MLM presented in Section 3. For the RNN-NADE, we compute the probability $P(p_i|\mathbf{p}_{<\mathbf{i}})$ for all time frames, and use that as prior information for the combined model, with the prior information denoted as $P_{MLM}(p,t)$, where $P_{MLM}(p = i, t) = P(p_i|\mathbf{p}_{<\mathbf{i}})$. For the RNN, the prediction output is directly denoted as $P_{MLM}(p,t)$, since pitch probabilities are independent.

As shown in [24], PLCA-based models use multinomial distributions; since the Dirichlet distribution is conjugate to the multinomial, a Dirichlet prior can be used to enforce structure on the pitch activation distribution $P_t(p)$. Following the procedure of [24], we define the Dirichlet hyperparameter for the pitch activation as:

$$\alpha_t(p) \propto P_t(p) P_{MLM}(p,t) \quad (10)$$

where $\alpha_t(p)$ essentially is a pitch activation probability which is filtered through a pitch indicator function computed from the symbolic prediction step (the denominator is simply for normalisation purposes).

The recording is then re-transcribed, using as additional information the prior computed from the transcription step. The modified update for the pitch activation which replaces

(5) is given by:

$$P_t(p) \propto \sum_{\omega,f,s} P_t(p,f,s|\omega)V_{\omega,t} + \kappa\alpha_t(p) \qquad (11)$$

where κ is a weight parameter expressing how much the prior should be imposed; as in [24], the weight decreases from 1 to 0 throughout the iterations. To summarize, the transcription creates a symbolic prediction, which in turn improves the subsequent re-transcription of the music signal. An overview of the complete transcription-prediction system architecture can be seen in Fig. 1.

5. EVALUATION

5.1 Dataset

For testing the transcription system, we employ the Bach10 dataset [13], which is a freely available multi-track collection of multiple-instrument polyphonic music. It consists of ten recordings of J.S. Bach chorales, performed by violin, clarinet, saxophone, and bassoon. Pitch ground truth for each instrument is also provided. Due to the tonal and homogeneous content of the dataset (single composer, single music language), it is suitable for testing the incorporation of music language models in a multiple-instrument transcription system. For training the transcription system, pre-extracted and pre-shifted spectral templates are extracted for the instruments present in the dataset, using isolated note samples from the RWC database [14].

For training the MLMs we use the Nottingham dataset [1] , a collection of 1200 music pieces in symbolic ABC format, which contain simple chord combinations and tunes. We trained the RNN and the RNN-NADE models using both Stochastic Gradient Descent (SGD) and HF to compare performance. The inputs to both the models are sequences of length 200 where each frame of the sequence is a binary vector of length 88 which covers the full piano note range. We train both the RNN and the RNN-NADE to predict the next vector given a sequence of input vectors. We train the models by minimizing the negative log-likelihood of the sequences using the cross-entropy $\sum_i t_i \log p_i + (1 - t_i)\log(1 - p_i)$ where i sums over all the dimensions of the binary vector and t_i is the pitch target.

5.2 Metrics

For evaluating the performance of the proposed system for multi-pitch detection, we employ the precision (Pre), recall (Rec), and F-measure (F) metrics, which are commonly used in transcription evaluations [1]. As in the public evaluations on multi-pitch detection carried out through the MIREX framework [1], a detected note is considered correct if its pitch is the same as the ground truth pitch and its onset is within a 50ms tolerance interval of the ground-truth onset.

[1] ifdo.ca/∼seymour/nottingham/nottingham.html

Model	Pre
RNN (SGD)	67.89%
RNN (HF)	69.61%
RNN-NADE (SGD)	68.89%
RNN-NADE (HF)	**70.61**%

Table 1. Validation results for MLMs

5.3 Results

To validate the performance of the MLMs, we calculate the prediction precision on unseen sequences of music from the Nottingham dataset of folk melodies. We utilise 694 tracks for training, 173 tracks for validation and 170 for testing [2] . For both the RNN and RNN-NADE models we sample 10 vectors from the conditional distribution at each time-step and calculate the expected precision against the ground truth. The reported precision is found by finding the mean over the predictions of every frame. Table 1 shows the results of the validation experiments. These results are of the same order as the prediction accuracies reported in [9]. We found that for both the models, HF optimization gave better precision than SGD. Training with HF was also easier as there were less hyper parameters to be tuned when compared to SGD, where learning rate needs to be updated to make sure training is effective. The RNN models had a hidden layer of size 150, while the RNN-NADE models had a hidden layer of size 100 and the NADE consisted of a hidden layer of size 150.

Multi-pitch detection experiments are performed using the proposed system, with various configurations. A first configuration only considers the transcription system from Section 2. A second configuration takes the output of the transcription system and gives it as input to the prediction system of Section 3, where the final piano-roll is the output of the prediction step. A third configuration (presented in Section 4), re-transcribes the recording, having the prediction as a prior information for estimating the pitch activations. For the prediction system, experiments were performed using both the RNN-NADE and the RNN.

Results using the various system configurations are displayed in Table 2. It can be seen that the best performance is achieved by the 3rd configuration when using the NADE-HF model for prediction, which surpasses the acoustic-only transcription system by more than 3%. In general, it can be seen that using the prediction system as a post-processing step (2nd configuration) always leads to an improvement over the acoustic-only model (1st configuration). A similar trend can be observed when integrating the prediction information as a prior in the transcription system (configuration 3) compared to just using the prediction system as post-processing (configuration 2); an improvement is always reported. Another observation can be made when comparing the RNN-NADE with the RNN, with the former providing a clear improvement. For comparative purposes, we also trained MLMs using 500 MIDI files of J.S. Bach chorales [3] and tested the models on the

[2] http://www-etud.iro.umontreal.ca/∼boulanni/icml2012

[3] http://www.jsbchorales.net/sets.shtml

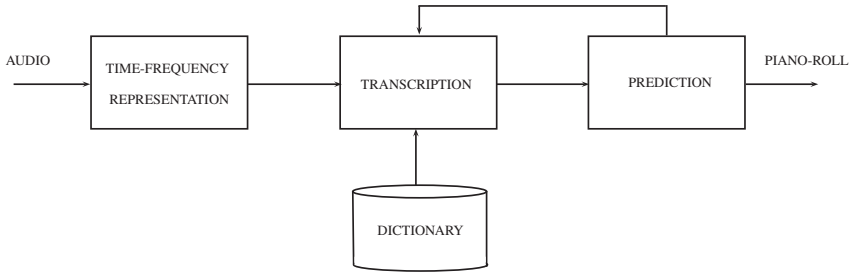


Figure 1. Proposed system diagram.

Configuration	F	Pre	Rec
Configuration 1	62.02%	58.51%	66.12%
Configuration 2 - NADE	62.62%	59.70%	65.92%
Configuration 3 - NADE	64.08%	61.96%	66.44%
Configuration 2 - RNN	62.29%	59.08%	65.98%
Configuration 3 - RNN	63.85%	61.14%	66.90%
Configuration 2 - NADE-HF	62.20%	59.14%	65.68%
Configuration 3 - NADE-HF	**65.16%**	**62.80%**	**67.78%**
Configuration 2 - RNN-HF	62.44%	59.28%	66.07%
Configuration 3 - RNN-HF	62.87%	60.03%	66.11%

Table 2. Transcription results using various system configurations.

Bach10 recordings. Using the Bach MLMs, the system reached $F = 63.58\%$, which is an improvement over the acoustic-only system, but is outperformed by the Nottingham language model.

Qualitatively, the MLMs are able to improve transcription performance by providing a rough estimate of which pitches are expected to appear in the recording (and which pitches are not expected to appear). The language models were trained using simple chord sequences (from the Nottingham dataset) that are representative of simple tonal music and are applicable as language models to the more complex Bach chorales. We believe that the reason for the J.S. Bach MLMs not performing as well as the Nottingham MLMs is due to the fact that predicting Bach's music is a complex task (many exceptions, key changes, modulations), whereas a simple tonal model like the Nottingham dataset can work as a general-purpose language model in many types of music (this is also verified in [9]).

By comparing with the method of [13] (where the Bach10 dataset was first introduced), the proposed method using the frame-based accuracy metric reaches 74.3% for the NADE-HF using the 3rd configuration, whereas the method of [13] reaches 69.7% (with unknown polyphony). As an example of the proposed system's performance, the spectrogram and raw output of the system using the 3rd configuration is displayed for a Bach10 recording Fig. 2, whereas the post-processed transcription output along with the ground truth for the same recording is shown in Fig. 3.

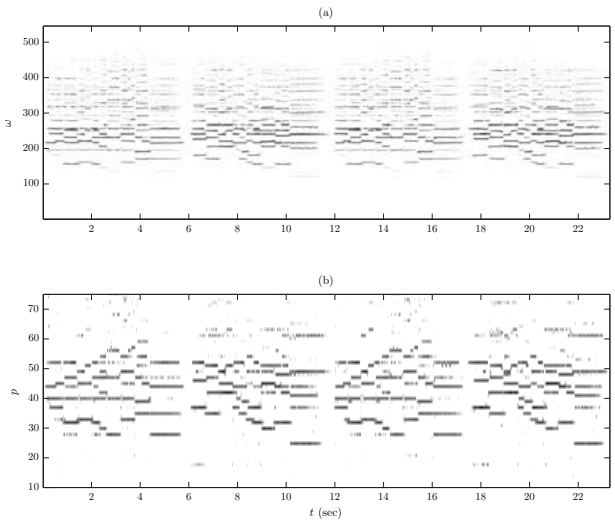


Figure 2. (a) The spectrogram $V_{\omega,t}$ for recording "Ach Lieben Christen" from the Bach10 dataset. (b) The pitch activation $P(p,t)$ using the transcription-prediction system using the 3rd configuration, with the NADE-HF.

6. CONCLUSIONS

In this paper, we proposed a system for automatic music transcription which incorporated prior information from a polyphonic music prediction model based on recurrent neural networks. The acoustic transcription model was based on probabilistic latent component analysis, and information from the prediction system was incorporated using Dirichlet priors. Experimental results using the multiple-instrument Bach10 dataset showed that there is a clear and significant improvement (3% in terms of F-measure) by combining a music language model with an acoustic model for improving the performance of the latter. These results also demonstrate that the MLM can be trained on symbolic music data from a different source as the acoustic data, thus eliminating the need to acquire collections of symbolic and corresponding acoustic data (which are scarce).

In the current system, the language models are trained on only one dataset. In the future, we would like to evaluate the proposed system using language models trained from different sources to see if this helps the MLMs generalize better. We will also investigate different system configurations, to test whether iterating the transcription and prediction steps leads to improved performance. We will also investigate the effect of using different RNN architectures like Long Short Term Memory (LSTM) and bi-directional RNNs and LSTMs. Finally, we would like to extend the current models for high-dimensional sequences to better fit the requirements for music language modelling.

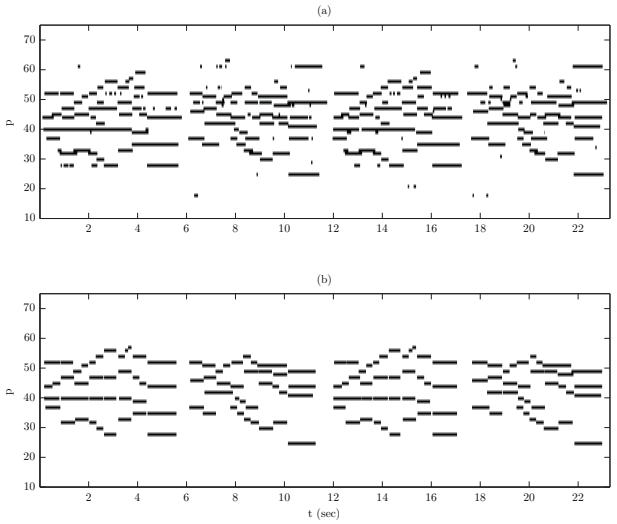


Figure 3. Transcription example for recording "Ach Lieben Christen" from the Bach10 dataset. (a) The post-processed output of the transcription-predicton system using the 3rd configuration, with the NADE-HF. (b) The pitch ground truth of the recording.

7. ACKNOWLEDGEMENT

SS is supported by a City University London Pump-Priming Grant and the Queen Mary University of London Postgraduate Research Fund. EB is supported by a City University London Research Fellowship. SC is supported by a City University London Research Studentship.

8. REFERENCES

[1] Music Information Retrieval Evaluation eXchange (MIREX). http://music-ir.org/mirexwiki/.

[2] E. Benetos, S. Cherla, and T. Weyde. An effcient shiftinvariant model for polyphonic music transcription. In *6th International Workshop on Machine Learning and Music*, 2013.

[3] E. Benetos and S. Dixon. A shift-invariant latent variable model for automatic music transcription. *Computer Music Journal*, 36(4):81–94, 2012.

[4] E. Benetos, S. Dixon, D. Giannoulis, H. Kirchhoff, and A. Klapuri. Automatic music transcription: challenges and future directions. *Journal of Intelligent Information Systems*, 41(3):407–434, December 2013.

[5] Y. Bengio, N. Boulanger-Lewandowski, and R. Pascanu. Advances in optimizing recurrent networks. In *ICASSP*, pages 8624–8628, May 2013.

[6] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Networks*, 5(2):157–166, 1994.

[7] N. Bertin, R. Badeau, and E. Vincent. Enforcing harmonicity and smoothness in Bayesian non-negative matrix factorization applied to polyphonic music transcription. *IEEE Trans. Audio, Speech, and Language Processing*, 18(3):538–549, March 2010.

[8] S. Böck and M. Schedl. Polyphonic piano note transcription with recurrent neural networks. In *ICASSP*, pages 121–124, March 2012.

[9] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *29th Int. Conf. Machine Learning*, 2012.

[10] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent. High-dimensional sequence transduction. In *ICASSP*, pages 3178–3182, May 2013.

[11] A. T. Cemgil. *Bayesian Music Transcription*. PhD thesis, Radboud University of Nijmegen, 2004.

[12] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.

[13] Z. Duan, B. Pardo, and C. Zhang. Multiple fundamental frequency estimation by modeling spectral peaks and non-peak regions. *IEEE Trans. Audio, Speech, and Language Processing*, 18(8):2121–2133, November 2010.

[14] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka. RWC music database: music genre database and musical instrument sound database. In *ISMIR*, Baltimore, USA, October 2003.

[15] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[16] H. Jaeger. Adaptive nonlinear system identification with echo state networks. In *Advances in neural information processing systems*, pages 593–600, 2002.

[17] H. Larochelle and I. Murray. The neural autoregressive distribution estimator. *Journal of Machine Learning Research*, 15:29–37, 2011.

[18] D. D. Li and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, October 1999.

[19] J. Martens and I. Sutskever. Learning recurrent neural networks with Hessian-free optimization. In *28th Int. Conf. Machine Learning*, pages 1033–1040, 2011.

[20] J. Nam, J. Ngiam, H. Lee, and M. Slaney. A classification-based polyphonic piano transcription approach using learned feature representations. In *ISMIR*, pages 175–180, October 2011.

[21] L. Rabiner and B.-H. Juang. Fundamentals of speech recognition. 1993.

[22] S.A. Raczynski, E. Vincent, and S. Sagayama. Dynamic Bayesian networks for symbolic polyphonic pitch modeling. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(9):1830–1840, 2013.

[23] C. Schörkhuber and A. Klapuri. Constant-Q transform toolbox for music processing. In *7th Sound and Music Computing Conf.*, Barcelona, Spain, July 2010.

[24] P. Smaragdis and G. Mysore. Separation by "humming": user-guided sound extraction from monophonic mixtures. In *IEEE WASPAA*, pages 69–72, October 2009.

[25] P. Smaragdis, B. Raj, and M. Shashanka. A probabilistic latent variable model for acoustic modeling. In *Neural Information Processing Systems Workshop*, Whistler, Canada, December 2006.

[26] I. Sutskever, G. E. Hinton, and G. W. Taylor. The recurrent temporal restricted Boltzmann machine. In *Advances in Neural Information Processing Systems*, pages 1601–1608, 2008.

[27] P. J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

TOWARDS MODELING TEXTURE IN SYMBOLIC DATA

Mathieu Giraud LIFL, CNRS Univ. Lille 1, Lille 3
Florence Levé MIS, UPJV, Amiens LIFL, Univ. Lille 1
Florent Mercier Univ. Lille 1
Marc Rigaudière Univ. Lorraine
Donatien Thorez Univ. Lille 1
{mathieu, florence, florent, marc, donatien}@algomus.fr

ABSTRACT

Studying *texture* is a part of many musicological analyses. The change of texture plays an important role in the cognition of musical structures. Texture is a feature commonly used to analyze musical audio data, but it is rarely taken into account in symbolic studies. We propose to formalize the texture in classical Western instrumental music as melody and accompaniment layers, and provide an algorithm able to detect homorhythmic layers in polyphonic data where voices are not separated. We present an evaluation of these methods for parallel motions against a ground truth analysis of ten instrumental pieces, including the first movements of the six quatuors op. 33 by Haydn.

1. INTRODUCTION

1.1 Musical Texture

According to Grove Music Online, texture refers to *the sound aspects of a musical structure*. One usually differentiates *homophonic* textures (rhythmically similar parts) and *polyphonic* textures (different layers, for example melody with accompaniment or countrapuntal parts). Some more precise categorizations have been proposed, for example by Rowell [17, p. 158 – 161] who proposes eight "textural values": orientation (vertical / horizontal), tangle (interweaving of melodies), figuration (organization of music in patterns), focus vs. interplay, economy vs. saturation, thin vs. dense, smooth vs. rough, and simple vs. complex. What is often interesting for the musical discourse is the *change of texture*: J. Dunsby, recalling the natural tendency to consider a great number of categories, asserts that *"one has nothing much to say at all about texture as such, since all depends on what is being compared with what"* [5].

Orchestral texture. The term *texture* is used to describe orchestration, that is the way musical material is layed out on different instruments or sections, taking into account registers and timbres. In his 1955 *Orchestration* book, W. Piston presents seven types of texture: orchestral unison, melody and accompaniment, secondary melody, part writing, contrapuntal texture, chords, and complex textures [15].

In 1960, Q. R. Nordgren [13] asks: *"Is it possible to measure texture?"*. He proposes to quantify the horizontal and vertical relationships of sounds making up the texture beyond the usual homophonic/polyphonic or light/heavy categories. He considers eight features, giving them numerical values: the number of instruments, their range, their register and their spacing, the proportion and register of gap, and doubling concentrations with their register. He then analyzes eight symphonies by Beethoven, Mendelssohn, Schumann and Brahms with these criteria, finding characteristic differences between those composers.

Non-orchestral texture. However, the term texture also relates to music produced by a smaller group of instruments, even of same timbre (such as a string quartet), or to music produced by a unique polyphonic instrument such as the piano or the guitar. As an extreme point of view, one can consider texture on a monophonic instrument: a simple monophonic sequence of notes can sound as a melody, but also can figure accompaniment patterns such as arpeggiated chords or Alberti bass.

Texture in musical analysis. Studying texture is a part of any analysis, even if texture often does not make sense on its own. As stated by J. Levy, *"although it cannot exist independently, texture can make the functional and sign relationships created by the other variables more evident and fully effective"* [10]. Texture plays a significant role in the cognition of musical structures. J. Dunsby attributes two main roles to texture: the illusion it creates and the expectation it arouses from the listeners towards familiar textures [5]. J. Levy shows with many examples how texture can be a sign in Classic and Early Romantic music, describing the role of accompaniment patterns, solos and unison to raise the attention of the listener before important structural changes [10].

1.2 Texture and Music Information Retrieval

Texture was often not as deeply analyzed and formalized as other parameters (especially melody or harmony). In the field of Music Information Retrieval (MIR), the notion of texture is often used in audio analysis, reduced to timbral description. Any method dealing with audio signals is somewhat dealing with timbre and texture [3, 9]. Based on audio texture, there were for example studies on segmentation. More generally, the term "sound texture" can be used to describe or synthesize non-instrumental audio signals, such as ambient sounds [18, 19].

Among the studies analyzing scores represented by symbolic data, few of them take texture into account. In 1989, D. Huron [7] explains that the three common meanings about the texture term are the volume, the diversity of elements used and the "surface" description, the first two being more easily formalizable. Using a two-dimensional space based on onset synchronization and similar pitch motion, he was able to capture four broad categories of textures: monophony, homophony, polyphony and heterophony. He found also that different musical genres occupy a different region of the defined space.

Some of the features of the jSymbolic library, used for classification of MIDI files, concern musical texture [11, 12]. *"[They] relate specifically to the number of independent voices in a piece and how these voices relate to one another."* [11, p. 209]. The features are computed on MIDI files where voices are separated, and include statistical features on choral or orchestral music organization: maximum, average and variability of the number of notes, variability between features of individual voices (number of notes, duration, dynamics, melodic leaps, range), features of the loudest voice, highest and lowest line, simultaneity, voice overlap, parallel motion and pitch separation between voices.

More recently, Tenkanen and Gualda [20] detect articulative boundaries in a musical piece using six features including pitch-class sets and onset density ratios. D. Rafailidis and his colleagues segment the score in several textural streams, based on pitch and time proximity rules [2,16].

1.3 Contents

As we saw above, there are not many studies on modeling or automatic analysis of texture. Even if describing musical texture could be done on a local level of a score, it requires some high-level musical understanding. We thus think that it is a natural challenge, both for music modeling and for MIR studies.

In this paper, we propose some steps towards the modeling and the computational analysis of texture in Western classical instrumental music. We choose here not to take into account orchestration parameters, but to focus on textural features given by local note configurations, taking into account the way these may be split into several layers. For the same reason, we do not look at harmony or at motives, phrases, or pattern large-scale repetition.

The following section presents a formal modeling of the texture and a ground truth analysis of first movements of ten string quartets. Then we propose an algorithm discovering texture elements in polyphonic scores where voices are not separated, and finally we present an evaluation of this algorithm and a discussion on the results.

2. FORMALIZATION OF TEXTURE

2.1 Modeling Texture as Layers

We choose to model the texture, by grouping notes into sets of *"layers"*, also called *"streams"*, sounding as a whole grouped by perceptual characteristics. Auditory stream segregation was introduced by Bregman, who studied many parameters influencing this segregation [1]. Focusing on the information contained on a symbolic score, notes can be grouped in such layers using perceptual rules [4, 16]. The number of layers is not directly the number of actual (monophonic) voices played by the instruments. For instance, in a string quartet where all instruments are playing, there can be as few as only one perceived layer, several voices blending in homorhythmy. On the contrary, some figured patterns in a unique voice can be perceived as several layers, as in a Alberti bass.

More precisely, *we model the texture in layers according to two complementary views*. First, we consider two main roles for the layers, that is how they are perceived by the listeners: *melodic* (mel) layers (dominated by contiguous pitch motion), and *accompaniment* (acc) layers (dominated by harmony and/or rhythm). Second, we describe how each layer is composed.

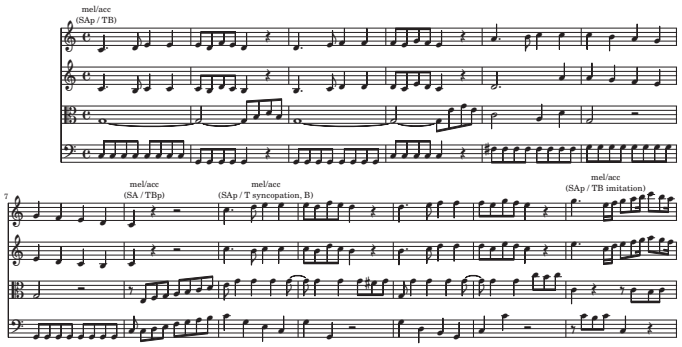
- A melodic layer can be either a monophonic voice (solo), or two or more monophonic voices in homorhythmy (h), or within a tighter relation, such as (from most generic to most similar) parallel motion (p), octave (o) or unison (u) doubling. The h/p/o/u relations do not need to be exact: for example, a parallel motion can be partly in thirds, partly in sixths, and include some foreign notes (see Figure 1).
- An accompaniment layer can also be described by h/p/o/u relations, but it is often worth focusing on its rhythmic component: for example, such a layer can contain sustained, sparse or repeated chords, Alberti bass, pedal notes, or syncopation.

The usual texture categories can then be described as:

- **mel/acc** – the usual accompanied melody;
- **mel/mel** – two independent melodies (counterpoint, imitation...);
- **mel** – one melody (either solo, or several voices in h/p/o/u relation), no accompaniment;
- **acc** – only accompaniment, when there is no noticeable melody that can be heard (as in some transitions for example).

The formalism also enables to describe more layers, such as mel/mel/mel/mel, acc/acc, or mel/acc/acc.

Limitations. This modeling of texture is often ambiguous, and has limitations. The distinction between melody and accompaniment is questionable. Some melodies can contain repeated notes, arpeggiated motives, and strongly imply some harmony. Limiting the role of the accompaniment to harmony and rhythm is also over-simplified. Moreover, some textural gestures are not modeled here, such as upwards or downwards scales. Finally, what Piston calls "complex textures" (and what is perhaps the most interesting), interleaving different layers [15, p. 405], can not




```
 1,  75 : mel/acc (SAp / TB)
 8,  82 : mel/acc (SA / TBp)
 9,  83 : mel/acc (SAp / T syncopation, B)
13,  87 : mel/acc (SAp / TB imitation)
19,  93 : mel/acc (SAo / TBh)
20,  94 : mel/acc (SAp / TBhr)
21,  95 : imitation/acc (SA / TB)
25,  99 : imitation/acc (SA / TB)
    102 : mel/acc (S / ATB)
29, 103 : mel/acc (S / ATBhr)
30      : mel/acc (S / ATBh)
    104 : mel/acc (SAh/ TBh)
31, 105 : mel/acc (S / ATBh sparse chords)
33, 107 : acc/mel/acc (S / ATp / B)
  ...
```

Figure 1. Beginning of the string quartet K. 157 no. 4 by W. A. Mozart, with the ground truth analysis describing textures. We label as S / A / T / B (sopran / alt / tenor / bass) the four instruments (violin I / violin II / viola / cello). The first eight measures have a melodic layer "SAp" made by a parallel motion (with thirds), however the parallel motion has some exceptions (unison on c, strong beat on measures 1 and 8, and small interruption at the beginning of measure 5).

always be modeled by this way. Nevertheless, the above formalization is founded for most music of the Classical and of the Romantic period, and corresponds to a way of melody/accompaniment writing.

2.2 A Ground Truth for Texture

We manually analyzed the texture on 10 first movements of string quartets: the six quartets op. 33 by Haydn, three early quartets by Mozart (K. 80 no. 1, K. 155 no. 2 and K. 157 no. 4), and the quartet op. 125 no. 1 by Schubert. These pieces covered the textural features we wanted to elucidate. We segmented each piece into non-overlapping segments based only on texture information, using the formalism described above.

It is difficult to agree on the signifiance on short segments and on their boundaries. Here we choose to report the texture with a resolution of *one measure*: we consider only segments during at least one measure (or filling the most part of the measure), and round the boundaries of these segments to bar lines.

We identified 691 segments in the ten pieces, and Table 1 details the repartition of these segments. The ground truth file is available at `www.algomus.fr/truth`, and Figure 1 shows the analysis for the beginning of the string quartet K. 157 no. 4 by Mozart.

The segments are further precised by the involved voices and the h/p/o/u relations. For example, focusing on the most represented category "mel/acc", there are 254 segments labelled either "S / ATB" or "S / ATBh" (melodic layer at the first violin) and 81 segments labelled "SAp / TB" or "SAp / TBh" (melodic layer at the two violins, in a parallel move). Note that h/p/o/u relations were evaluated here in a subjective way. The segments may contain some small interruptions that do not alter the general perception of the h/p/o/u relation.

3. DISCOVERING SYNCHRONIZED LAYERS

We now try to provide a computational analysis of texture starting from a polyphonic score where voices are not separated. A first idea is to *first segment the score into layers by perception principles, and then to try to qualify some of these layers.* One can for example use the algorithm of [16] to segment the musical pieces into layers (called "streams"). This algorithm relies on a distance matrix, which tells for each possible pair of notes whether they are likely to belong to the same layer. The distance between two notes is computed according to their synchronicity, pitch and onset proximity (among others criteria); then for each note, the list of its k-nearest neighbors is established. Finally, notes are gathered in clusters. A melodic stream can be split into several small chunks, since the diversity of melodies does not always ensure coherency within clusters; working on larger layers encompass them all. Even if this approach produces good results in segmentation, many layers are still too scattered to be detected as full melodic or accompaniment layers. Nonetheless, classification algorithms could label some of these layers as melodies or accompaniments, or even detect the type of the accompaniment.

The second idea, that we will develop in this paper, is to *detect directly noteworthy layers from the polyphonic data*. Here, we choose to focus on perceptually significant relations based on homorhythmic features. The following paragraphs define the notion of *synchronized layers*, that is sequences of notes related by some homorhythmy relation (h/p/o/u), and show how to compute them.

3.1 Definitions: Synchronized Layers

A *note* n is given as a triplet $(n.pitch, n.start, n.end)$, where $n.pitch$ belongs to a pitch scale (that can be defined diatonically or by semitones), and $n.start$ and $n.end$ are two positions with $n.start < n.end$. Two notes n and m are *synchronized* (denoted by $n \equiv_h m$) if they have the same start and the same end.

A *synchronized layer* (SL) is a set of two sequences of consecutive synchronized notes (in other words, these sequences correspond to two "voices" in homorhythmy). Formally, two sequences of notes $n_1, n_2 ... n_k$ and $m_1, m_2 ... m_k$ form a synchronized layer when:

- for all i in $\{1, \ldots, k\}$, $\quad n_i.start = m_i.start$

	tonality	length	mel/acc	mel/mel	acc/mel/acc	acc/mel	mel	acc	others	h	p	o	u
Haydn op. 33 no. 1	B minor	91m	38	0	8	1	0	0	5	19	21	1	0
Haydn op. 33 no. 2	E-flat major	95m	37	0	2	4	0	0	7	34	13	0	0
Haydn op. 33 no. 3	C major	172m	68	0	0	0	3	13	6	29	50	1	0
Haydn op. 33 no. 4	B-flat major	90m	25	0	1	0	0	0	6	16	6	0	0
Haydn op. 33 no. 5	G major	305m	68	0	3	4	7	0	5	56	45	6	0
Haydn op. 33 no. 6	D major	168m	58	0	1	3	15	0	29	43	42	0	2
Mozart K. 80 no. 1	G major	67m	36	4	6	0	2	0	3	5	33	3	0
Mozart K. 155 no. 2	D major	119m	51	0	0	0	1	0	0	21	32	4	1
Mozart K. 157 no. 4	C major	126m	29	0	3	6	2	0	7	18	22	2	0
Schubert op. 125 no. 1	E-flat major	255m	102	0	0	0	20	2	0	54	8	46	2
		1488m	512	4	24	18	50	15	68	295	272	63	5

Table 1. Number of segments in the ground truth analysis of the ten string quartets (first movements), and number of h/p/o/u labels further describing these layers.

- for all i in $\{1, \ldots, k\}$, $\quad n_i.end = m_i.end$
- for all i in $\{1, \ldots, k-1\}$, $\quad n_i.end = n_{i+1}.start$

This definition can be extended to any number of voices. As p/o/u relations have a strong musical signification, we want to be able to enforce them. One can thus restrain the relation $\equiv_h$, considering the pitch information:

- we denote $n \equiv_\delta m$ if the interval between the two notes n and m is δ. The nature of the interval δ depends on the pitch model: for example, the interval can be diatonic, such as in "third" (minor or major), or an approximation over the semitone information, such as in "3 or 4 semitones". Some synchronized layers with $\equiv_\delta$ relations correspond to parallel motions;
- we denote $n \equiv_o m$ if notes n and m are separated by any number of octaves;
- we denote $n \equiv_u m$ where there is an exact equality of pitches (unison).

Given a relation $\equiv \in \{\equiv_h, \equiv_\delta, \equiv_o, \equiv_u\}$, we say that a synchronized layer respects the relation $\equiv$ if its notes are pairwise related according to this relation. The relation $\equiv_h$ is an equivalence relation, but the restrained relations do not need to be equivalence relations: Some $\equiv_\delta$ relations are not transitive.

For example, in Figure 1, there is between voices S and A (corresponding to violins I and II), in the first two measures:

- a synchronized layer ($\equiv_h$) on the two measures;
- and a synchronized layer ($\equiv_{\text{third}}$) on the two measures, except the first note.

Note that this does not correspond exactly to the "musical" ground truth (parallel move on at least the first four measures) because of some rests and of the first synchronized notes that are not in thirds.

A synchronized layer is *maximal* if it is not strictly included in another synchronized layer. Note that two maximal synchronized layers can be overlapping, if they are not synchronized. Note also that the number of synchronized layers may grow exponentially with the number of notes.

3.2 Detection of a Unique Synchronized Layer

A very noticeable textural effect is when *all* voices use the same texture at the same time. For example, a sudden striking unison raises the listener's attention. We can first check if all notes in a segment of the score belong to a unique synchronized layer (within some relation). For example, we consider that all voices are in octave doubling or unison if it lasts at least two quarters.

3.3 Detection of Maximal Synchronized Layers

In the general case, the texture has several layers, and the goal is thus to extract layers using *some* of the notes. Remember that we work on files where polyphony is not separated into voices: moreover, it is not always possible to extract voices from a polyphonic score, for example on piano music. We want to extract maximal synchronized layers. However, as their number may grow exponentially with the number of notes, we will compute only the start and end positions of maximal synchronized layers.

The algorithm is a kind of 1-dimension interval chaining [14]. The idea is as follows. Recursively, two voices $n_1, \ldots, n_k$ and $m_1, \ldots, m_k$ are synchronized if and only if $n_1, \ldots, n_{k-1}$ and $m_1, \ldots, m_{k-1}$ are synchronized, n_k and m_k are synchronized and finally $n_{k-1}.end = n_k.start$. Formally, the algorithm is described by the following:

Step 1. *Compute a table with left-maximal SL.* Build the table $\textit{leftmost\_start}_\equiv[j]$ containing, for each ending position j, the leftmost starting position of a SL respecting $\equiv$ ending in j. This can be done by dynamic programming with the following recurrence:

$$\textit{leftmost\_start}_\equiv[j] = \begin{cases} \min\{\textit{leftmost\_start}_\equiv[i] \mid i \in S_\equiv(j)\} \\ \quad \text{if } S_\equiv(j) \text{ is not empty} \\ j \quad \text{if } S_\equiv(j) \text{ is empty} \end{cases}$$

where $S_\equiv(j)$ is the set of all starting positions of synchronized notes ending at j respecting the relation $\equiv$:

$$S_\equiv(j) = \left\{ n.start \;\middle|\; \begin{array}{l} \text{there are two different notes } n \equiv m \\ \text{such that } n.end = j \end{array} \right\}$$

Step 2. *Output only (left and right) maximal SL.* Output (i, j) with $i = \textit{leftmost\_start}_\equiv[j]$ for each j, such that $j = \max\{j_o \mid \textit{leftmost\_start}_\equiv[j_o] = \textit{leftmost\_start}_\equiv[j]\}$

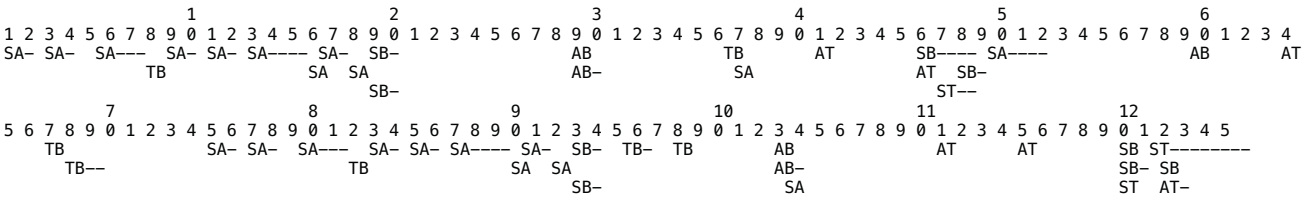


Figure 2. Result on the parallel move detection on the first movement of the string quartet K. 157 no. 4 by Mozart. The top lines display the measure numbers. The algorithm detects 52 synchronized layers respecting the $\equiv_p$ relation. 39 of these 52 layers overlap layers identified in the truth with p/o/u relations. The parallel motions are further identified by their voices (S / A / T / B), but this information is not used in the algorithm which works on non-separated polyphony.

The first step is done in $O(nk)$ time, where n is the number of notes and $k \leq n$ the maximal number of simultaneously sounding notes, so in $O(n^2)$ time. The second step is done in $O(n)$ time by browsing from right to left the table *leftmost_start*$_{\equiv}$, outputing values i when they are seen for the first time.

To actually retrieve intervals, we can store in the table *leftmost_start*$_{\equiv}[j]$ a pair (i, ℓ), where ℓ is the list of notes/intervals from which the set of SL can be built (this set may be very large, but not ℓ). The time complexity is now $O(n(k + w))$, where $w \leq n$ is the largest possible size of ℓ. Thus the time complexity is still in $O(n^2)$. This allows, in the second step, to filter the SL candidates according to additional criteria on ℓ.

Note finally that the definition of synchronized layer can be extended to include consecutive notes separated with rests. The same algorithm still applies, but the value of k rises to the maximum number of notes that can be linked in that way.

4. RESULTS AND DISCUSSION

We tested the proposed algorithm to look for synchronized layers respecting $\equiv_\delta$ relation (constant pitch interval, including parallel motion) on the ten pieces of our corpus given as `.krn` Humdrum files [8]. Although the pieces are string quartets, we consider them as non-separated polyphonic data, giving as input to the algorithm a single set of notes. The algorithm finds 434 layers. Figure 2 shows an example of the output of the algorithm. Globally, on the corpus, the algorithm labels 797 measures (that is 53.6% of the length) as synchronized layers.

Evaluation against the truth. There are in the truth 354 layers with p/o/u relations: mainly parallel moves, and some octave doubling and unisons. As discussed earlier, these layers reported in the truth correspond to a musical interpretation: they are not as formalized as our definition of synchronized layer. Moreover, less information is provided by the algorithm than in the ground truth: when a parallel motion is found, the algorithm cannot provide at which voice/instrument it appears, since we worked from polyphonic data with no voice separation.

Nevertheless, we compared the layers predicted by the algorithms with the ones of the truth. Results are summarized on Table 2. A computed layer is marked as true positive (TP) as soon as it overlaps a p/o/u layer of the truth.

356 of the 434 computed synchronized layers are overlapping the p/o/u layers of the truth, thus 82.0% of the computed synchronized layers are (at least partially) musically relevant. These 356 layers map to 194 p/o/u layers in the truth (among 340, that is a sensitivity of 58.0%): a majority of the parallel moves described in the truth are found by the algorithm.

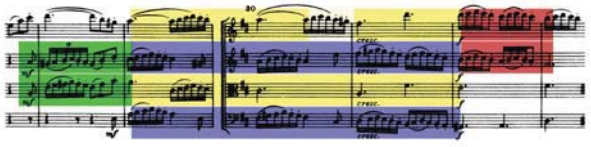


Figure 3. Haydn, op. 33 no. 6, m. 28-33. The truth contains four parallel moves.

Merged parallel moves. If one restricts to layers where borders coincide with the ones in the truth (same start, same end, with a tolerance of 2 quarters), the number of truth layers found falls from 194 to 117. This is because the algorithm often merge consecutive parallel moves. An example of this drawback is depicted on Figure 3. Here a melody is played in imitation, resulting in parallel moves involving all voices in turn. The algorithm detects a unique synchronized layer, which corresponds to a global perception but gives less information about the texture. We should remember here that the algorithm compute boundaries of synchronized layers and not actual instances, which would require some sort of voice separation and possibly generate a large number of instances.

False positives. Only 78 false positives are found by the algorithm. Many false positives (compared to the truth) are parallel moves detected inside a homorhythmy $\equiv_h$ relation between 3 ou 4 voices. In particular, the algorithm detects a parallel move as soon as there are sequences of repeated notes in at least two voices. This is the case in in op. 33 no. 4 by Haydn which contains many homorhythmies in repeated notes, for which we obtain 30 false positives. Even focusing on layers with a real "move", false positive could also appear between a third voice and two voices with repeated notes. Further research should be carried to discard these false positives either in the algorithm or at a later filtering stage.

	hits length	hits	TP	FP	truth-overlap	truth-exact
Haydn op. 33 no. 1	40m (44%)	37	32 (86.5%)	5	14 / 22 (63.6%)	7 / 22
Haydn op. 33 no. 2	21m (22%)	17	15 (88.2%)	2	7 / 13 (53.9%)	7 / 13
Haydn op. 33 no. 3	73m (42%)	48	44 (91.7%)	4	27 / 51 (52.9%)	15 / 51
Haydn op. 33 no. 4	19m (21%)	47	17 (36.2%)	30	5 / 6 (83.3%)	3 / 6
Haydn op. 33 no. 5	235m (77%)	58	47 (81.0%)	11	27 / 51 (52.9%)	11 / 51
Haydn op. 33 no. 6	63m (37%)	24	21 (87.5%)	3	19 / 44 (43.2%)	11 / 44
Mozart K. 80 no. 1	45m (67%)	27	26 (96.3%)	1	20 / 36 (55.6%)	14 / 36
Mozart K. 155 no. 2	76m (64%)	46	44 (95.7%)	2	27 / 37 (73.0%)	15 / 37
Mozart K. 157 no. 4	62m (49%)	52	39 (75.0%)	13	15 / 24 (62.5%)	8 / 24
Schubert op. 125 no. 1	163m (64%)	78	71 (91.0%)	7	33 / 56 (58.9%)	20 / 56
	797m (54%)	434	356 (82.0%)	78	194 / 340 (57.1%)	111 / 340

Table 2. Evaluation of the algorithm on the ten string quartets of our corpus. The columns TP and FP show respectively the number of true and false positives, when comparing computed parallel moves with the truth. The columns truth-overlap shows the number of truth parallel moves that were matched by this way. The column truth-exact restricts these matchings to computed parallel moves for which borders coincide to the ones in the truth (tolerance: two quarters).

5. CONCLUSION AND PERSPECTIVES

We proposed a formalization of texture in Western classical instrumental music, by describing melodic or accompaniment "layers" with perceptive features (h/p/o/u relations). We provided a first algorithm able to detect some of these layers inside a polyphonic score where tracks are not separated, and tested it on 10 first movements of string quartets. The algorithm detects a large part of the parallel moves found by manual analysis. We believe that other algorithms implementing textural features, beyond h/p/o/u relations, should be designed to improve computational music analysis. The corpus should also be extended, for example with music from other periods or piano scores.

Finally, we believe that this search of texture, combined with other elements such as patterns and harmony, will improve algorithms for music structuration. The ten pieces of our corpus have a *sonata form* structure. The tension created by the exposition and the development is resolved during the recapitulation, and textural elements contribute to this tension and its resolution [10]. For example, the medial caesura (MC), before the beginning of theme S, has strong textural characteristics [6]. Textural elements predicted by algorithms could thus help the structural segmentation.

6. REFERENCES

[1] A. S. Bregman. *Auditory scene analysis*. Bradford, Cambridge, 1990.

[2] E. Cambouropoulos. Voice separation: theoretical, perceptual and computational perspectives. In *Int. Conf. on Music Perception and Cognition (ICMPC)*, 2006.

[3] R. B. Dannenberg and M. Goto. *Handbook of Signal Processing in Acoustics*, chapter Music Structure Analysis, pages 305–331. Springer, 2008.

[4] D. Deutsch and J. Feroe. The internal representation of pitch sequences in tonal music. *Psychological Review*, 88(6):503–522, 1981.

[5] J. M. Dunsby. Considerations of texture. *Music and letters*, 70(1):46–57, 1989.

[6] J. Hepokoski and W. Darcy. The medial caesura and its role in the eighteenth-century sonata exposition. *Music Theory Spectrum*, 19(2):115–154, 1997.

[7] D. Huron. Characterizing musical textures. In *Int. Computer Music Conf. (ICMC)*, pages 131–134, 1989.

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

[9] A. Klapuri and M. Davy. *Signal Processing Methods for Music Transcription*. Springer, 2006.

[10] J. M. Levy. Texture as a sign in classic and early romantic music. *J. of the American Musicological Society*, 35(3):482–531, 1982.

[11] C. McKay. *Automatic music classification with jMIR*. PhD thesis, McGill University, 2010.

[12] C. McKay and I. Fujinaga. jSymbolic: A feature extractor for MIDI files. In *Int. Computer Music Conf. (ICMC)*, pages 302–305, 2006.

[13] Q. R. Nordgren. A measure of textural patterns and strengths. *J. of Music Theory*, 4(1):19–31, Apr. 1960.

[14] E. Ohlebusch and M. I. Abouelhoda. *Handbook of Computational Molecular Biology*, chapter Chaining Algorithms and Applications in Comparative Genomics. 2005.

[15] W. Piston. *Orchestration*. Norton, 1955.

[16] D. Rafailidis, A. Nanopoulos, Y. Manolopoulos, and E. Cambouropoulos. Detection of stream segments in symbolic musical data. In *Int. Conf. on Music Information Retrieval (ISMIR)*, pages 83–88, 2008.

[17] L. Rowell. *Thinking about Music: An Introduction to the Philosophy of Music*. Univ. of Massachusetts, 1985.

[18] N. Saint-Arnaud and K. Popat. Computational auditory scene analysis. chapter Analysis and Synthesis of Sound Textures, pages 293–308. Erlbaum, 1998.

[19] G. Strobl, G. Eckel, and D. Rocchesso. Sound texture modeling: a survey. In *Sound and Music Computing (SMC)*, 2006.

[20] A. Tenkanen and F. Gualda. Detecting changes in musical texture. In *Int. Workshop on Machine Learning and Music*, 2008.

COMPUTATIONAL MODELS FOR PERCEIVED MELODIC SIMILARITY IN A CAPPELLA FLAMENCO SINGING

N. Kroher, E. Gómez
Universitat Pompeu Fabra
emilia.gomez@upf.edu, nadine.kroher@upf.edu

C. Guastavino
McGill University & CIRMMT
catherine.guastavino@mcgill.ca

F. Gómez
Technical University of Madrid
fmartin@eui.upm.es

J. Bonada
Universitat Pompeu Fabra
jordi.bonada@upf.edu

ABSTRACT

The present study investigates the mechanisms involved in the perception of melodic similarity in the context of a cappella flamenco singing performances. Flamenco songs belonging to the same style are characterized by a common melodic skeleton, which is subject to spontaneous improvisation containing strong prolongations and ornamentations. For our research we collected human similarity judgements from naïve and expert listeners who listened to audio recordings of a cappella flamenco performances as well as synthesized versions of the same songs. We furthermore calculated distances from manually extracted high-level descriptors defined by flamenco experts. The suitability of a set of computational melodic similarity measures was evaluated by analyzing the correlation between computed similarity and human ratings. We observed significant differences between listener groups and stimuli types. Furthermore, we observed a high correlation between human ratings and similarities computed from features from flamenco experts. We also observed that computational models based on temporal deviation, dynamics and ornamentation are better suited to model perceived similarity for this material than models based on chroma distance.

1. INTRODUCTION

The task of modeling perceived melodic similarity among music pieces is a multi-dimensional task whose complexity increases when human judgements are influenced by implicit knowledge about genre-specific musicological aspects and contextual information. Nevertheless, such computational models are of utmost importance for automatic similarity retrieval and recommendation systems in large music databases. Furthermore, analysis of melodic similarity among large amounts of data can provide important clues for musicological studies regarding style classification, similarity and evolution. In the past, numerous approaches have focused on melodic similarity measures, mainly computed from automatically aligned score-like representations. For a complete review of symbolic note similarity measures we refer the reader to [1]. Several previous studies have related computational measures to human ratings. In an extensive study in [14], expert ratings of similarity between western pop songs and generated variants were compared to 34 computational measures. The best correlation was observed for a hybrid method combining various weighted distance measures, which is successfully used to automatically retrieve variants of a given melody from a folk song database. In similar studies, human similarity ratings were compared to transportation distances [16] and statistical descriptors related to tone, interval and note duration distribution [17]. In order to gain a deeper insight into the perception process of melodic similarity, Volk and van Kranenburg studied the relationship between musical features and human similarity-based categorization, where a large collection of folk songs was manually categorized into tune families [15]. Furthermore, human similarity judgement based on various musical facets were gathered. Results indicate that songs perceived as similar tend to show strong similarities in rhythm, pitch contour and contained melodic motifs, whereas the individual importance of these criteria varies among the data. When dealing with audio recordings for which no score is available, it seems natural to focus on the alignment and comparison of the time-frequency representation of the melodic contour. In the context of singing voice assessment, Molina et al. used *dynamic time warping* to align fundamental frequency contours and calculate melodic and rhythmic deviations between them [2].

Despite the growing interest in non-Western music traditions, most algorithms are designed and evaluated on Western commercial music. In a first genre-specific approach to melodic similarity in flamenco music, Cabrera et al. computed melodic similarity among a cappella singing performances from automatic descriptions [3]. The two standard distance measures implemented, the *edit* distance and

the correlation between pitch and interval histograms, obtained rather poor results when compared to expert judgements. As proposed by Mora et al., better results for intra- and inter-style similarity can be obtained for a similarity measure based on manually extracted high-level features (i.e., the direction of melodic movement in a specific part of the performance) [4]. Such studies elucidate the need for exploration of particular characteristics of non-Western music genres and the adaptation of existing music information retrieval systems to such styles.

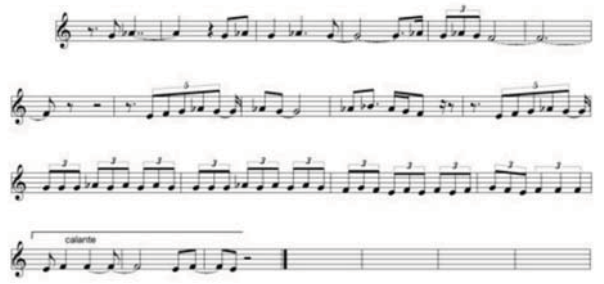
The present study addresses perceived melodic similarity in a cappella flamenco singing from different standpoints: with the aim of gaining insight into the mechanisms involved in perceiving melodies as more or less similar, we gathered similarity ratings among performances of the same style from naïve listeners as well as flamenco experts and analyzed them in terms of intra-subject and intra-group agreement. In order to isolate the melody from other variables such as lyrics, expression and dynamics, we gathered the same ratings for synthesized melodic contours. We furthermore evaluated three computational models for melodic similarity by analyzing the correlation between computed similarity and human ratings. We compared the results to distances computed from manually extracted high-level features defined by experts in the field. The rest of the paper is organized as follows. In Section 2 we provide background information on flamenco music and the *martinete* style, which is the focus of this study. We give a detailed description of the database used in the present experiment in Section 3. Section 4 summarizes the methodology of the listening experiments, the extracted high-level features and the implemented computational similarity models. We give the results of the correlation analysis in Section 5 and conclude our study in Section 6.

2. BACKGROUND

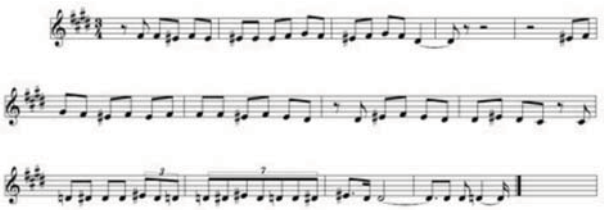
Flamenco is an oral tradition whose roots are as diverse as the cultural influences of its area of origin, Andalusia, a region in southern Spain. Its characteristics are influenced by music traditions of a variety of immigrants and colonizations that settled in the area throughout the past centuries, among them Visigoths, Arabs, Jews and to a large extend gipsies, who decisively contributed to shape the genre as we know it today. For a comprehensive and complete study on history and style, we refer to [5–7]. Flamenco germinated and nourished mainly from the singing tradition and until now the singing voice represents its central element, usually accompanied by the guitar and rhythmic hand-clapping. In the flamenco jargon, songs, but also styles, are referred to as *cantes*.

2.1 The flamenco singing voice

Flamenco singing performances are usually spontaneous and highly improvisational. Songs are passed from generation to generation and only rarely manually transcribed. Even though there is no distinct ideal for timbre and several voice types can be identified, the flamenco singing voice can be generally characterized as matt, breathy, and containing few high frequency harmonics. Moreover, singers usually lack the singer's formant [13]. Melodic movements appear mainly in conjunct degrees within a small pitch range (*tessitura*) of a major sixth interval and are characterized by insistence on recitative notes. Furthermore, singers use a large amount melisma, microtonal ornamentation and pitch glides during note attacks [4].



(a) Performance by *Antonio Mairena*



(b) Performance by *Chano Lobato*

Figure 1. Manual transcriptions of performances a *debla* "En el barrio de Triana"; Transcription: Joaquin Mora

2.2 The flamenco martinete

Martinete is considered one of the oldest styles and forms part of the sub-genre of the *tonás*, a group of unaccompanied singing styles, or *cantes*. As in other *cantes*, songs belonging to *martinete* style are characterized by a common melodic skeleton, which is subject to strong spontaneous ornamentation and expressive prolongations. The untrained listener might perceive two performances of the same *cante* as very different and the fact that they belong to the same style is not obvious at all. To illustrate this principle, Figure 1 shows the transcription of two a cappella performances in Western music notation, both belonging to the same style (*debla*) [4].

Furthermore, the *martinete* is characterized by a solemn performance in slow tempo with free rhythmic interpretation. Traditionally, the voice is accompanied by hammer strokes on an anvil. The tonality corresponds mainly to the major mode, whereas the third scale degree may be lowered occasionally, converting the scale to the minor mode.

3. MUSIC COLLECTION

In consultation with flamenco experts, we gathered 12 recordings of *martinete* performances, covering the most representative singers of this style. This dataset represents

Singer	Percussion
Antonio Mairena	No
Chano Lobato	No
Chocolate	Yes
Jacinto Almadén	No
Jesus Heredia	No
Manuel Simón	Yes
Miguel Vargas	No
Naranjito	No
Paco de Lucia	No
Talegon de Córdoba	Yes
Tomás Pavón	No
Turronero	No

Table 1. Dataset containing 12 *martinete* performances.

a subset of the *tonás* [1] dataset, which contains a total of 56 *martinete* recordings. The average duration of the extracted excerpts containing the first verse is approximately 20 seconds. We limited our study to such a small set, mainly due to the duration of the listening experiment. As an additional stimuli for the listening experiments, we furthermore created synthesized versions of all excerpts. We used the method described in [8] to extract fundamental frequency and energy envelopes and re-synthesize with a sinusoid.

We selected the first verse of each recording, containing the characteristic exposition of the melodic skeleton. Although some *martinete* recordings contain additional accompaniment (guitar, bowing string or wind instruments), we limited our selection to a cappella recordings without rhythmic accompaniment or with very sparse one, as it is found traditionally. We intentionally incorporated a wide range of interpretation characteristics, regarding richness in ornamentation, tempo, articulation and lyrics. Among the singers listed in Table 1, *Tomás Pavón* is to be mentioned as the most influential artist in the a cappella singing styles, performing the *martinete* in an exemplar way. Furthermore, *Antonio Mairena* and *Chocolate* are thought to be the main references for their singing abilities and knowledge of the singing styles. *Chano Lobato* omits some of the basic notes during the melodic exposition and the performance has been included as an example of strong deviation in the melodic interpretation.

4. METHODOLOGY

4.1 Human similarity ratings

In order to obtain a ground truth for perceived melodic similarity among the selected excerpts, we conduct a listening experiment in Montreal (Canada) with 24 naïve listeners with little or no previous exposure to flamenco and in Sevilla (Spain) with 3 experts, as described in [9]. After evaluating various experiment designs (i.e. pair-wise comparison), we decided to collect the similarity ratings in a free sorting task [19]. Using the *sonic mapper* [2] software, subjects were asked to create groups of similar interpretations, leaving the number of groups open. The participants were explicitly instructed to focus on the melody only, neglecting differences in voice timbre, lyrics, percussion accompaniment and sound quality. Nevertheless, in order to isolate the melodic line as a similarity criterion, the experiment had also been conducted with the synthesized versions of the excerpts described above. For each excerpt we extracted the fundamental frequency as described in [8] with a window length of 33 ms and a hop size of 0.72 ms. The pitch contour was synthesized with a single sine wave. A similarity matrix was computed based on the number of times a pair of performances had been grouped together. We compared individual participants' similarity matrices using *Mantel* tests. The *Mantel* test can be considered as the most widely used method to account for distance correlations [12]. We used *zt*, a simple tool for *Mantel* test, developed by Bonnet and Vande Peer [18]., and measured the correlation between participant matrices. We observed that the average correlation for novices is $\mu = 0.0824$, with a $\sigma = 0.2109$ and the average p-value: $\mu = 0.3391$, $\sigma = 0.2139$ (min=0.002). This indicates a very low agreement among them, and indicates differences in perception of melodic similarity depending on the listener's background. Although we should take these results with caution given the small number of experts, we found higher correlation values among them, with an average correlation $\mu = 0.1891$, and $\sigma = 0.1170$. For a detailed description of the procedure and the analysis, we refer to [9].

4.2 Manually extracted high-level features

We manually extracted six high-level features defined by experts in the field. As illustrated above, two *cantes* having the same main notes and different ornamentation would be perceived as the same *cante* by a flamenco aficionado. This fact makes the automatic computation of the features unfeasible. Because of that, we had to rely on manual extraction.

The high-level features were the following.

1. Repetition of the first hemistich. A hemistich is half-line of a verse; the presence of this repetition is important in these *cantes*.
2. Clivis/flexa at the end of the first hemistich. A clivis is a descending melodic movement. Here it refers to a descending melodic contour between main notes. Again, the ornamentation is not taken into account when detecting the presence of the clivis.
3. Highest scale degree in the two first hemistichs. The highest scale degree reached during the *cante* is an important feature.
4. Frequency of the highest degree in the second hemistich. How many times that highest degree is reached is also significant.

[1] http://mtg.upf.edu/download/datasets/tonas

[2] http://www.music.mcgill.ca/ gary/mapper/

5. Final note of the second hemistich.

6. Duration (fast / regular / slow).

A distance matrix was obtained by calculating the Euclidean distance among the feature vectors. The feature vectors were mostly composed of categorical data and we used a standardized Euclidean distance. For a detailed explanation of the descriptors and their musicological background, the reader is referred to [4].

4.3 Computational similarity measures

We implemented three computational measures based on fundamental frequency envelopes and automatic transcriptions and evaluated their suitability for modeling the perceived melodic similarity by analyzing the correlation between computed distance matrices and human judgements. The fundamental frequency contours as well as the automatically generated symbolic note representations were obtained using the system described in [8].

4.3.1 Dynamic time warping alignment

Similar to [2] we used a *dynamic time warping* algorithm to align melodies and estimate their rhythmic and pitch similarity. Since vocal vibrato and microtonal ornamentations strongly influence the cost matrix, we instead align continuous contours of quantized pitch values obtained with the automatic transcription described in [8]. The cost matrix M describes the squared frequency deviation between all possible combinations of time frames between the two analyzed contours $f0_1$ and $f0_2$, where α is a constraint limiting the maximum cost:

$$M_{i,j} = min((f0_1[i] - f0_2[j])^2, \alpha) \quad (1)$$

The *dynamic time warping algorithm* determines the optimal path among the matrix M from first to last frame. The deviation of the slope of the path p with length N from the diagonal path gives a measure for temporal deviation $(DTWtemporal)$,

$$\Delta_{temp} = \frac{\sum_{i=1}^{N}(p[i] - p_{diag}[i])^2}{N} \quad (2)$$

while the average over its elements defines the pitch deviation $(DTWpitch)$:

$$\Delta_{pitch} = \frac{\sum_{i=1}^{N} p[i]}{N}. \quad (3)$$

We used a *MATLAB* implementation [3], which extends the algorithm with several restrictions in order to obtain a musically meaningful temporal alignment. Figure 2 shows the cost matrix and Figure 3 the unaligned and aligned pitch sequences.

[3] http://www.ee.columbia.edu/ dpwe/resources/matlab/dtw/

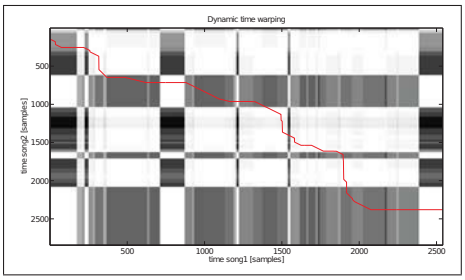


Figure 2. Dynamic time warping: Cost matrix and optimal path.

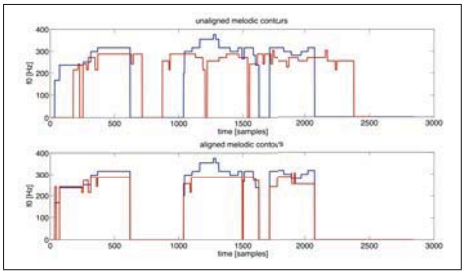


Figure 3. Unaligned (top) and aligned (bottom) melodic contours.

4.3.2 Global performance descriptors

As described in [10], we extracted a total of 13 global descriptors from automatic transcriptions and computed a similarity matrix based on the Euclidean distance among feature vectors. In order to determine the most suitable descriptors for this task, we analyzed the *phylogenetic tree* (Figure 4) computed from the distance matrix of expert similarity ratings. Here, we identify two main clusters, at large distance from each other.

Using these two clusters as classes in a classification task, we perform a support vector machine (SVM) subset selection in order to identify the descriptors that are best suited to distinguish the two clusters. We accordingly extracted the six best ranked descriptors for all songs and computed the similarity matrix from the Euclidean distances among feature vectors. The extracted descriptors are summarized below:

1. **Amount of silence**: Percentage of silent frames.

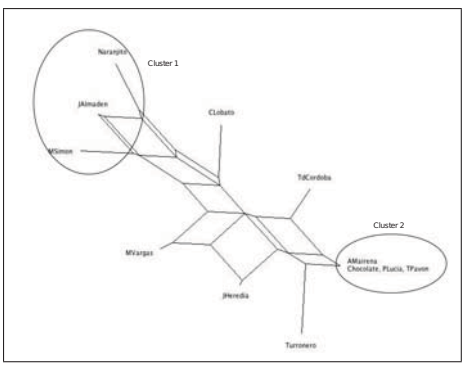


Figure 4. Phylogenetic tree generated from expert similarity judgements.

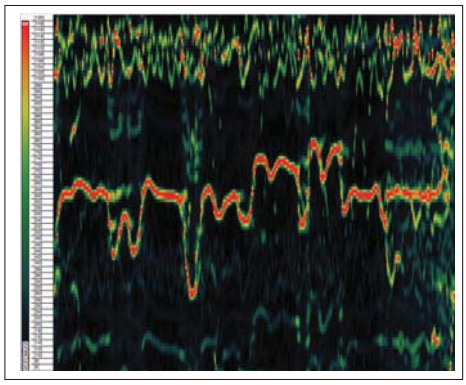


Figure 5. Harmonic pitch class profile for a sung phrase with a resolution of 12 bins per semitone.

2. **Average note duration** in seconds.
3. **Note duration fluctuation**: Standard deviation of the note duration in seconds.
4. **Average volume** of the notes relative to the normalized maximum.
5. **Volume fluctuation**: standard deviation of the note volume relative to normalized maximum.
6. **Amount of ornamentation**: Average per-frame distance in [Hz] between the quantized note value and the fundamental frequency contour.

4.3.3 Chroma similarity

We implemented a similarity measure presented in [11] in the context of cover identification: First, the harmonic pitch class profiles (HPCP) are extracted on a global and a frame basis. The resulting pitch class histogram describes the relative strength of the 12 pitch classes of the equal-tempered scale. HPCPs are robust to detuning as well as variation in timbre and dynamics. After adjusting the key of one sequence to the other, a binary similarity matrix is computed based on the frame-wise extracted HPCPs. Again, dynamic time warping was used to find the best possible path among the similarity matrix. For a detailed description of the algorithm, we refer the reader to [11].

4.4 Evaluation

We evaluated the suitability of the computational models for this task by analyzing the correlation between computed similarity and human ratings. A common method to evaluate a possible relation between two distance matrices is the Mantel test [12]: first, the linear correlation between two matrices is measured with the *Pearson correlation*, which gives a value r between -1 and 1. A strong correlation is indicated by a value significantly different from zero. To verify that a relation exists, the value is compared to correlations to permuted versions of the matrices. Here, 10000 random permutations are performed. The *confidence value* p corresponds to the proportion of permutations giving a higher correlation than the original matrix. Consequently, a confidence value close to zero confirms an existing correlation.

5. RESULTS

Figure 6 shows the comparison of the computed similarity measures by means of correlation r and confidence value p for the different participant groups and stimuli types. We first note that the distance measure obtained from manually extracted high-level descriptors seems to reflect best the perceived melodic similarity for both, expert and naïve listeners. Even though the computed similarity correlates strongly with the expert ratings, the also strong relation with the non-expert similarity judgments is still surprising, given the fact that the descriptors are based on rather abstract musicological concepts. We furthermore find a weaker, but still significant correlation between human ratings and the temporal deviation measure of the *dynamic time warping* algorithm as well as the vector distance among performance descriptors. On the other hand, we find no relation between human ratings and the pitch deviation from the dynamically aligned sequences, nor the chroma similarity measure. Given the fact that the selected performance descriptors are related to dynamic and temporal behavior and ornamentation and the temporal deviation measure does not consider the absolute pitch difference of the aligned sequences, we can speculate that for the given material these factors influence perceived similarity stronger than differences in the pitch progression. *Martinete* presents a particularly interesting case, since the skeleton of the melodic contour and at least its outer envelope is preserved throughout the performances. Notice also that in all cases the found correlation with the similarity ratings of real recordings is stronger than for the synthesized versions. Since none of the computational methods take voice timbre or lyrics into account, we can preclude that these factors influenced human judgement. It is however possible that it was more difficult for the listener to internalize these synthesized sequences compared to real recordings given their artificial nature and consequently judging similarity was more difficult and less precise.

6. CONCLUSIONS

The present study investigates the mechanisms involved in the perception of melodic similarity for the particular case of a cappella flamenco singing. We compared human judgements from experts and naïve listeners for audio recordings and synthesized melodic contours. Computational models are furthermore used to create distance matrices and evaluated based on their correlation with human ratings. We observed a significantly higher agreement among experts and a stronger correlation among computational models and the ratings based on real recordings than when comparing to ratings for synthesized melodies. Furthermore, we discover that models based on descriptors related to rhythm, dynamics and ornamentation are better suited to recreate similarity judgements than models based on absolute pitch distance. We obtained the highest corre-

Subject group	Expert listeners		Naive listener	
Stimuli type	*real*	*synth*	*real*	*synth*
High-level	r=0.585 p=0.001	r=0.424 p=0.001	r=0.429 p=0.000	r=0.202 p=0.054
DTW temporal	r=0.306 p=0.047	r=0.213 p=0.044	r=0.333 p=0.003	r=0.245 p=0.123
DTW pitch	r=-0.118 p=0.256	r=-0.094 p=0.224	r=-0.130 p=0.198	r=-0.096 p=0.204
Perform. descrip.	r=0.431 p=0.011	r=0.207 p=0.044	r=0.308 p=0.0123	r=0.176 p=0.061
Chroma	r=0.108 p=0.239	r=0.107 p=0.187	r=0.090 p=0.247	r=0.102 p=0.193

Figure 6. Correlation between computed similarity and human ratings. Statistically significant results are marked grey.

lation for both expert and non-expert ratings for a similarity measure computed from manually extracted high-level features. The problem of how to compute the high-level features automatically is still open. This problem is equivalent to that of automatically detecting ornamentation and main notes in a flamenco *cante*.

Acknowledgements
The authors would like to thank Joaquin Mora for providing the manual transcriptions and Joan Serrá for computing the chroma similarity measures. This research is partly funded by the COFLA (Proyectos de Excelencia de la Junta de Andalucia, P12-TIC-1362) and SIGMUS (Spanish Ministry of Economy and Competitiveness, TIN2012-36650) research projects as well as the PhD fellowship program of the Department of Information and Communication Technologies, Universitat Pompeu Fabra.

7. REFERENCES

[1] A. Marsden: "Interrogating Melodic Similarity: A Definitive Phenomenon or the Product of Interpretation?" *Journal of New Music Research*, Vol. 4, No. 44, pp. 323–335, 2012.

[2] E. Molina, I. Barbancho, E. Gómez, A. M. Barbanco, and L. J. Tardón: "Fundamental frequency alignment vs. note-based melodic similarity for singing voice assessment," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013.

[3] J. J. Cabrera, J. M. Díaz-Báñez, F. J. Escobar, E. Gómez, F. Gómez, and J. Mora: "Comparative Melodic Analysis of A Cappella Flamenco Cantes," *Proceedings of the Conference on Interdisciplinary Musicology*, 2008.

[4] J. Mora, F. Gómez, E. Gómez, F. J. Escobar, and J. M. Díaz-Báñez: "Melodic Characterization and Similarity in A Cappella Flamenco Cantes," *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR)*, 2010.

[5] J. Blas Vega, and M. Ríos Ruiz: *Diccionario enciclopdico ilustrado del flamenco*, Cinterco,1988.

[6] J. L. Navarro ,and M. Ropero: *Historia del flamenco*, Tartessos, 1995.

[7] J. M. Gamboa: *Una historia del flamenco*, Espasa-Calpe, 2005.

[8] E. Gómez, and J. Bonada: "Towards Computer-Assisted Flamenco Transcription: An Experimental Comparison of Automatic Transcription Algorithms as Applied to A Cappella Singing," *Computer Music Journal*, Vol. 37, No. 2, pp. 73-90, 2013.

[9] E. Gómez, C. Guastavino, F. Gómez, and J. Bonada: "Analyzing Melodic Similarity Judgements in Flamenco A Cappella Singing," *Proceedings of the International Conference on Music Perception and Cognition*, 2012.

[10] N. Kroher: *The Flamenco Cante: Automatic Characterization of Flamenco Singing by Analyzing Audio Recordings*, Master Thesis, Universitat Pompeu Fabra, 2013.

[11] J. Serra, E. Gómez, P Herrera, and X. Serra: "Chroma Binary Similarity and Local Alignment Applied to Cover Song Identification," *IEEE Transactions on Audio, Speech and Language Processing*, Vol. 16, No. 6, pp. 1138-1151, 2008..

[12] N. Mantel, and R. S. Valand: "A technique of non-parametric multivariate analysis," *Biometrics*, Vol. 26, pp. 547-558, 1970.

[13] J.Sundberg: "The acoustics of the singing voice," *Scientific American*, Vol. 236 (3), pp.104-116, 1977.

[14] D. Muellensiefen, and K. Frieler: "Modelling experts' notions of melodic similarity," *Musicae Scientiae*, Discussion Forum 4A, pp.183-210, 2007.

[15] A. Volk, and P. van Kranenburg: "Melodic similarity among folk songs: An annotation study on similarity-based categorization in music," *Musicae Scientiae*, 16 (3) pp.317-339, 2012.

[16] R. Typke, and F. Wiering: "Transportation distances and human perception of melodic similarity," *Musicae Scientiae*, Discussion Forum 4A, pp.153-181, 2007.

[17] T. Eerola, T Jaervinen, J. Louhivuori, and P. Toiviainen, : "Statistical Features and Perceived Similarity of Folk Melodies," *Music Perception*, 18 (3), pp.275-296, 2001.

[18] E. Bonnet, and Y. Van de Peer : "zt: a software tool for simple and partial Mantel tests," *Journal of Statistical software*, 7 (10), pp.1-12, 2002.

[19] B. Giordano, C. Guastavino, E. Murphy, M. Ogg, and B.K. Smith: "Comparison of Dissimilarity Estimation Methods". *Multivariate Behavioral Research*, 46, 1-33, 2011.

THE VIS FRAMEWORK: ANALYZING COUNTERPOINT IN LARGE DATASETS

Christopher Antila and Julie Cumming
McGill University
christopher@antila.ca; julie.cumming@mcgill.ca

ABSTRACT

The *VIS Framework for Music Analysis* is a modular Python library designed for "big data" queries in symbolic musical data. Initially created as a tool for studying musical style change in counterpoint, we have built on the `music21` and *pandas* libraries to provide the foundation for much more.

We describe the musicological needs that inspired the creation and growth of the VIS Framework, along with a survey of similar previous research. To demonstrate the effectiveness of our analytic approach and software, we present a sample query showing that the most commonly repeated contrapuntal patterns vary between three related style periods. We also emphasize our adaptation of typical n-gram-based research in music, our implementation strategy in VIS, and the flexibility of this approach for future researchers.

1. INTRODUCTION

1.1 Counterpoint

"The evolution of Western music can be characterized in terms of a dialectic between acceptable vertical sonorities on the one hand... and acceptable melodic motions on the other." [12] A full understanding of polyphonic music (with more than one voice or part) requires description in terms of this dialectic, which is called counterpoint. Whereas music information retrieval research (such as [6]) typically describes polyphonic music only in terms of vertical (simultaneous or harmonic) intervals, musicologists interested in contrapuntal patterns also want to know the horizontal (sequential or melodic) intervals in each voice that connect the vertical intervals. Since counterpoint describes how pitches in independent voices are combined in polyphonic music, a computerized approach to counterpoint analysis of symbolic music can provide a wealth of information to musicologists, who have previously relied primarily on prose descriptions of musical style. [1]

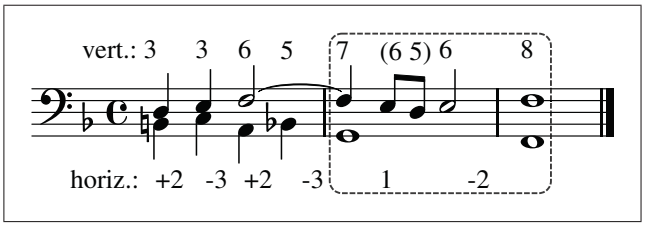


Figure 1. Symbolic score annotated with vertical and horizontal intervals. A common contrapuntal module appears in the box.

Figure 1 shows the counterpoint between two voices in a fragment of music. We annotated the vertical intervals above the score, and the lower voice's horizontal intervals below. Note that we show intervals by diatonic step size, counting number of lines and spaces between two notes, rather than semitones. We describe this contrapuntal module further in Section 2.1. By using intervals rather than note names, we can generalize patterns across pitch levels, so the same pattern may start on any pitch. For this article, we ignore interval quality (e.g., major or minor third) by using diatonic intervals (e.g., third), allowing generalization across mode and key. We do use interval quality for other queries—this is a choice available in VIS at runtime.

To allow computerized processing of contrapuntal patterns, we encode the counterpoint between two voices with alternating vertical and horizontal intervals. In Figure 1, the first two beats are "3 +2 3." We call these patterns interval n-grams, where n is the number of vertical intervals. Our n-gram notation system is easily intelligible to music theorists and musicologists, and allows us to stay close to musicology.

1.2 Research Questions

Until recently, musicologists' ability to accurately describe polyphonic textures was severely limited: any one person can learn only a limited amount of music in a lifetime, and the computer-based tools for describing or analyzing polyphonic music in detail are insufficiently precise for many repertoires. Descriptions of musical style and style change are often vague, derived from intuitive impressions and personal knowledge of repertoire rather than quantifiable

[1] We wish to thank the following people for their contributions: Natasha Dillabough, Ichiro Fujinaga, Jane Hatter, Jamie Klassen, Alexander Morgan, Catherine Motuz, Peter Schubert. The ELVIS Project was supported by the Digging into Data Challenge; the Canadian team responsible for the work described in this paper was additionally funded by the Social Sciences and Humanities Research Council of Canada.

evidence. Our project attempts the opposite by quantitatively describing musical style change using counterpoint.

We chose counterpoint not only because musicologists are already aware of its importance, but because it allows us to consider structure in all polyphonic music, which includes the majority of surviving Western music created after 1300. Our project's initial goal is to find the most frequently-repeated contrapuntal patterns for different periods, genres, and composers, to help form detailed, evidence-based descriptions of style periods and style change by knowing which features change over time and when. In addition, statistical models will allow a fresh approach to attribution problems (determining the composer of a piece where it is not otherwise known), by enabling us to describe some of the factors that distinguish a composer's style.

1.3 The VIS Framework

Our project's most important accomplishment is the VIS Framework—the software we developed to answer the research questions described above. (VIS stands for "vertical interval successions," which is a way to describe counterpoint). Currently VIS's primary function is to find contrapuntal patterns in symbolic music, recording them with the notation described above in Figure 1 so they may be counted. However, we designed the framework to allow a much broader set of queries, and we intend to add support for additional musical dimensions (like meter and harmony) as well as more complicated statistical experiments (like Markov-chain modeling).

We used the *Counterpoint Web App*, a Web-based user interface for VIS's counterpoint functionality, to run the analyses presented in this article. [2] Such Web-based software encourages musicologists to participate in data-driven analysis even if they are otherwise unable to program. The Web App's visual design, the use of musicologist-friendly terms and user workflows, and the ability to output analysis results on musical scores are significant advantages. At the same time, programmers are encouraged to download and extend the VIS Framework using its well-documented Python API. While our Framework provides a guide for structuring analysis workflows, each analytic step benefits from our integration of the `music21` and *pandas* libraries. Together, these allow analytic approaches more amenable to musicians and statisticians, respectively. [3]

2. BACKGROUND

2.1 Contrapuntal Modules

A *contrapuntal module* is a repeated contrapuntal pattern made from a series of vertical (harmonic) and horizontal (melodic) intervals—a repeated interval n-gram. [11] We are primarily interested in the frequency and nature of two-voice contrapuntal modules. VIS allows us to computerize tedious score analysis previously done by hand, as when Peter Schubert identified modules in Palestrina. [13] While two-voice contrapuntal modules are the primary structural element of much Renaissance music, we can find contrapuntal modules in nearly all polyphonic music, so our software and research strategies will be useful for a wide range of music. [3]

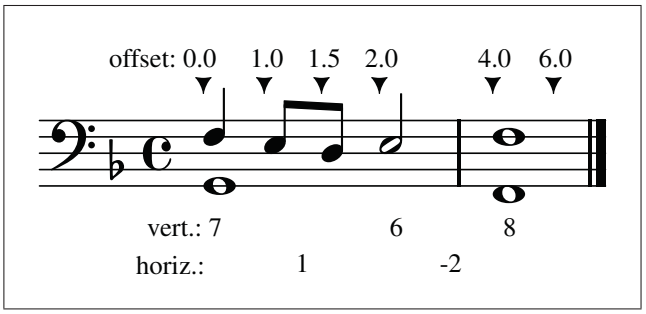


Figure 2. "Cadence" contrapuntal module from Figure 1, with `music21` offset values.

Figure 2 shows a representation of the "7 1 6 -2 8" interval 3-gram (a 3-gram because there are three vertical intervals). Using a half-note rhythmic offset, the first vertical interval is a seventh, the horizontal motion of the lower part is a unison (1), there is a vertical sixth, the lower part moves down by a second (-2), and the final vertical interval is an octave. In modal counterpoint, this is a highly conventionalized figure used to signal a cadence—a closing or concluding gesture for a phrase or piece. This is the same 3-gram as in the box in Figure 1.

Importantly, our analysis method requires that voicing information is encoded in our files. MIDI files where all parts are given in the same channel cannot be analyzed usefully with our software.

2.2 Previous Uses of n-Grams in MIR

We have chosen to map musical patterns with n-grams partly because of their previous use in natural language processing. [4] Some previous uses of n-grams in music analysis, and computerized counterpoint analysis, are described below.

J. Stephen Downie's dissertation presents a method for indexing melodic n-grams in a large set of folk melodies that will be searched using "Query by Humming" (QBH). [7] Downie's system is optimized for what he calls "lookup," rather than "analysis," and he admits that it lacks the detail required by musicologists. Importantly, Downie only indexes horizontal intervals: melody rather than counterpoint.

Another QBH lookup system, proposed by Shyamala Doraisamy, adapts n-grams for polyphonic music. [5, 6] While this system does account for polyphony, it does not record horizontal intervals so it lacks the detailed contrapuntal information we seek. Furthermore, Doraisamy's intervals are based on MIDI note numbers rather than the diatonic steps preferred by musicologists. Finally, the largest interval allowed by Doraisamy's tokenization strategy is 26

[2] Visit `counterpoint.elvisproject.ca`.

[3] Refer to `pandas.pydata.org` and `mit.edu/music21`.

[4] As in the *Google Ngram Viewer*; refer to `books.google.com/ngrams`.

semitones—just over two octaves, and therefore narrower than the normal distance between outer voices even in Renaissance polyphony. Considering the gradual expansion of range in polyphonic music, to the extremes of the late 19th-century orchestra, the gradual appearance of large intervals may be an important style indicator.

Meredith has proposed geometric transformation systems for encoding multi-dimensional musical information in a computer-friendly way. [9, 10] We especially appreciate the multi-dimensional emphasis and the mathematical properties of these systems, and the software's ability to work even when voicing information is not available in the symbolic file.

Finally, Jürgensen has studied accidentals in a large fifteenth-century manuscript of organ intabulations with an approach very similar to ours, but carried out using the Humdrum Toolkit. [8] She locates cadences by identifying a contrapuntal model and then records the use of accidentals at the cadence. While she searches only for specific contrapuntal modules, we identify all of the n-grams in a test set in order to determine the most frequently recurring contrapuntal patterns.

2.3 Multi-Dimensional n-Grams in VIS

Considering these previous uses of n-grams and counterpoint in MIR, we designed our software with the flexibility to accommodate our requirements, as well as those of future analysis strategies. By tokenizing n-grams with strings that minimally transform the input, musicologists can readily understand the information presented in an n-gram. This strategy offers a further benefit to programmers, who can easily create n-grams that include different musical dimensions without necessarily developing a new token transformation system. Users may choose to implement any tokenization strategy on top of our existing n-gram-indexing module.

The example 3-gram shown in Figure 2 is tokenized internally as "7 1," "6 -2," "8 END." Although there appear to be $2n - 1$ tokens, we consider a vertical interval and its following horizontal interval as a combined unit—as though it were a letter in an n-gram as used in computational linguistics. The simplicity afforded by using strings as tokens, each of which may contain an arbitrary array of musical information, has been advantageous.

Indeed, the difficulty of determining a musical analogue to the letter, word, and phrase divisions used in computational linguistics may be one of the reasons that computer-driven research has yet to gain much traction in mainstream musicology. That music lacks an equivalent for space characters poses an even greater problem in this regard: while some music does use clear breaks between phrases, their exact placement can often be disputed among experts. Musicologists also wish to account for the multiple simultaneous melody lines of polyphonic music, which has no equivalent in natural language. These are the primary motivating factors behind our multi-dimensional interval n-gram tokens that encode both vertical and horizontal intervals. As our research continues, context models and multiple viewpoint systems, in the style of Conklin and Witten, will partially obviate the questions of which n value to use, and of how best to incorporate varied musical elements. [1]

The popularity of Python within scientific computing communities allows us to benefit from any software that accepts *pandas* data objects. The easy-to-learn, object-oriented API of `music21`, along with the relatively high number of supported file formats, are also significant advantages. In the 1980s, a music analysis toolkit consisting of a collection of *awk* scripts was sensible, but Humdrum's limitation to UNIX systems and a single symbolic file format pose undesirable limitations for a big data project.

3. EXPERIMENT

3.1 Data Sets

We present an experiment to quantitatively describe style change in the Renaissance period, providing a partial answer for our primary research question. [5] We assembled test sets for three similar style periods, named after a representative composer from the period: Ockeghem (1440–85), Josquin (1485–1521), and Palestrina (1540–85). The pieces in the test set were chosen to represent the style period as accurately as our project's database allowed. [6] The twenty-year gap between the later periods is a result of less symbolic music being available from those decades. Each set consists of a mixture of sacred and secular vocal music, most with four parts, in a variety of genres, from a variety of composers. Though we analyzed n-grams between two and twenty-eight vertical intervals long, we report our results only for 3-grams because they are the shortest contrapuntal unit that holds meaning. Note that we include results from all possible two-part combinations, reflecting Renaissance contrapuntal thinking, where many-part textures are composed from a series of two-part structures. [3, 13]

The Ockeghem test set consists of 50 files: 28 in the MIDI format and 22 in **kern. For the composers, 8 pieces were written by Busnoys, 32 by Ockeghem, and 10 are late works by Dufay. The longest repeated n-gram was a 25-gram.

The Josquin test set consists of 56 files: 18 MIDI, 23 **kern, 9 MusicXML, and 6 NoteWorthy Composer. For the composers, 3 pieces were written by Agricola, 7 by Brumel, 6 by Compre, 2 by Fvin, 12 by Isaac, 19 by Josquin, 3 by Mouton, 2 by Obrecht, and 2 by la Rue. The longest repeated n-gram was a 28-gram.

Finally, the Palestrina test set consists of 53 files: 30 MIDI, 15 **kern, 6 MusicXML, and 2 NoteWorthy Composer. For the composers, 15 pieces were written by Palestrina, 9 by Rore, 28 by Victoria, and 1 by Wert. The longest repeated n-gram was a 26-gram.

3.2 Methodology

The *VIS Framework* uses a modular approach to query design, dividing analysis tasks into a series of well-defined

[5] You may download our test sets from `elvisproject.ca/ismir2014`.

[6] Visit `database.elvisproject.ca`.

steps. [7] We intend the module break-down to be helpful for musicologists who wish to reason about and design their own queries. Thus, musicological concerns drove the creation of many of the analysis steps, such as the filtering modules described below. The interval n-gram frequency experiment in this article uses the following modules: *NoteRestIndexer*, *IntervalIndexer*, *HorizontalIntervalIndexer*, *FilterByOffsetIndexer*, *FilterByRepeatIndexer*, *NGramIndexer*, *ColumnAggregator*, and finally the *FrequencyExperimenter*. [8]

The *NoteRestIndexer* finds note names and rests from a `music21` *Score*. The *IntervalIndexer* and *HorizontalIntervalIndexer* calculate vertical and horizontal intervals, respectively.

The *FilterByOffsetIndexer* uses a basic algorithm to filter weak-beat embellishing tones that otherwise obscure structural counterpoint. We regularize observations to a given rhythmic offset time interval using the `music21` offset, measured in quarter lengths. Refer to Figure 2 as an example, where vertical intervals are filtered with a 2.0 offset. Events beginning on a multiple of that duration will be retained (like the notes at 0.0, 2.0, and 4.0). Events lasting for multiples of that duration will appear to be repeated (like the note at 4.0, which is also recorded at 6.0). Events not beginning on a multiple of the duration will be removed (like the notes at 1.0 and 1.5) or shifted to the following offset, if no new event occurs. For this study, we chose a half-note (2.0) offset interval in accordance with Renaissance notation practices, but this can be changed in VIS at runtime.

The *FilterByRepeatIndexer* removes events that are identical to the immediately preceding event. Because of its placement in our workflow for this experiment, subsequent vertical intervals will not be counted if they use the same pitches. Our interval n-grams therefore necessarily involve contrapuntal *motion*, which is required for proper pattern recognition. Such repeated events arise in musical scores, for example, when singers recite many words on the same pitch. The *FilterByOffsetIndexer* may also create repeated events, as at offset 6.0 in Figure 2. Users may choose not to run this module.

In this article, our *NGramIndexer* includes results from all pairs of part combination. Users may exclude some combinations at runtime, choosing to limit their query to the highest and lowest parts, for example. On receiving intervals from the *FilterByRepeatIndexer*, the *NGramIndexer* uses the gliding window technique to capture all possible overlapping interval n-grams. The indexer also accepts a list of tokens that prevent an n-gram from being counted. We use this feature to avoid counting contrapuntal patterns that include rests. Finally, the *NGramIndexer* may add grouping characters, surrounding "vertical" events in brackets and "horizontal" events in parentheses to enhance legibility of long n-grams. The 3-grams in this article are short enough that grouping characters are unnecessary; on the other hand, the legibility of the "[10] (+2) [9] (1) [8] (+2) [7] (1) [6] (-2) [8]" 6-gram found 27 times in the Palestrina test set greatly benefits from grouping characters.

The *FrequencyExperimenter* counts the number of occurrences of each n-gram. These results, still specific to part combinations within pieces, are then combined with the *ColumnAggregator*.

On receiving a spreadsheet of results from VIS, we calculated the number of n-grams as the percentage total of all n-grams in each of the test sets. For each set, we also counted the total number of 3-grams observed (including all repetitions of all 3-grams), the number of distinct 3-gram types (whether repeated or not), and the number of 3-gram types that occur more than once; these are shown below in Table 1.

3.3 Results

Due to the limited time span represented in this study, we wish to suggest avenues for future exploration, rather than offer conclusive findings. We present a visualization of the experimental results in Figure 3, a hybrid between a Venn diagram, word cloud (i.e., a 3-gram cloud), and a timeline. The diagram includes interval 3-grams that constitute greater than 0.2% of the 3-grams in at least one of the test sets. When a 3-gram appears in an intersection of style periods, that 3-gram constitutes greater than 0.2% of the 3-grams in those sets. As in a world cloud, the font size is scaled proportionately to a 3-gram's frequency in the test sets in which it is common. Most visually striking is the confirmation of musicologists' existing experiential knowledge: certain contrapuntal patterns are common to all three style periods, including the cadence module ("7 1 6 -2 8") and two other 3-grams that end with the "7 1 6" cadential suspension. These results make sense because cadences are an essential feature of musical syntax.

Test Set	Total	Types	Repeated Types
Ockeghem	30,640	10,644	4,509 (42%)
Josquin	31,233	9,268	4,323 (47%)
Palestrina	33,339	10,773	5,023 (47%)

Table 1. Summary of 3-gram repetitions in our query.

In addition to the common cadential patterns noted above, both Figure 3 and Table 1 show evidence of stylistic change over time. Most notably, the Josquin and Palestrina test sets show a higher level of repetition than the Ockeghem set. The number of 3-grams included in Figure 3 is higher in the Josquin test set (with seventeen 3-grams) than either the Ockeghem or Palestrina sets (both with eleven 3-grams). Yet Table 1 indicates the Josquin and Palestrina sets both have a higher percentage of 3-gram types that are repeated at least once (47% in both sets, compared to 42% in the Ockeghem set). These data suggest an increase in repetition of contrapuntal modules from the Ockeghem to the Josquin generations, which was retained in the Palestrina generation. Figure 3 only partially reinforces this suggestion: while five 3-grams are unique to the Ockeghem set, six are unique to the Josquin set, but only one is unique

[7] This section refers to the 2.x release series.

[8] For more information about the VIS Framework's analysis modules and overall architecture, please refer to our Python API at `vis.elvisproject.ca`.

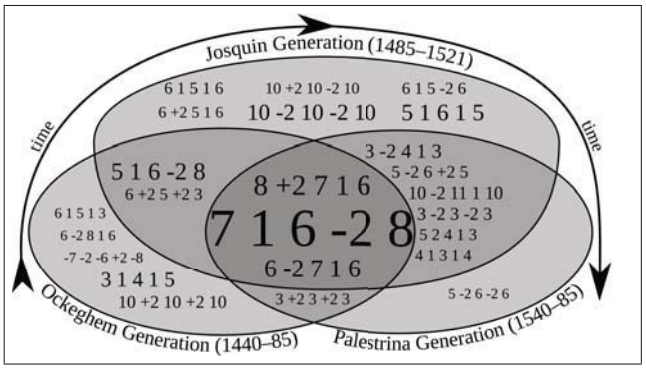


Figure 3. Frequency of contrapuntal modules is different between temporally-adjacent style periods.

to the Palestrina set. Moreover, the "5 -2 6 -2 6" module, unique to the Palestrina set, is the least common 3-gram in Figure 3—how did contrapuntal repetition both decrease in Palestrina's generation *and* remain the same?

Previous research by Cumming and Schubert may help us explain the data. In 2008, Cumming noted that exact repetition became much more common in Josquin's lifetime than in Ockeghem's. [2] Schubert showed that composers tended to repeat contrapuntal patterns in inversion during Palestrina's lifetime, so that the lower voice is moved above the original upper voice. [13] Inversion changes a contrapuntal pattern's vertical intervals in a consistent way that preserves, but switches, the horizontal intervals of the two parts. For example, "7 1 6 -2 8" inverts at the octave to "2 -2 3 +2 1." While humans can recognize both forms as two versions of the same pattern, VIS currently shows only exact repetition; future enhancements will permit us to equate the original and the inversion. This decision may explain why our data show lower rates of repetition for the Palestrina test set.

We find further evidence of stylistic change in Figure 3: certain patterns that musicologists consider to be common across all Renaissance music are in fact not equally common in our three test sets. For example, motion by parallel thirds and tenths appears to be more common in certain style periods than others, and in a way that does not yet make sense. The Palestrina set shares ascending parallel thirds ("3 +2 3 +2 3") with the Ockeghem and descending parallel thirds ("3 -2 3 -2 3") with the Josquin set. Ascending parallel tenths ("10 +2 10 +2 10") are more common in the Ockeghem set, and descending parallel tenths ("10 -2 10 -2 10") in the Josquin set. In particular, descending parallel thirds are an order of magnitude less common in the Ockeghem test set than the Josquin or Palestrina (constituting 0.013%, 0.272%, and 0.225% of 3-grams in their test set, respectively). Conventional musicological wisdom suggests these 3-grams will be equally common in all three test sets, and that parallel tenths will be more common than parallel thirds in later style periods, as the range between voices expands. Since the reasons for such a deviation are not yet known, we require further investigation to study the changing nature of contrapuntal repetition during the Renaissance period. Yet even with these preliminary findings it is clear that evidence-based research has much to offer musicology.

4. FUTURE WORK

Our research will continue by extending VIS to add the option of equivalence classes that can group, for example, inversionally-related interval n-grams. We will also build on previous work with melody- and harmony-focussed multiple viewpoint systems to create an all-voice contrapuntal prediction model. [1, 14]

Our experiments will continue with larger test sets for increased confidence in our findings, also adding style periods earlier than the Ockeghem and later than the Palestrina sets, and subdividing our current style periods. This will help us reassess boundaries between style periods, and exactly what such a boundary entails. We will also compare results of single pieces with test sets of various sizes.

Finally, we will implement additional multi-dimensional n-gram tokens, for example by adding the note name of the lowest voice. This approach would encode Figure 2 as "7 F 1 6 F -2 8 E." In Renaissance music, this type of n-gram will clarify the relationships between contrapuntal modules and a piece's mode.

5. CONCLUSION

The *VIS Framework for Music Analysis* is a musicologist-friendly Python library designed to analyze large amounts of symbolic musical data. Thus far, our work has concentrated on counterpoint—successions of vertical intervals and the horizontal intervals connecting them—which some scholars view as composers' primary concern throughout the development of Western music. Our software uses multi-dimensional n-grams to find and count the frequency of repeated contrapuntal patterns, or modules. In particular, by retaining all inputted dimensions and using strings as tokens (rather than integers or characters), we simultaneously allow musicologists to quickly understand the content of an n-gram while also avoiding the challenge of developing a new tokenization strategy for every musical dimension added to the n-gram. We hope this flexibility and ease-of-use encourages musicologists and non-expert programmers, who would otherwise be discouraged from computer-based music analysis, to experiment more freely.

The results of our query presented in this article, which compares the most commonly-repeated contrapuntal modules in three Renaissance style periods, show the type of insight possible from computerized music research. The time-consuming effort required for previous work on contrapuntal modules is greatly reduced when analysts have access to specialized computer software. We analyzed more than 150 polyphonic compositions for interval n-grams between two and twenty-eight vertical intervals in length, which would have taken months or years for a human. Even with simple mathematical strategies like counting the frequency of interval n-grams to know which are most common, we can confirm existing intuitive knowledge about

the foundations of counterpoint while also suggesting avenues for future research on the nature of musical repetition.

6. REFERENCES

[1] D. Conklin and I. Witten. Multiple viewpoint systems for music prediction. *Journal of New Music Research*, 24(1):51–73, 1995.

[2] J. Cumming. From variety to repetition: The birth of imitative polyphony. In Bruno Bouckaert, Eugeen Schreurs, and Ivan Asselman, editors, *Yearbook of the Alamire Foundation*, number 6, pages 21–44. Alamire, 2008.

[3] J. Cumming. From two-part framework to movable module. In Judith Peraino, editor, *Medieval music in practice: Studies in honor of Richard Crocker*, pages 177–215. American Institute of Musicology, 2013.

[4] M. S. Cuthbert and C. Ariza. music21: A toolkit for computer-aided musicology and symbolic music data. In *Proceedings of the International Symposium on Music Information Retrieval*, pages 637–42, 2010.

[5] S. Doraisamy. *Polyphonic Music Retrieval: The n-gram approach*. PhD thesis, University of London, 2004.

[6] S. Doraisamy and S. Rger. Robust polyphonic music retrieval with n-grams. *Journal of Intelligent Information Systems*, 21(1):53–70, 2003.

[7] J. S. Downie. *Evaluating a Simple Approach to Music Information Retrieval: Conceiving melodic n-grams as text*. PhD thesis, University of Western Ontario, 1999.

[8] F. Jürgensen. Cadential accidentals in the Buxheim organ book and its concordances: A midfifteenth-century context for musica ficta practice. *Acta Musicologica*, 83(1):39–68, 2011.

[9] D. Meredith. A geometric language for representing structure in polyphonic music. In *Proceedings of the International Society for Music Information Retrieval*, pages 133–8, 2012.

[10] D. Meredith, K. Lemström, and G. Wiggins. Algorithms for discovering repeated patterns in multidimensional representations of polyphonic music. *Journal of New Music Research*, 41(4):321–45.

[11] J. A. Owens. *Composers at Work: The Craft of Musical Composition 1450–1600*. Oxford University Press, 1997.

[12] P. Schubert. Counterpoint pedagogy in the renaissance. In T. Christensen, editor, *The Cambridge History of Western Music Theory*, pages 503–33. Cambridge University Press, 2002.

[13] P. Schubert. Hidden forms in Palestrina's first book of four-voice motets. *Journal of the American Musicological Society*, 60(3):483–556, 2007.

[14] R. Whorley, G. Wiggins, C. Rhodes, and M. Pearce. Multiple viewpoint systems: Time complexity and the construction of domains for complex musical viewpoints in the harmonisation problem. *Journal of New Music Research*, 42(3):237–66, 2013.

HIERARCHICAL APPROACH TO DETECT COMMON MISTAKES OF BEGINNER FLUTE PLAYERS

Yoonchang Han, Kyogu Lee
Music and Audio Research Group
Seoul National University, Seoul, Republic of Korea
{yoonchanghan,kglee}@snu.ac.kr

ABSTRACT

Music lessons are a repetitive process of giving feedback on a student's performance techniques. The manner in which performance skills are improved depends on the particular instrument, and therefore, it is important to consider the unique characteristics of the target instrument. In this paper, we investigate the common mistakes of beginner flute players and propose a hierarchical approach to detect such mistakes. We first examine the structure and mechanism of the flute, and define several types of common mistakes that can be caused by incorrect assembly, poor blowing skills, or mis-fingering. We propose tailored algorithms for detecting each case by combining deterministic signal processing and deep learning, to quantify the quality of a flute sound. The system is structured hierarchically, as mis-fingering detection requires the input sound to be correctly assembled and blown to discriminate minor sound difference. Experimental results show that it is possible to identify different mistakes in flute performance using our proposed algorithms.

1. INTRODUCTION

The most important part of a music lesson is giving a student feedback on his or her performance, posture, and playing skills so that the student can play the sound correctly. Music lesson methods vary depending on the instrument being learned; therefore, audio signal processing for music education should make extensive use of prior knowledge regarding playing style, common mistakes, unique characteristics, and constraints of the target instrument. However, most existing music signal analysis techniques use a general-purpose model, and relatively little attention is paid to an instrument-specific approach. A general-purpose model is advantageous because it can be applied to various types of instruments. However, this model lacks the capability to capture instrument-specific sound characteristics. There are always common mistakes that beginners make, but little is known about how to detect these automatically.

The goal of this paper is to investigate common beginner's mistakes when playing a specific instrument—the flute, in this case—and to analyze the spectral characteristic of each case to give the student appropriate feedback on his or her performance. Because the sound of a musical instrument is affected by numerous factors, in our work, we first divide the factors that usually lead beginners to play the wrong sound into three parts: incorrect flute assembly, blowing skill, and fingering.

The rest of the paper is organized as follows: We briefly present existing works related to our proposed idea. Then, we investigate possible mistakes in flute performance by examining the structure and mechanism of the flute, and several types of common mistakes and the resulting sounds are explained. Next, we present an overall system structure to distinguish each mistake, along with a detail explanation of each proposed algorithm. We then present the experimental results to demonstrate the feasibility of the proposed system, followed by our conclusion and directions for future work.

2. RELATED WORK

The characteristics of musical instruments depend on their sound production mechanism. The characteristics of one instrument can greatly differ from those of others, and each instrument's characteristics may not be captured equally well as another even when using the same computational model [2]. However, there has been minimal research regarding an instrument-specific model. Some examples of instrument-specific approaches involve the use of a violin [8, 14-16], guitar [1], bells [9], and tabla [5]. For instance, the violin transcription system in [8] makes use of characteristics such as highest and lowest pitch, possible play style (e.g., upper octave duophony), vibrato, and loudness. The training system in [14] uses a common envelope style of violin sound for note segmentation prior to real-time pitch detection, and [9] uses the acoustic characteristics of a church bell, as well as the rules of a bell charming performance, for transcription and estimating the number of bells. In addition, a chord transcription system designed for guitar in [1] outperforms the non-guitar-specific method.

As shown above, using prior knowledge of the characteristics of a target instrument creates new possibilities in music signal processing, and can also improve the per-

formance of the system. However, there are still many instruments to be studied, and the flute is one of them.

3. COMMON MISTAKES OF A FLUTE PLAYER

3.1 Assembling the Flute

Like most woodwind instruments, the flute needs to be assembled before it is played. The flute consists of a head joint, body joint, and foot joint, as shown in Figure 1. The connecting part between the body and foot joint is very short, while the connecting part between the head joint and body joint is a few centimeters long. This intentionally designed adjustable part is called the tuning slide, and it can be used for changing the total length of the flute to various sizes, which affects the overall pitch of the flute. For instance, if the head joint is placed very deep into the tuning slide of the body, the pitch will be increased for every note. By contrast, if the head joint is pulled out too far, the overall pitch will drop owing to the longer wavelength.

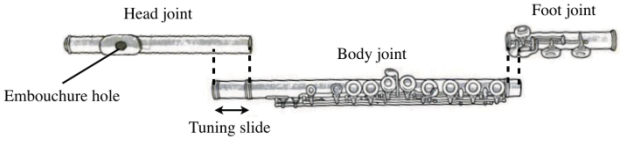


Figure 1. Flute consists of head joint, body joint, and foot joint (modified after [11]).

Another method of pitch tuning is adjusting the cork part of the head joint, as shown in Figure 2. This can be adjusted by a screw. Pushing the cork will raise the pitch of all notes. However, this is beyond the scope of this paper, as this screw is normally not adjusted by flute performers but by flute technicians.

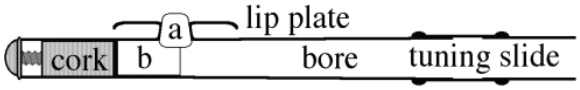


Figure 2. Schematic of a flute head joint [13].

Trained performers use this variable tuning slide for pitch tuning. The pitch of the flute is sensitive to the conditions of the surrounding environment, such as humidity and temperature. However, novice flutists are not sensitive to minor pitch shifting, and they may play the flute in the wrong overall pitch without recognizing it.

3.2 Blowing Embouchure

The flute generates sound by blowing a rapid air jet across the embouchure[1] hole, as shown in Figure 3. Hence, the quality of the generated sound is highly dependent on the blowing skill of the performer. Blowing skill involves lip position and the thickness/stability of the air jet. Clear tone production is challenging for beginners because the method of tone production for the flute is not supported by mechanical parts; rather, it depends only on the player's blowing skill [4].

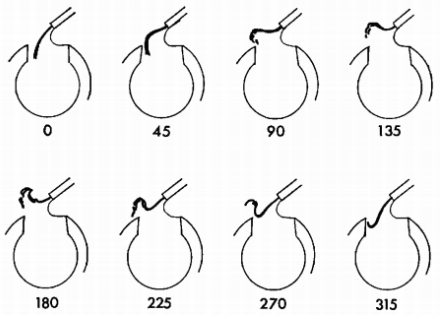


Figure 3. Airstream oscillation of the flute embouchure hole. The labels indicate the phase angles of the acoustic current at the hole [3].

Tone quality and octave of the sound are related to blowing skill. The flute has a range of three octaves, starting from middle C (C4), with several less-used notes in octaves 3 and 7. The blowing pressure determines the octave of the sound, as shown in Figure 4. Greater blowing pressure can be achieved by blowing a narrower and stronger air jet. To generate a stable and clean sound, it is important to keep this blowing pressure reasonably steady. Failure to do this will result in fluctuating sound and noise, which is highly unpleasant and typically the first hurdle for beginners to overcome in their training.

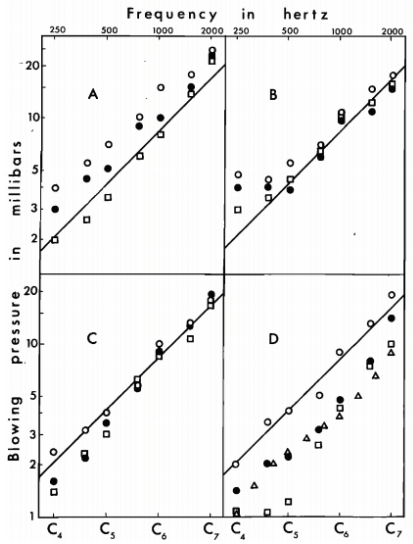


Figure 4. Air jet blowing pressure has a roughly linear relationship to fundamental frequency. A, B, C, and D are different performers, and different shapes represent different dynamics [12].

3.3 Fingering

Novice flutists frequently make mistakes in fingering owing to their lack of familiarity with the irregular fingering rules of the flute. High-octave fingering is comparatively more complex than low-octave fingering [4], which is the reason why flute lessons usually start with the lowest octave and move step-by-step to higher octaves. Hence, we

[1] Mouthpiece of a musical instrument.

focus on octaves 4 and 5, which are the octaves that beginner flute players initially study.

Most of the octave 5 fingerings are identical to those of octave 4, as shown in Figure 3. However, the fingering for C and D, as well as the sharps of these notes, require different fingerings than those of octave 4. These notes can be played with octave 4 fingering using a faster and sharper air jet, but this results in a slightly airy timbre, compared to the sound when the flute is correctly fingered. As this airy timbre is not significantly noticeable, and most of the notes in octaves 4 and 5 share the same fingering, many beginners do not notice that they used octave 4 fingerings to play octave 5, unless the instructor spots it.

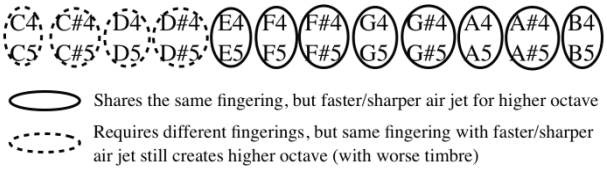


Figure 5. Fingering of octave 4 and 5 flute notes. Note that C, D, and sharps of these require different fingering, unlike E, F, G, A, and B.

Another fingering-related problem is the proper positioning of the fingers. The open-hole flute requires that the flutist use his or her fingers to block the holes in the keys. Most professional flutists prefer the open-hole flute owing to its advantages in tone production and intonation adjustment [4]. However, this is not considered in our system because beginners who have trouble with blocking open-hole keys can avoid this problem by putting plastic plugs in the holes until they get used to playing the open-hole flute.

4. PROPOSED SYSTEM & METRICS

The overall system comprises several steps. In the first step, the system determines whether the flute is assembled correctly using entire input audio. Next, once the flute sound is detected as coming from a correctly assembled flute, the system measures if sound of the each note is a clear, correctly blown sound or an airy-timbered sound. Finally, the properly blown sound is identified as sound generated from either correct fingering or incorrect fingering. The system is hierarchically structured, because mis-fingering detection does not work well for fluctuated sound or head joint pushed/pulled sound as it requires discriminating minor sound difference. The input audio is resampled to 16 kHz first, and he system architecture is shown in Figure 6.

4.1 Assembling Error Detection

Some mistakes can cause modifications to the overall pitch, and some mistakes result in poor timbre. The assembling error affects only the overall pitch of the generated sound. As mentioned in 3.1, the distance the head joint is pushed in or pulled out from the tuning slide of the body joint determines the overall pitch. To this end, a quantized chromagram from Harte and Sandler is used to detect the tuning center [6].

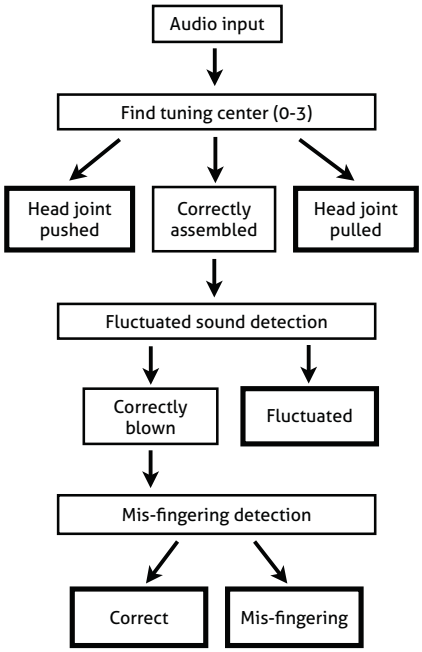


Figure 6. Flow diagram of the overall system. The bold box indicates where the system sends feedback to the user.

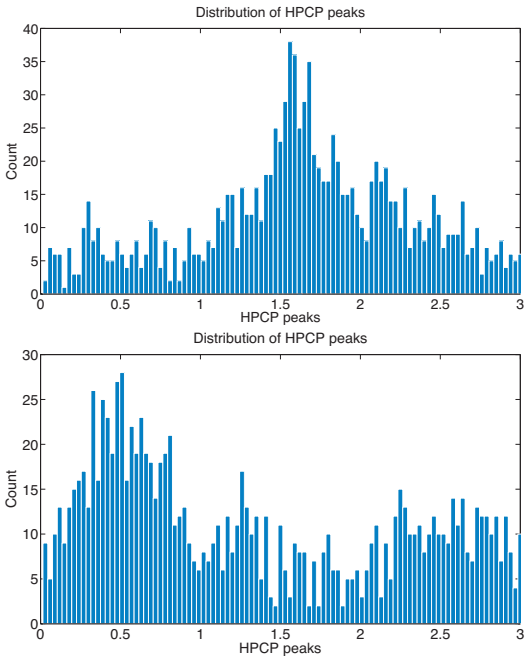


Figure 7. HPCP peaks histogram within a semitone for correctly (up) and loosely assembled (down) case.

To determine the tuning center, a spectrum of linear frequency spectra is Constant-Q transformed and summed across octaves to produce a harmonic pitch class profile (HPCP). A 36-bin quantized chromagram is used to determine the semitone center, and three bins were al-

located for each semitone. By observing the distribution of peak positions across the width of a semitone, as shown in Figure 7, it is possible to determine the tuning center of the instrument. Because three bins are allocated for each semitone, the tuning center of a perfectly tuned sound would ideally be 1.5. Therefore, the system will consider the input sound to be correctly tuned when the tuning center value is approximately 1.5. If the detected tuning center is too low (less than 1), the system sends feedback to the user that the head joint is too loosely assembled. Conversely, the system tells the user that the head joint is assembled more tightly than necessary when the tuning center is high (greater than 1).

4.2 Fluctuated Sound Detection

Incorrect lip position on the embouchure, along with an irregular stream of blown wind, results in a highly unpleasant and fluctuating tone. This sound contains many inharmonic partials in a spectrum, and it is clearly visible on a spectrogram. Performing binary masking on a spectrogram makes these inharmonic partials more obvious, as shown in the second row of Figure 8.

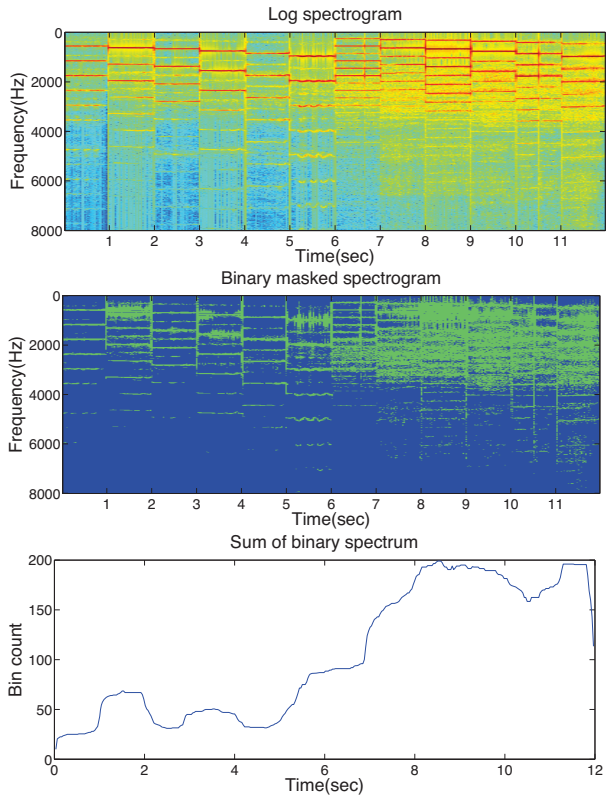


Figure 8. Log spectrogram, binary masked spectrogram, and sum of bins for each frame for D, E, F, G, A, and B of octave 5. Up to 6 second is correctly blown sound and from 6 to 12 second is fluctuated sound.

Binary masking is performed as follows:

$$X_b(k) = \begin{cases} 0 & X(k) < \theta \\ 1 & X(k) > \theta \end{cases} \quad (1)$$

where X is the log spectrum, X_b is the binary masked spectrum, and θ is the threshold constant. Empirically, a value between -20 and -30 works well for θ, depends on recording environment. Note that these values are obtained when natural log multiplied by 20 is used for the log spectrum. Using this binary masked spectrogram, the sum of the number of positive valued bins of each spectrum can be used as a measurement for determining how the sound fluctuates owing to poor blowing skill. This can be expressed as follows:

$$F(k) = \sum_{k=0}^{N-1} X_b(k) \quad (2)$$

where F is the amount of fluctuation. The third row of Figure 8 is F value obtained from (2) with 1 second median filtering, and it is possible to observe the value is much higher for fluctuated sound than correctly blown sound.

4.3 Mis-fingering detection

As mentioned in 3.3, for C5, C#5, D5, and D#5, using octave 4 fingering with a faster and sharper air jet still generates octave 5 pitches even without correct fingering, although the timbre is slightly airy. To detect this timbral difference, we decided to use both the Mel-frequency cepstral coefficient (MFCC)—a widely used, hand-designed feature—and sparse filtering (SF) [10]—a deep-layered, unsupervised feature learning method. SF works by optimizing the sparsity of feature distribution, and it works well on a range of data modalities without specific tuning. Both single- and double-layered sparse filtering were used with 200 units for each layer. The obtained feature was classified into two classes (correct/incorrect) using a random forest (RF) classifier, which exhibits better performance than a support vector machine or back-propagation neural network in a variety of cases [7]. The flow diagram for mis-fingering detection is shown below.

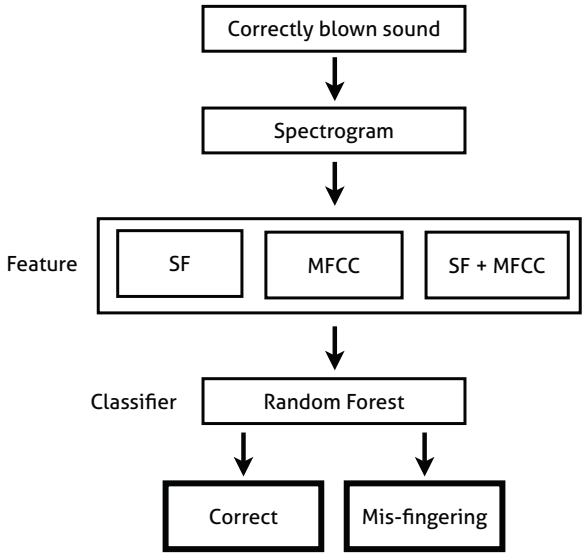


Figure 9. Flow diagram for mis-fingering detection.

5. EXPERIMENT

5.1 Objective & Procedure

The goal of our experiment was to explore whether the proposed system and algorithms work well for detecting the mistakes of beginner flutists. Flute sound samples were obtained from two intermediates (who have played the flute for one to two years) and one expert (who holds an exam score of Grade 8 with a Distinction). Flutes used for the experiment were a B foot joint with open holes, and a silver head with nickel body and foot. The correct flute sound, fluctuating sound, head joint pulled, and head joint pushed sound were recorded for octaves between 4 and 5. The length of the collected audio was 30 seconds for each semitone. The case of correct and incorrect fingering for C5, C#5, D5, and D#5 was recorded for 10 minutes each to obtain sufficient training data. The input audio was recorded at 44.1 kHz mono and downsampled to 18 kHz. Tuning center was calculated from whole target audio as it is not time-varying characteristics. Meanwhile, fluctuating and mis-fingering detection was performed framewise. Different window and hop size were used for each experiment, as each mistake detection algorithm requires different spectral resolution.

5.2 Results

The experimental results show that the system successfully distinguishes each mistake. To find tuning center, a 74 ms window and 18 ms hop size were used. As shown in Table 1, the tuning center of a correctly played sample is close to 1.5, which is the exact center. Also, tuning center values for the head joint when it is pushed and pulled fell into the expected range, which were (0–1) and (2–3), respectively.

Mistake cases	Tuning center value (0 to 3)		
	Player 1	Player 2	Player 3
Correct	1.68	1.59	1.62
Head joint pushed	2.55	2.49	2.43
Head joint pulled	0.48	0.45	0.75

Table 1. Tuning center values of correct, head joint pushed, and head joint pulled flute sound of three different flutists.

Next, Figure 10 is a framewise distribution of fluctuation measure (1) for correct and fluctuated flute sound. A 64 ms window and 32 ms hop size were used, with θ value of -25dB. The median value of the correct flute sound is 50, and most of the values fall between 63 and 37. The fluctuating flute sound has a median value of 167, and most of the values fall between 150 and 178. This means that these cases are clearly distinguishable using the proposed metric.

Finally, Table 2 shows the ten-fold cross-validation results of the proposed mis-fingering classification using single-layer SF, double-layer SF, and MFCC as a feature, and RF as a classifier. A 16 ms window and 10 ms hop size were used, and SF was used with 200 units per layer.

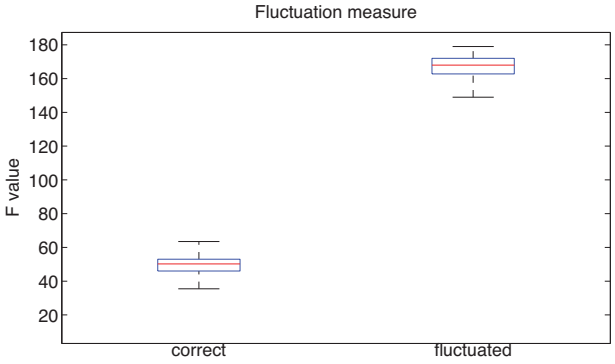


Figure 10. Box plot of fluctuation measurements. The central marks indicate the median, and the edges are the first and third quartiles.

Method	Accuracy (%)
Spectrogram + SF (single)	90.24
Spectrogram + SF (double)	90.02
MFCC	90.89
MFCC + SF (single)	90.33
MFCC + SF (double)	91.35

Table 2. Mis-fingering classification ten-fold cross-validation result using SF/ MFCC as a feature and RF as a classifier.

The result shows that the combination of the MFCC and double-layered SF performs the best; however, all of the approaches perform reasonably well within a not very meaningful margin. The result indicates that the MFCC, a handcrafted feature, is still useful in separating the timbral differences of the flute. Further, although SF is not designed for the purpose of timbre analysis, it works quite well without fine-tuning, as mentioned in [10]. In the experiment, single-layered SF worked better when the input is a spectrogram, but double-layered SF showed better performance when the input is MFCC.

6. CONCLUSION & FUTURE WORK

The objective of our work is to use audio signal analysis to give a student feedback on his or her flute performance to help fix mistakes, as a lesson teacher would do. To achieve this goal, we examined the mechanism and structure of the flute. We also investigated the common mistakes of beginner flute players. We determined several types of common mistakes and developed a hierarchical system to detect such cases by observing the tuning cen-

ter, fluctuation metric, and a mis-fingering detection algorithm. As a result, we have successfully identified common mistake cases from input audio, which can be used as feedback that would be provided by a lesson teacher-. Head-joint assembling errors were detected by determining the tuning center of the flute sound. Fluctuating sound caused by poor blowing skills was separated from the correct flute sound by measuring the amount of noisy harmonic contents. Finally, mis-fingering cases were detected by analyzing their timbre using MFCC and SF with an RF classifier.

There remain some problems to be tackled in this mistake detection algorithm for real-world user applications. First, the mis-fingering detection algorithm may be affected by the material or maker of the flute because the algorithm detects very minor changes in timbre. In the experiment, only two types of flute (silver head with nickel body, and foot) were used. However, the flute can be made of various types of metal, such as silver, gold, and platinum. Moreover, various flute makers have their own timbral characteristics, which may influence the classification results. Second, the experiment was done on the frame level, but the user perceives the score based on the note level. Hence, the system should be used along with appropriate onset-offset detection to give more user-friendly feedback.

We believe that this type of timbre-related and user-behavior-oriented feedback is highly important for the next-generation music transcription systems, especially those used for educational purposes. Playing the instrument with correct onset and pitch is not a very difficult part of being a good player, but making a beautiful timbre is what really takes time. This paper focuses only on the flute; however, our overall approach, including analyzing mistake cases and determining customized solutions, can be applied to various instruments in a similar way.

7. ACKNOWLEDGEMENTS

This research was supported by the MSIP (Ministry of Science, ICT & Future Planning), Korea, under the ITRC (Information Technology Research Center) support program supervised by the NIPA (National IT Industry Promotion Agency" (NIPA-2013-H0301-13-4005).

8. REFERENCES

[1] A. M. Barbancho, A. Klapuri, L. J. Tardon, and I. Barbancho: "Automatic Transcription of Guitar Chords and Fingering from Audio," IEEE TASLP, 20(3): 915–921, 2012.

[2] E. Benetos, S. Dixon, D. Giannoulis, H. Kirchoff, and A. Klapuri: "Automatic Music Transcription: Breaking the Glass Ceiling," In *ISMIR*, 2012.

[3] J. W. Coltman: "Sounding Mechanism of the Flute and Organ Pipe." *The Journal of the Acoustical Society of America* 44, 1968.

[4] C. Delaney, *Teacher's Guide for the Flute*. Rev. 11/98, Selmer, 1969.

[5] O. Gillet and G. Richard: "Automatic Labelling of Tabla Signals," In *ISMIR*, 2003.

[6] C. Harte and M. Sandler: "Automatic Chord Identification using a Quantised Chromagram." In *Audio Engineering Society Convention 118*. 2005.

[7] M. Liu, M. Wang, J. Wang, and D. Li: "Comparison of Random Forest, Support Vector Machine and Back Propagation Neural Network for Electronic Tongue Data Classification: Application to the Recognition of Orange Beverage and Chinese Vinegar," Elsevier Sensors and Actuators, pp. 970–980, Vol. 177, 2013.

[8] A. Loscos, Y. Wang, and W. J. Boo: "Low Level Descriptors for Automatic Violin Transcription." In *ISMIR*, 2006.

[9] M. Marolt: "Automatic Transcription of Bell Chiming Recordings." In *IEEE TASLP*, 20(3): pp. 844–853, 2012.

[10] J. Ngiam, P. W. Koh, Z. Chen, S. Bhaskar and A. Y. Ng., "Sparse Filtering," In NIPS, 2011.

[11] H. Pinksterboer, *Tipbook Flute and Piccolo: The Complete Guide*, Hal Leonard, 2009.

[12] T. D. Rossing, F. Richard Moore, and P. A. Wheeler: *The Science of Sound*. Vol. 2. Massachusetts. Addison-Wesley, 1990.

[13] J. Smith, J. Wolfe, and M. Green: "Head Joint, Embouchure Hole and Filtering Effects on the Input Impedance of Flutes." In *Proc. of the Stockholm Music Acoustics Conference*, pp. 295–298. 2003.

[14] J. Wang, S. Wang, W. Chen, K. Chang, and H. Chen: "Real-Time Pitch Training System for Violin Learners," *Multimedia and Expo Workshops (ICMEW), IEEE*, 2012.

[15] R. S. Wilson: "First Steps Towards Violin Performance Extraction using Genetic Programming," In John R. Koza, editor, *Genetic Algorithms and Genetic programming*, pp. 253–262, 2002.

[16] J. Yin, Y. Wang, and D. Hsu: "Digital Violin Tutor: An Integrated System for Beginning Violin Learners," ACM Multimedia, Hilton, Singapore, 2005.

ROBUST JOINT ALIGNMENT OF MULTIPLE VERSIONS OF A PIECE OF MUSIC

Siying Wang **Sebastian Ewert** **Simon Dixon**
Queen Mary University of London, UK
{siying.wang,s.ewert,s.e.dixon}@qmul.ac.uk

ABSTRACT

Large music content libraries often comprise multiple versions of a piece of music. To establish a link between different versions, automatic music alignment methods map each position in one version to a corresponding position in another version. Due to the leeway in interpreting a piece, any two versions can differ significantly, for example, in terms of local tempo, articulation, or playing style. For a given pair of versions, these differences can be significant such that even state-of-the-art methods fail to identify a correct alignment. In this paper, we present a novel method that increases the robustness for difficult to align cases. Instead of aligning only pairs of versions as done in previous methods, our method aligns multiple versions in a joint manner. This way, the alignment can be computed by comparing each version not only with one but with several versions, which stabilizes the comparison and leads to an increase in alignment robustness. Using recordings from the Mazurka Project, the alignment error for our proposed method was 14% lower on average compared to a state-of-the-art method, with significantly less outliers (standard deviation 53% lower).

1. INTRODUCTION

Recent years have seen significant efforts to create large, comprehensive music collections. Music content providers (e.g. Spotify, iTunes, Pandora) rely on their existence, while national libraries and charitable organizations create and curate them in order to provide access to cultural heritage. For a given piece of music, large collections often contain various related recordings (cover songs, different interpretations), videos (official clip, live concert) and musical scores (in different formats such as MIDI and MusicXML, covering several editions). To identify and link these different versions, various automatic alignment methods have been proposed in recent years. Such *synchronization methods* have been used to facilitate navigation in large collections [1], to implement score following in real-time [2–5], to compare different interpretations of a piece [6], to identify cover songs [7] or to simplify complex audio processing tasks [8].

In general, the goal of music synchronization is, given a position in one version of a piece of music, to locate the corresponding position in another version. To compute a synchronization, existing methods align two versions of a piece at a time, even if several relevant versions are available. For example, in [9, 10] a score of a piece is automatically aligned to a corresponding audio recording, while in [11] two acoustic realizations are being synchronized. As shown previously, current methods yield in many cases alignments of high accuracy [9–11]. However, musicians can interpret a piece in diverse ways, which can lead to significant local differences in terms of articulation and note lengths, ornamental notes, or the relative loudness of notes (balance). If such differences are substantial, the alignment accuracy of state-of-the-art methods can drop significantly.

To increase alignment robustness for difficult cases, the main idea in this paper is to exploit the fact that multiple versions of a piece are often available and can be aligned in a joint way. This way, we can exploit the additional information that each version provides about how a certain position in a piece can be realized by a musician. As a consequence, while two given recordings might be rather different and hard to align, both of them might actually be more similar to a third recording and including such a recording within the alignment process can lead to an increase in overall robustness. To compute our joint synchronization, we modify a multiple sequence alignment method typically employed in biological signal processing and combine it with strategies developed in a musical context based on Multiscale-DTW (FastDTW) and chroma-based onset features for increased computational efficiency and synchronization accuracy. In the following, we describe technical details of this method in Section 2. Then, we report on some of our experiments in Section 3. Conclusions and prospects for future work are given in Section 4.

2. ALIGNMENT METHOD

Various methods have been proposed to align two given data sequences, including Dynamic Time Warping (DTW) and Hidden Markov Models (HMM) [2], Conditional Random Fields (CRF) [9], and Particle Filter / Monte-Carlo Sampling (MCS) based methods [4, 5]. With the exception of MCS methods, which are online methods, the remaining three methods operate in an offline fashion and are quite

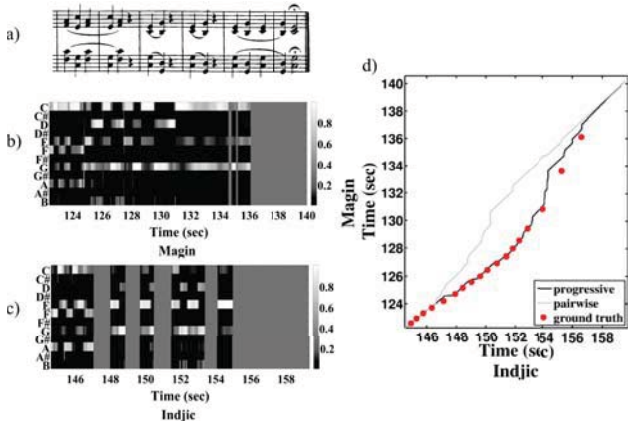


Figure 1. Alignment of two interpretations of Chopin Op. 24 No. 2, measures 115-120: **(a)** Score for the six measures. **(b)/(c)** Chroma features for an interpretation by Magin and Indjic, respectively; chroma features with uniform energy distribution are the result of silence in the recording. **(d)** Alignment results for our baseline pairwise (gray) and proposed method (black).

similar from an algorithmic point of view. We describe our proposed method as an extension to DTW. However, the underlying ideas are applicable in HMM and CRF contexts as well.

2.1 Baseline Pairwise Alignment

To summarize DTW-based alignment, let $X := (x_1, x_2, \ldots, x_N)$ and $Y := (y_1, y_2, \ldots, y_M)$ be two feature sequences with $x_n, y_m \in \mathcal{F}$, where $\mathcal{F}$ denotes a suitable feature space. Furthermore, let $c : \mathcal{F} \times \mathcal{F} \to \mathbb{R}$ denote a local cost measure on $\mathcal{F}$. We define a resulting $(N \times M)$ cost matrix C by $C(n, m) := c(x_n, y_m)$. An alignment between X and Y is defined as a sequence $p = (p_1, \ldots, p_L)$ with $p_\ell = (n_\ell, m_\ell) \in [1:N] \times [1:M]$ for $\ell \in [1:L]$ satisfying $1 = n_1 \leq n_2 \leq \ldots \leq n_L = N$ and $1 = m_1 \leq m_2 \leq \ldots \leq m_L = M$ (boundary and monotonicity condition), as well as $p_{\ell+1} - p_\ell \in \{(1,0), (0,1), (1,1)\}$ (step size condition). An alignment p having minimal total cost among all possible alignments is called an *optimal alignment*. To determine such an optimal alignment, one recursively computes an $(N \times M)$-matrix D, where the matrix entry $D(n, m)$ is the total cost of the optimal alignment between $(x_1, \ldots, x_n)$ and $(y_1, \ldots, y_m)$:

$$D(n,m) := \min \begin{cases} D(n-1, m-1) + w_1 C(n,m), \\ D(n-1, m) + w_2 C(n,m), \\ D(n, m-1) + w_3 C(n,m), \end{cases}$$

for $n, m > 1$. Furthermore, $D(n, 1) := \sum_{k=1}^{n} w_2 C(k, 1)$ for $n > 1$, $D(1, m) = \sum_{k=1}^{M} w_3 C(1, k)$ for $m > 1$, and $D(1, 1) := C(1, 1)$. The weights $(w_1, w_2, w_3) \in \mathbb{R}_+^3$ can be used to adjust the preference over the three step sizes. By tracking the choice for the minimum starting from $D(N, M)$ back to $D(1, 1)$, an optimal alignment can be derived in a straightforward way [2]. In a musical context, $\mathcal{F}$ typically denotes the space of normalized chroma features, c is usually a cosine (or Euclidean) distance with weights set to $(w_1, w_2, w_3) = (2, 1, 1)$ to remove a bias for the diagonal direction [2, 11].

A main difficulty in aligning music stems from the degree of freedom a musician has in interpreting a score, in particular regarding the local tempo, balance (relative loudness of concurrent notes), articulation and playing style. If several differences occur together, standard alignment methods sometimes fail to identify the musically correct alignment. In Fig. 1(b)/(c), we see chroma features for two interpretations of Chopin Op. 24 No. 2 measures 115-120 (Fig. 1(a)) as performed by Magin and by Indjic, respectively. Besides the tempo, we see differences in the interpretation of pauses (the uniform energy distributions in the features correspond to silence), articulation and in the balance (relative loudness of notes). In this case, the differences are significant such that pairwise DTW-based approaches [10, 11] fail to compute the correct alignment, see upper path in Fig. 1(d). The red dots indicate corresponding beat positions in the two versions.

2.2 Joint Alignment of Multiple Versions

Comparing several versions of a piece, interpretations vary in different ways and to different extents. If several versions of a piece are available, each version provides an example of how a specific position in a piece can be realized, and this additional information can be used to stabilize the alignment for difficult sections. A straightforward strategy to compute a joint alignment could be to extend DTW to allow for more than two versions. For example, to align three versions, one can define an order-3 cost tensor in a straightforward way and apply the same dynamic programming techniques as used in DTW [12] (note that a cost matrix for two versions is an order-2 tensor). However, assuming that each feature sequence to be aligned is roughly of length N, the time and memory requirement to align K recordings would be in $O(N^K)$, which prohibits the alignment of more than a very few recordings.

In computational biology, multiple sequence alignment is a well-studied problem. Most popular are so called *profile-based methods* and *progressive alignment methods* [12]. Profile-based methods employ a specific type of HMM, which is trained via Expectation-Maximization (EM) on the set of feature sequences to be aligned. Each state of the resulting *profile-HMM* corresponds to a position in a so called average-sequence: the sequence of means of the observation probabilities of the HMM-states, see [12] for details. A multiple synchronization is then computed by aligning each sequence to the average-sequence via the Viterbi algorithm. This procedure has been attempted in a musical context with limited success [13]. We believe this is due to, using EM training, whereby aligned features are essentially averaged (with Gaussian observation probabilites), which results in a loss of information and can lead to a loss of alignment accuracy.

Using progressive alignment such averaging is not necessary. The underlying idea is to successively build a data structure referred to as a *template*, which provides efficient access to several aligned feature sequences, see Fig. 2(a).

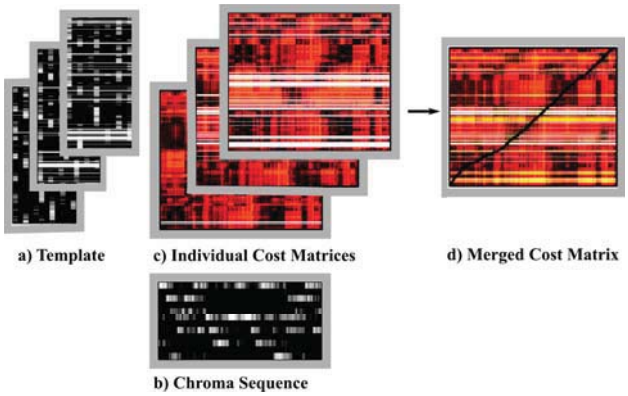


Figure 2. Progressive alignment: Three aligned chroma sequences contained in the template **(a)** are compared to the chroma sequence **(b)**. The resulting individual cost matrices **(c)** are merged into one **(d)**, which is used to compute the alignment. The white lines in (a) and (c) indicate the positions of gap symbols.

By comparing a given feature sequence (Fig. 2(b)) to the sequences contained in the template, the alignment can be computed not only using one cost matrix (as in pairwise alignment) but several matrices in parallel - one for each sequence in the template (Fig. 2(c)). By suitably combining the information provided by each individual cost matrix, the influence of strong local differences on the alignment, that often only occur between specific pairs of versions, can be attenuated. As shown in Section 3, this can lead to a significant boost in alignment robustness.

To describe this procedure in more detail, we assume that we have K different versions of a piece and that their feature sequences are denoted by $X^k = (x^k_1, \ldots, x^k_{N_k})$ for $k \in [1:K]$. In each step of the progressive alignment, the template Z contains several of these feature sequences that have been stretched to have the same length. Initially, Z only consists of X^1. The remaining feature sequences are then successively aligned to Z, and after each alignment Z is updated by adding one more sequence. To this end, let $\widetilde{Z} = (\widetilde{z}_1, \ldots, \widetilde{z}_{\widetilde{L}})$ denote the current template which contains $k-1$ sequences of length $\widetilde{L}$ (i.e. each $\widetilde{z}_\ell$ contains $k-1$ features), X^k the sequence to be aligned, and $p = (p_1, \ldots, p_L) = \big((n_1, m_1), \ldots, (n_L, m_L)\big)$ an alignment between $\widetilde{Z}$ and X^k. Intuitively, to add X^k to $\widetilde{Z}$, we use p to stretch $\widetilde{Z}$ and X^k such that corresponding features are aligned and become part of the same element of Z. However, whenever features need to be copied to do the stretching (step sizes $(1,0)$ and $(0,1)$), we rather insert a special *gap* symbol instead of the features themselves. More precisely, let $Z = (z_1, \ldots, z_L)$ denote the updated template, $z_n(k)$ denote the k-th feature in the n-th element of Z, and G denote the gap symbol [1]. Set $z_1 = (\widetilde{z}_1(1), \ldots, \widetilde{z}_1(k-1), x^k_1)$, then for $l = (2, 3, \ldots, L)$:

$$z_\ell = \begin{cases} (\widetilde{z}_{n_\ell}(1), \ldots, \widetilde{z}_{n_\ell}(k-1), x^k_{m_\ell}), & p_\ell - p_{\ell-1} = (1,1) \\ (\widetilde{z}_{n_\ell}(1), \ldots, \widetilde{z}_{n_\ell}(k-1), G), & p_\ell - p_{\ell-1} = (1,0) \\ (G, \ldots, G, x^k_{m_\ell}), & p_\ell - p_{\ell-1} = (0,1) \end{cases}$$

[1] Since chroma features contain only non-negative entries, the gap symbol can often be encoded as a pseudo-feature having negative entries.

The gap symbol and its influence will be further discussed in Section 3.

The alignment procedure itself is almost identical to standard DTW; only the local cost measure has to be adjusted to take the properties of the template into account. For a template Z comprising $k-1$ feature sequences and a feature sequence X, we define a template-aware cost function $c_T : (\mathcal{F} \cup G)^{k-1} \times \mathcal{F} \to \mathbb{R}$ as

$$c_T(z_n, x_m) = \sum_{r=1}^{k-1} \begin{cases} c(z_n(r), x_m), & z_n(r) \neq G, \\ \mathcal{C}_G, & z_n(r) = G, \end{cases}$$

where $\mathcal{C}_G > 0$ is a constant referred to as the *gap penalty*.

The influence a single additional recording can have using progressive alignment is illustrated in Fig. 1(d). Here, we included a third performance by Poblocka in the alignment, which could be considered as being "between" the two versions shown in Fig. 1 in terms of articulation style and balance. As we can see, the resulting path (black) follows the ground-truth markings (red dots) quite closely and improves significantly over the pairwise result.

2.3 Order of Alignments and Iterative Processing

The alignment of the first two versions in our progressive approach is equivalent to standard pairwise alignment. Errors in this first step influence to some degree all subsequent alignment steps. We discuss now two strategies that can help to increase the reliability of the first few alignments in our progressive approach. First, the order in which the alignments are computed is of importance, and we should start with recordings that are easy to align. In computational biology, a common approach to identify a reasonable order is referred to as the *guide tree approach* [12]. While there are various ways to implement such an approach, we consider the following procedure. First, for each pair of recordings, we compute the total cost of an optimal alignment between the pair to identify the pair having the lowest *average cost*, which is defined as the total cost of the alignment divided by its length L. We call the feature sequences for the recordings in this pair X^1 and X^2. For the next recording, we identify the one being jointly closest to X^1 and X^2. To this end, we sum for each of the remaining recordings the average cost of the alignments between the recording and X^1, and the recording and X^2. We call the feature sequence of the recording with the lowest sum X^3. We continue with this procedure until all recordings are in order. We refer to this strategy as *DTW-cost-based order*.

While this strategy leads to a useful order, its computational costs are significant. In our experiments, we found an alternative based on a much simpler strategy: We sorted the versions according to their length, starting with the shortest recordings. In the following, we refer to this strategy as *length-based order*. In Section 3, we compare both ordering strategies and discuss their behavior.

A second strategy to improve the reliability of the first alignments is referred to as *iterative progressive alignment*. The idea behind this strategy is, after all versions are aligned and included in the template, to remove one version from

ID	Piece	No. Rec.	No. Pairs
M17-4	Opus 17 No. 4	62	1891
M24-2	Opus 24 No. 2	62	1891
M30-2	Opus 30 No. 2	34	561
M63-3	Opus 63 No. 3	81	3240
M68-3	Opus 68 No. 3	49	1176

Table 1. Chopin Mazurkas and their identifiers used in our experiments. The last two columns indicate the number of performances available for the respective piece and the number of evaluated unique pairs.

the template and realign it, starting with the first version that was aligned. This way, errors made early in the progressive alignment can potentially be corrected. We implemented this extension as well and discuss it in Section 3.

2.4 Increasing the Computational Efficiency and Alignment Accuracy

Since progressive alignment shares its algorithmic roots with standard DTW, we can incorporate extensions that were successfully used with DTW-based methods. In particular, the methods described in [10, 14] employ a variant of DTW referred to as multiscale DTW (FastDTW) to increase the computational efficiency. The general idea is to recursively project an alignment computed at a coarse feature resolution level to a next higher resolution, and to refine the projected alignment on that resolution. This way, the matrix D only has to be evaluated around the projected path. This multiscale approach typically leads to a significant drop in runtime by up to a factor of 30, see [14].

Furthermore, the authors in [10] introduce a type of features that indicate onset positions separately for each chroma. These chroma-based onset features (DLNCO features) are then combined with normalized chroma features. As shown by the experiments in [10], these combined features can lead to a significant increase in alignment accuracy for pairwise methods. In the following, we employ the same features and cost measure as used in [10].

3. EXPERIMENTS

To illustrate the performance of our proposed method as well as the influence of certain parameters, we conducted a series of experiments using recordings taken from the Mazurka Project [2] , which compiled a database of over 2700 recorded performances by more than 130 distinct pianists for 49 Mazurkas composed by Frédéric Chopin. The recordings are dated between 1902 and today, and were made under strongly varying recording conditions. For our experiments, we employ a subset of five Mazurkas and 288 recordings, for which manually annotated beat positions are available, see Table 1. Performances with structural differences compared to the majority of recordings (such as additional repetitions of a part of a piece) were excluded from our experiments.

[2] http://www.mazurka.org.uk

3.1 Evaluation Measure

To evaluate the accuracy of an alignment between two different versions of a piece, we employ the beat annotations as ground truth. To this end, we use the alignment to locate for each annotated beat position in the one version a corresponding position in the other version. Using the manual beat annotations for the other version, we can then compute the absolute difference between the correct beat position and the one obtained from the alignment. By averaging these differences for all beats, we obtain the *average beat deviation (ABD)* for a given alignment, which we measure in milli-seconds. For our evaluation, we compute this measure for each Mazurka and each pair of recordings. For example, for M17-4 our setup contains 62 recordings, which results in $\binom{62}{2} = 1891$ unique pairs and corresponding average beat deviation values, see Table 1.

3.2 Pairwise vs Progressive Alignment

In a first experiment, we compare the alignment accuracy for pairwise and progressive alignment. Since the pairwise method described in [10] employs the same features and cost measure as our proposed progressive method, we use [10] as a baseline (other pairwise methods [11] showed a similar behavior). In particular, we use a temporal resolution of 20ms for both chroma and onset-indicator (DLNCO) features. The DTW weights are set to $(w_1, w_2, w_3) = (2, 1.5, 1.5)$. As proposed in [10], we use the cosine distance for the chroma features and the Euclidean distance for the DLNCO features. Moreover, for our proposed progressive alignment, we use the length-based alignment order and set the gap penalty parameter to the highest value the cost measure c can assume. The distribution of the average beat deviation (ABD) values for all pairs is summarized for each of the five Mazurkas separately in the boxplots [3] shown in Fig. 3, as well as in column A and B in Table 2.

Comparing the results for pairwise and progressive alignment, we can see that the mean ABD drops slightly using the progressive approach for most examples. For example, the mean ABD for M17-4 drops from 68ms using pairwise alignment to 59ms using our progressive method (decrease by 13%). On average, the mean ABD drops by 14%. More importantly though, the progressive alignment is significantly more stable. In particular, the inter-quartile range is smaller for all five Mazurkas using the progressive alignment (Fig. 3). Further, the number of alignments with a very high ABD is significantly reduced. This can be measured by the standard deviation (std), which for M17-4 using pairwise alignment is 19ms, while progressive alignment leads to an std of 12ms. This difference is even greater for other Mazurkas (M24-2 and M63-3). On average, the std is reduced by more than 50%. So overall, while our proposed procedure also led to an increase in

[3] We use standard boxplots: the red bar indicates the median, the blue box gives the $25th$ and $75th$ percentiles (p_{25} and p_{75}), the black bars correspond to the smallest data point greater than $p_{25} - 1.5(p_{75} - p_{25})$ and the largest data point less than $p_{75} + 1.5(p_{75} - p_{25})$, and the red crosses are called outliers.

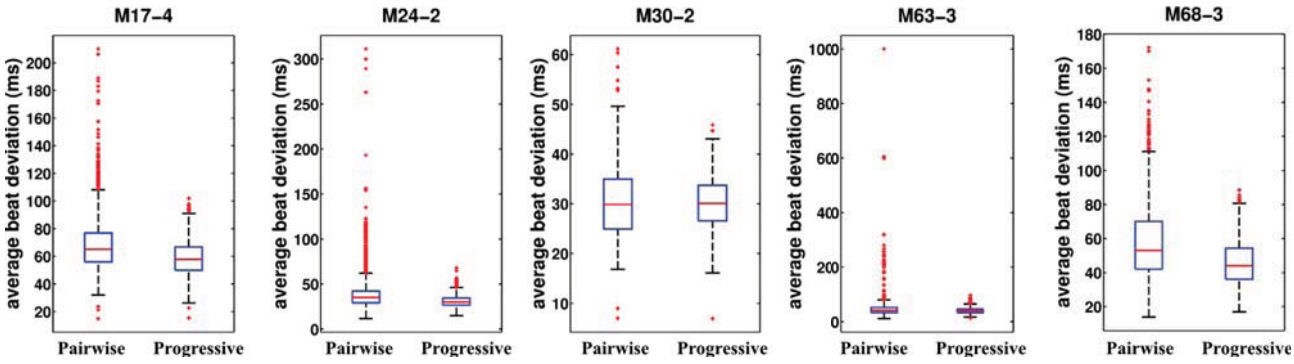


Figure 3. Comparison of the baseline pairwise alignment method with our proposed progressive alignment method. The boxplots illustrate the distribution of the average beat deviation values for each Mazurka separately.

M17-4	**[A]**	**[B]**	**[C]**	**[D]**	**[E]**	**[F]**	**[G]**
min	15	15	17	15	15	15	19
mean	68	59	68	63	76	80	91
max	210	102	118	116	789	129	252
std	19	12	13	13	94	13	22
M24-2							
min	12	15	17	12	15	16	11
mean	39	31	38	33	31	46	56
max	311	68	118	59	68	98	320
std	20	6	12	7	6	9	22
M30-2							
min	7	7	7	7	16	6	6
mean	30	30	31	29	31	40	43
max	61	46	49	53	46	64	80
std	8	5	6	6	5	7	9
M63-3							
min	11	13	15	12	13	14	9
mean	46	40	46	40	40	53	62
max	1000	97	99	99	97	109	1000
std	32	11	12	11	11	11	33
M68-3							
min	14	17	21	15	17	21	12
mean	58	46	57	53	46	71	86
max	172	89	144	105	89	179	335
std	23	13	18	15	13	21	34

Table 2. Statistics over the average beat deviation (ABD) values for the five Mazurkas and for 7 different alignment approaches (see text). **[A]**: Pairwise alignment. **[B]**: Proposed progressive alignment. **[C]**: Proposed without gap symbols. **[D]**: Proposed using DTW-cost-based alignment order. **[E]**: Proposed using iterative alignment. **[F]**: Proposed without DLNCO features. **[G]**: Pairwise without DLNCO features. All values in milli-seconds.

alignment accuracy on average, the main effect is a gain in robustness against strongly incorrect alignments.

3.3 Gap Penalties

In the next experiment, we investigate the influence of the gap penalty parameter by testing a slightly modified version of our proposed method. To this end, we modify the way the template is creating by setting $z_\ell = (\widetilde{z}_{n_\ell}(1), \ldots, \widetilde{z}_{n_\ell}(k-1), x^k_{m_\ell})$ for $\ell \in [1:L]$, i.e. we do not insert gap symbols but copy features as necessary to create the new template (comparing to Section 2.2). The results using this modification are shown in column C in Table 2. Comparing these values to our proposed method (column B) and the reference pairwise method (column A), we see that this gap-less version typically improves over pairwise alignment in terms of maximum ABD values and the standard deviation, just as the proposed method. For example, for M17-4, the max ABD in column A is 210ms, while the max ABD in column C is 118ms. However, we do not observe a decrease in the mean ABD compared to pairwise alignment. For example, for M17-4, while using gaps the mean ABD drops from 68ms (column A) to 59ms (column B), it stays on a similar level in column C (68ms). The reason could be that by copying the features to create the template, some temporal precision is lost and this results in a minor loss of alignment accuracy.

3.4 Alignment Order

Next, we investigate the influence of the order in which we compute the progressive alignment, comparing the length-based and the DTW-cost-based strategy (see Section 2.3). The results are given in columns B and D of Table 2, respectively. As we can see, there are no significant differences between both strategies. For example, for M17-4, the mean ABD using the length-based strategy is 59ms (column B), while using the DTW-cost-based strategy the ABD slightly increases to 63ms. The other statistical values show a similar behavior. Since these results do not disclose any obvious advantages for the DTW-cost-based strategy, we therefore propose to simply use the length-based strategy. Interestingly, using the length-based strategy but starting with the longest recordings led to worse results.

Since (local) tempo differences can usually be handled quite well using DTW, it is not obvious why sorting by length yields a useful order. However, the fact that it does could indicate that there might be a correlation between the chosen tempo and other expressive parameters, such as articulation or balance, as strong differences in these parameters typically lead to difficulties for the alignment. Furthermore, the fact that according to our evaluation the shorter recordings were easier to align, could indicate that a high tempo could limit the range of possible realizations of expressive parameters in a performance. However, further studies would be necessary to confirm such theories.

3.5 Iterative Alignment

In a further experiment, we investigate whether iterative processing could further improve the alignment accuracy, compare Section 2.3. To this end, we use two iterations: the first iteration corresponds to progressive alignment, and in the second iteration, each version is removed from the template once and is then realigned. The results for this extension are given in column E of Table 2. Overall, the iterative variant led to a slight decrease in ABD in almost all examples, which is not even visible in Table 2 as we rounded all values. On the contrary, we observed a significant increase in ABD for M17-4 using the iterative variant. Here, the realignment led to a misalignment of several shorter recordings. Therefore, the results do not indicate any significant advantages of using iterative alignment.

3.6 Influence of Onset-Indicator Features

In a final experiment, we investigate the influence of the chroma-based onset-indicator (DLNCO) features [10] on the alignment accuracy when using progressive alignment. To this end, we disabled the DLNCO features in our proposed method, and computed the alignment only based on the normalized chroma features. The results of this experiment are given in column F in Table 2. As a further reference, we disabled the DLNCO features in our baseline pairwise method as well (column G).

As we can see, the minimum over the ABD values remains unaffected for most of the Mazurkas, which means that easy to align pairs can be aligned with chroma features alone just as well. For example, for M17-4, the minimum value in column F is identical to the one in column B. However, we see a significant increase in ABD in all other statistical values. For example, the mean ABD for M17-4 for our proposed method including DLNCO features is 59ms (column B), while disabling the DLNCO leads to a mean ABD of 80ms (column F). Similar observations can be made comparing the pairwise results. Overall, the results seem to indicate that including onset-indicator features indeed leads to a significant increase in alignment accuracy also for progressive alignments.

4. CONCLUSION

In this paper, we introduced a method for aligning multiple versions of a piece of music in a joint way. The availability of multiple versions to compare against during the alignment, stabilized the comparison for hard-to-align recordings and led to an overall increase in alignment accuracy and, in particular, in alignment robustness. Our experiments using real-world recordings from the Mazurka Project demonstrated that our proposed method can indeed be used to raise the alignment accuracy compared to previous methods that are limited to pairwise alignments. For the future, we plan to further investigate the behaviour of our procedure. In particular, we plan to analyze how other ordering strategies influence the alignment accuracy. We will also further explore different strategies to implement a cost for the gap symbol and to make it more adaptive.

Acknowledgements: This work was partly funded by the China Scholarship Council (CSC), EPSRC Grant EP/J010375/1, and the Queen Mary Postgraduate Research Fund (PGRF).

5. REFERENCES

[1] M. Müller, M. Clausen, V. Konz, S. Ewert, and C. Fremerey, "A multimodal way of experiencing and exploring music," *Interdisciplinary Science Reviews (ISR)*, vol. 35, no. 2, pp. 138–153, 2010.

[2] R. B. Dannenberg and C. Raphael, "Music score alignment and computer accompaniment," *Communications of the ACM, Special Issue: Music Information Retrieval*, vol. 49, no. 8, pp. 38–43, 2006.

[3] A. Arzt, S. Böck, S. Flossmann, H. Frostel, M. Gasser, and G. Widmer, "The complete classical music companion v0.9," in *Proceedings of the AES International Conference on Semantic Audio*, London, UK, 18–20 2014, pp. 133–137.

[4] N. Montecchio and A. Cont, "A unified approach to real time audio-to-score and audio-to-audio alignment using sequential Montecarlo inference techniques," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Prague, Czech Republic, 2011, pp. 193–196.

[5] Z. Duan and B. Pardo, "A state space model for online polyphonic audio-score alignment," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Prague, Czech Republic, 2011, pp. 197–200.

[6] G. Widmer, S. Dixon, W. Goebl, E. Pampalk, and A. Tobudic, "In search of the Horowitz factor," *AI Magazine*, vol. 24, no. 3, pp. 111–130, 2003.

[7] J. Serrà, E. Gómez, P. Herrera, and X. Serra, "Chroma binary similarity and local alignment applied to cover song identification," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 16, pp. 1138–1151, 2008.

[8] S. Ewert, B. Pardo, M. Müller, and M. D. Plumbley, "Score-informed source separation for musical audio recordings: An overview," *IEEE Signal Processing Magazine*, vol. 31, no. 3, pp. 116–124, May 2014.

[9] C. Joder, S. Essid, and G. Richard, "A conditional random field framework for robust and scalable audio-to-score matching," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 8, pp. 2385–2397, 2011.

[10] S. Ewert, M. Müller, and P. Grosche, "High resolution audio synchronization using chroma onset features," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Taipei, Taiwan, 2009, pp. 1869–1872.

[11] S. Dixon and G. Widmer, "MATCH: A music alignment tool chest," in *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, London, GB, 2005, pp. 492–497.

[12] R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison, *Biological Sequence Analysis : Probabilistic Models of Proteins and Nucleic Acids*. New York, USA: Cambridge University Press, 1999.

[13] H. I. Robertson, "Testing a new tool for alignment of musical recordings," Master's thesis, McGill University, 2013.

[14] M. Müller, H. Mattes, and F. Kurth, "An efficient multiscale approach to audio synchronization," in *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, Victoria, Canada, 2006, pp. 192–197.

FORMALIZING THE PROBLEM OF MUSIC DESCRIPTION

Bob L. Sturm
Aalborg University
Denmark
bst@create.aau.dk

Rolf Bardeli
Fraunhofer IAIS
Germany
rolf.bardeli@iais.fraunhofer.de

Thibault Langlois
Lisbon University
Portugal
tl@di.fc.ul.pt

Valentin Emiya
Aix-Marseille Université
CNRS UMR 7279 LIF
valentin.emiya@lif.univ-mrs.fr

ABSTRACT

The lack of a formalism for "the problem of music description" results in, among other things: ambiguity in what problem a music description system must address, how it should be evaluated, what criteria define its success, and the paradox that a music description system can reproduce the "ground truth" of a music dataset without attending to the music it contains. To address these issues, we formalize the problem of music description such that all elements of an instance of it are made explicit. This can thus inform the building of a system, and how it should be evaluated in a meaningful way. We provide illustrations of this formalism applied to three examples drawn from the literature.

1. INTRODUCTION

Before one can address a problem with an algorithm (a finite series of well-defined operations that transduce a well-specified input into a well-specified output) one needs to define and decompose that problem in a way that is compatible with the formal nature of algorithms [17]. A very simple example is the problem of adding any two positive integers. Addressing this problem with an algorithm entails defining the entity "positive integer", the function "adding", and then producing a finite series of well-defined operations that applies the function to an input of two positive integers to output the correct positive integer.

A more complex example is "the problem of music description." While much work in music information retrieval (MIR) has proposed systems to attempt to address the problem of music description [4, 12, 29], and much work attempts to evaluate the capacity of these systems for addressing that problem [9, 20], we have yet to find any work that actually *defines* it. (The closest we have found is that of [24].) Instead, there are many allusions to the problem: predict the "genre" of a piece of recorded music [25]; label music with "useful tags" [1]; predict what a listener will "feel" when "listening" to some music [29]; find music "similar" to some other music [26]. These allusions are deceptively simple, however, since behind them lie many problems and questions that have major repercussions on the design and evaluation of any proposed system. For example, What is "genre"? What is "useful"? How is "feeling" related to "listening"? "Similar" in what respects?

With respect to the problem of music description, some work in MIR discusses the meaningfulness, worth, and futility of designing artificial systems to describe music [28]; the idea of and the difficulty in "ground truth" [3, 6, 15]; the size of datasets [5], a lack of statistics [10], the existence of bias [16], and the ways such systems are evaluated [21, 22, 27]. Since a foundational goal of MIR is to develop systems that can imitate the human ability to describe music, these discussions are necessary. However, what remains missing is a formal definition of the problem of music description such that it can be addressed by algorithms, and relevant and valid evaluations can be designed.

In this work, we formalize the problem of music description and try to avoid ambiguity arising from semantics. This leads to a rather abstract form, and so we illustrate its aspects using examples from the literature. The most practical benefit of our formalization is a specification of all elements that should be explicitly defined when addressing an instance of the problem of music description.

2. FORMALISM

We start our formalization by defining the domain of the problem of music description. In particular, we discriminate between the music that is to be described and a recording of it since the former is intangible and the latter is data that a system can analyze. We then define the problem of music description, a recorded music description system (RMDS), and the analysis of such a system. This leads to the central role of the use case.

2.1 Domain

Denote a *music universe*, Ω, a set of music, e.g., Vivaldi's "The Four Seasons", the piano part of Gershwin's "Rhapsody in Blue", and the first few measures of the first movement of Beethoven's Fifth Symphony. A member of Ω is intangible. One cannot hear, see or point to any member of Ω; but one can hear a performance *of* Vivaldi's "The Four Seasons", read sheet music notating the piano part *of* Gershwin's "Rhapsody in Blue", and point to a printed score *of* Beethoven's Fifth Symphony. Likewise, a recorded performance of Vivaldi's "The Four Seasons" is *not* Vivaldi's "The Four Seasons", and sheet music notating the piano part of Gershwin's "Rhapsody in Blue" is *not* the piano part of Gershwin's "Rhapsody in Blue".

In the tangible world, there may exist tangible recordings of the members of Ω. Denote the *tangible music recording universe* by $\mathcal{R}_\Omega$. A member of $\mathcal{R}_\Omega$ is a recording of an element of $\omega \in \Omega$. A *recording* is a tangible object, such as a printed CD or score. Denote one recording of $\omega \in \Omega$ as $r_\omega \in \mathcal{R}_\Omega$. There might be many recordings of an ω in $\mathcal{R}_\Omega$. We say the music ω is *embedded* in r_ω; it enables for a listener an indirect sense of ω. For instance, one can hear a live or recorded performance of ω, and one can read a printed score of ω. The acknowledgment of and distinction between intangible music and tangible recordings *of* music is essential since systems cannot work with intangible music, but only tangible recordings.

2.2 Music Description and the Use Case

Denote a *vocabulary*, $\mathcal{V}$, a set of symbols or tokens, e.g., "Baroque", "piano", "knock knock", scores employing common practice notation, the set of real numbers $\mathbb{R}$, other music recordings, and so on. Define the *semantic universe* as

$$\mathcal{S}_{\mathcal{V},A} := \{s = (v_1, \ldots, v_n) | n \in \mathbb{N}, \\ \forall 1 \leq i \leq n [v_i \in \mathcal{V}] \wedge A(s)\} \quad (1)$$

where $A(\cdot)$ encompasses a semantic rule, for instance, restricting $\mathcal{S}_{\mathcal{V},A}$ to consist of sequences of cardinality 1. Note that the *description* s is a sequence, and not a vector or a set. This permits descriptions that are, e.g., time-dependent, such as envelopes, if $\mathcal{V}$ and $A(\cdot)$ permit it. In that case, the order of elements in s could be alternating time values with envelope values. Descriptions could also be time-frequency dependent.

We define *music description* as pairing an element of Ω or $\mathcal{R}_\Omega$ with an element of $\mathcal{S}_{\mathcal{V},A}$. The *problem of music description* is to make the pairing *acceptable with respect to a use case*. A *use case* provides specifications of Ω and $\mathcal{R}_\Omega$, $\mathcal{V}$ and $A(\cdot)$, and success criteria. *Success criteria* describe how music or a music recording should be paired with an element of the semantic universe, which may involve the sanity of the decision (e.g., tempo estimation must be based on the frequency of onsets), the efficiency of the decision (e.g., pairing must be produced under 100 ms with less than 10 MB of memory), or other considerations.

To make this clearer, consider the following use case. The music universe Ω consists of performances by Buckwheat Zydeco, movements of Vivaldi's "The Four Seasons", and traditional Beijing opera. The tangible music recording universe $\mathcal{R}_\Omega$ consists of all possible 30-second digital audio recordings of the elements in Ω. Let the vocabulary $\mathcal{V} = \{$"Blues", "Classical"$\}$; and define $A(s) := [|s| \in \{0, 1\}]$. The semantic universe is thus, $\mathcal{S}_{\mathcal{V},A} = \{(), ($"Blues"$), ($"Classical"$)\}$. There are many possible success criteria. One is to map all recordings of Buckwheat Zydeco to "Blues", map all recordings of Vivaldi's "The Four Seasons" to "Classical", and map all recordings of traditional Beijing opera to neither. Another is to map no recordings of Buckwheat Zydeco and Vivaldi's "The Four Seasons" to the empty sequence, and to map any recording of traditional Beijing opera to either non-empty sequence with a probability less than 0.1.

2.3 Recorded Music Description Systems

A *recorded music description system* (RMDS) is a map from the tangible music recording universe to the semantic universe:

$$\mathcal{S} : \mathcal{R}_\Omega \to \mathcal{S}_{\mathcal{V},A}. \quad (2)$$

Building an RMDS means making a map according to well-specified criteria, e.g., using expert domain knowledge, automatic methods of supervised learning, and a combination of these. An instance of an RMDS is a specific map that is already built, and consists of four kinds of components [21]: algorithmic (e.g., feature extraction, classification, pre-processing), instruction (e.g., description of $\mathcal{R}_\Omega$ and $\mathcal{S}_{\mathcal{V},A}$), operator(s) (e.g., the one inputting data and interpreting output), and environmental (e.g., connections between components, training datasets). It is important to note that $\mathcal{S}$ is not restricted to map any recording to a single element of $\mathcal{V}$. Depending on $\mathcal{V}$ and $A(\cdot)$, $\mathcal{S}_{\mathcal{V},A}$ could consist of sequences of scalars and vectors, sets and sequences, functions, combinations of all these, and so on. $\mathcal{S}$ could thus map a recording to many elements of $\mathcal{V}$.

One algorithmic component of an RMDS is a *feature extraction algorithm*, which we define as

$$\mathcal{E} : \mathcal{R}_\Omega \to \mathcal{S}_{\mathbb{F},A'} \quad (3)$$

i.e., a map from $\mathcal{R}_\Omega$ to a semantic universe built from the vocabulary of a feature space $\mathbb{F}$ and semantic rule $A'(\cdot)$. For instance, if $\mathbb{F} := \mathbb{C}^M$, $M \in \mathbb{N}$, and $A'(s) := [|s| = 1]$, then the feature extraction maps a recording to a single M-dimensional complex vector. Examples of such a map are the discrete Fourier transform, or a stacked series of vectors of statistics of Mel frequency cepstral coefficients. Another algorithmic component of an RMDS is a *classification algorithm*, which we define:

$$\mathcal{C} : \mathcal{S}_{\mathbb{F},A'} \to \mathcal{S}_{\mathcal{V},A} \quad (4)$$

i.e., a map from one semantic universe to another. Examples of such a map are k-nearest neighbor, maximum likelihood, support vector machine, and a decision tree.

To make this clearer, consider the RMDS named "RT GS" built by Tzanetakis and Cook [25]. $\mathcal{E}$ maps sampled audio signals of about 30-s duration to $\mathcal{S}_{\mathbb{F},A'}$, defined by single 19-dimensional vectors, where one dimension is spectral centroid mean, another is spectral centroid variance, and so on. $\mathcal{C}$ maps $\mathcal{S}_{\mathbb{F},A'}$ to $\mathcal{S}_{\mathcal{V},A}$, which is defined by $\mathcal{V} = \{$"Blues", "Classical", "Country", "Disco", "Hip hop", "Jazz", "Metal", "Pop", "Reggae", "Rock"$\}$, and $A(s) := [|s| = 1]$. This mapping involves maximizing the likelihood of an element of $\mathcal{S}_{\mathbb{F},A'}$ among ten multivariate Gaussian models created with supervised learning.

Supervised learning involves automatically building components of an $\mathcal{S}$, or defining $\mathcal{E}$ and $\mathcal{C}$, given a *training recorded music dataset*: a sequence of tuples of recordings sampled from $\mathcal{R}_\Omega$ and elements of $\mathcal{S}_{\mathcal{V},A}$, i.e.,

$$\mathcal{D} := \{(r_i, s_i) \in R_\Omega \times \mathcal{S}_{\mathcal{V},A} | i \in \mathcal{I}\} \quad (5)$$

The set $\mathcal{I}$ indexes the dataset. We call the sequence $(s_i)_{i \in \mathcal{I}}$ the *ground truth* of $\mathcal{D}$. In the case of RT GS, its training

recorded music dataset contains 900 tuples randomly selected from the dataset *GTZAN* [22,25]. These are selected in a way such that the ground truth of $\mathcal{D}$ has no more than 100 of each element of $\mathcal{S}_{\mathcal{V},A}$.

2.4 Analysis of Recorded Music Description Systems

Given an RMDS, one needs to determine whether it addresses the problem of music description. Simple questions to answer are: does Ω and $\mathcal{R}_\Omega$ of the RMDS encompass those of the use case? Does the $\mathcal{S}_{\mathcal{V},A}$ of the RMDS encompass that of the use case? A more complex question could be, does the RMDS meet the success criteria of the use case? This last question involves the design, implementation, analysis, and interpretation of valid experiments that are relevant to answering hypotheses about the RMDS and success criteria [21, 27]. Answering these questions constitutes an *analysis* of an RMDS.

Absent explicit success criteria of a use case, a standard approach for evaluating an RMDS is to compute a variety of *figures of merit* (FoM) from its "treatment" of the recordings of a testing $\mathcal{D}$ that exemplify the input/output relationships sought. Examples of such FoM are mean classification accuracy, precisions, recalls, and confusions. An implicit belief is that the correct output will be produced from the input only if an RMDS has learned criteria relevant to describing the music. Furthermore, it is hoped that the resulting FoM reflect the real world performance of an RMDS. The *real world performance* of an RMDS are the FoM that result from an experiment using a testing recording music dataset consisting of *all* members in $\mathcal{R}_\Omega$, rather than a sampling of them. If this dataset is out of reach, statistical tests can be used to determine significant differences in performance between two RMDS (testing the null hypothesis, "neither RMDS has 'learned better' than the other"), or between the RMDS and that of picking an element of $\mathcal{S}_{\mathcal{V},A}$ independent of the element from $\mathcal{R}_\Omega$ (testing the null hypothesis, "The RMDS has learned nothing"). These statistical tests are accompanied by implicit and strict assumptions on the measurement model and its appropriateness to describe the measurements made in the experiment [2, 8].

As an example, consider the evaluation of RT GS discussed above [25]. The evaluation constructs a testing $\mathcal{D}$ from the 100 elements of the dataset *GTZAN* not present in the training $\mathcal{D}$ used to create the RMDS. They treat each of the 100 recordings in the testing $\mathcal{D}$ with RT GS, and compare its output with the ground truth. From these 100 comparisons, they compute the percentage of outputs that match the ground truth (accuracy). Whether or not this is a high-quality estimate of the real world accuracy of RT GS depends entirely upon the definition of Ω, $\mathcal{R}_\Omega$, $\mathcal{S}_{\mathcal{V},A}$, as well as the testing $\mathcal{D}$ and the measurement model of the experiment.

There are many serious dangers to the interpretation of the FoM of an RMDS as reflective of its real world performance: noise in the measurements, an inappropriate measurement model [2], a poor experimental design and errors of the third kind [14], the lack of error bounds or error bounds that are too large [8], and several kinds of bias. One kind of bias comes from the very construction of testing datasets. For instance, if the testing dataset is the same as the training dataset, and the set of recordings in the dataset is a subset of $\mathcal{R}_\Omega$, then the FoM of an RMDS computed from the treatment may not indicate its real world performance. This has led to the prescription in machine learning to use a testing dataset that is disjoint with the training dataset, by partitioning for instance [13]. This, however, may not solve many other problems of bias associated with the construction of datasets, or increase the relevance of such an experiment with measuring the extent to which an RMDS has learned to describe the music in Ω.

2.5 Summary

Table 1 summarizes all elements defined in our formalization of the problem of music description, along with examples of them. These are the elements that must be explicitly defined in order to address an instance of the problem of music description by algorithms. Central to many of these are the definition of a use case, which specifies the music and music recording universe, the vocabulary, the desired semantic universe, *and* the success criteria of an acceptable system. (Note that "use case" is not the same as "user-centered.") If the use case is not unambiguously specified, then a successful RMDS cannot be constructed, relevant and valid experiments cannot be designed, and the analysis of an RMDS cannot be meaningful. Table 1 can serve as a checklist for the extent to which an instance of the problem of music description is explicitly defined.

3. APPLICATION

We now discuss two additional published works in the MIR literature in terms of our formalism.

3.1 Dannenberg et al. [7]

The use cases of the RMDS employed by Dannenberg et al. [7] are motivated by the desire for a mode of communication between a human music performer and an accompanying computer that is more natural than physical interaction. The idea is for the computer to employ an RMDS to describe the acoustic performance of a performer in terms of several "styles." Dannenberg et al. circumvent the need to define any of these "styles" by noting, "what really matters is the ability of the performer to consistently produce intentional and different styles of playing at will" [7]. As a consequence, the use cases and thus system analysis are centered on the performer.

One use case considered by Dannenberg et al. defines $\mathcal{V} = \{$"lyrical", "frantic", "syncopated", "pointillistic", "blues", "quote", "high", "low"$\}$, and the semantic rule $A(s) := [|s| \in \{1\}]$. The semantic universe $\mathcal{S}_{\mathcal{V},A}$ is then all single elements of $\mathcal{V}$. The music universe Ω is all possible music that can be played or improvised by the specific performer in these "styles." The tangible music recording universe $\mathcal{R}_\Omega$ is all possible 5-second acoustic recordings of the elements of Ω. Finally, the success criteria of this particular problem of music description includes the following requirements: reliable for a specific performer in an interactive performance, classifier latency of under

Element (Symbol)	Definition	Example
music universe (Ω)	a set of (intangible) music	{"Automatic Writing" by R. Ashley}
tangible music recording universe ($\mathcal{R}_\Omega$)	a set of tangible recordings of all members of Ω	{R. Ashley, "Automatic Writing", LCD 1002, Lovely Music, Ltd., 1996}
recording (r_ω)	a member of $\mathcal{R}_\Omega$	a 1-second excerpt of the 46 minute recording of "Automatic Writing" from LCD 1002
vocabulary ($\mathcal{V}$)	a set of symbols	{"Robert", "french woman", "bass in other room", "Moog"} $\cup$ $[0, 2760]$
semantic universe ($\mathcal{S}_{\mathcal{V},A}$)	$\{s = (v_1, \ldots, v_n) \vert n \in \mathbb{N}, \forall 1 \leq i \leq n [v_i \in \mathcal{V}] \wedge A(s)\}$, i.e., the set of all sequences of symbols from $\mathcal{V}$ permitted by the semantic rule $A(\cdot)$	{("Robert", 1), ("Robert", "Moog", 4.3), ("french woman", 104.3), ("french woman", "Moog", 459), ...}
semantic rule ($A(s)$)	a Boolean function that defines when sequence s is "permissible"	$A(s) := \big[(\vert s\vert \in \{2, 3, 4, 5\}) \wedge (\{v_1, \ldots, v_{\vert s\vert -1}\} \subseteq$ {"Robert", "french woman", "bass in other room", "Moog"} $\cup \{\}) \wedge (v_{\vert s\vert} \in [0, 2760])\big]$
music description	the pairing of an element of Ω or $\mathcal{R}_\Omega$ with an element of $\mathcal{S}_{\mathcal{V},A}$	label the events (character, time) in recording LCD 1002 of "Automatic Writing" by R. Ashley
the problem of music description	make this pairing acceptable with respect to the success criteria specified by the use case	make this pairing such that F-score of event "Robert" is at least 0.9
use case	specification of $\Omega, \mathcal{R}_\Omega, \mathcal{V}, A(s)$, and success criteria	see all above
system	a connected set of interacting and interdependent components of four kinds (operator(s), instructions, algorithms, environment) that together address a use case	system created in the Audio Latin Genre Classification task of MIREX 2013 by organizer from submission "AP1" and fold 1 of LMD [18]
operators	agent(s) that employ the system, inputting data, and interpreting outputs	Audio Latin Genre Classification organizer of MIREX 2013
instructions	specifications for the operator(s), like an application programming interface	MIREX 2013 input/output specifications for Train/Test tasks; "README" file included with "AP1"
algorithm	a finite series of well-defined ordered operations to transduce an input into an output	"Training.m" and "Classifying.m" MATLAB scripts in "AP1", etc.
environment	connections between components, external databases, the space within which the system operates, its boundaries	folds 2 and 3 of LMD [18], MIREX computer cluster, local MATLAB license file, etc.
recorded music description system (RMDS) ($\mathcal{S}$)	$\mathcal{S} : \mathcal{R}_\Omega \rightarrow \mathcal{S}_{\mathcal{V},A}$, i.e., a map from $\mathcal{R}_\Omega$ to $\mathcal{S}_{\mathcal{V},A}$	"RT GS" evaluated in [25]
feature extraction algorithm ($\mathcal{E}$)	$\mathcal{E} : \mathcal{R}_\Omega \rightarrow \mathcal{S}_{\mathbb{F},A'}$, i.e., a map from $\mathcal{R}_\Omega$ to an element of a semantic universe based on the feature vocabulary $\mathbb{F}$ and semantic rule $A'(s)$	compute using [19] the first 13 MFCCs (including zeroth coefficient) from a recording
feature vocabulary ($\mathbb{F}$)	a set of symbols	$\mathbb{R}^{13}$
classification algorithm ($\mathcal{C}$)	$\mathcal{C} : \mathcal{S}_{\mathbb{F},A'} \rightarrow \mathcal{S}_{\mathcal{V},A}$, i.e., a map from $\mathcal{S}_{\mathbb{F},A'}$ to the semantic universe	single nearest neighbor
recorded music dataset	$\mathcal{D} := (\{r_\omega \in \mathcal{R}_\Omega, s \in \mathcal{S}_{\mathcal{V},A}\}_i)_{i \in \mathcal{I}}$, i.e., a sequence of tuples of recordings and elements of the semantic universe, indexed by $\mathcal{I}$	*GTZAN* [22, 25]
"ground truth" of $\mathcal{D}$	$(s_i)_{i \in \mathcal{I}}$, i.e., the sequence of "true" elements of the semantic universe for the recordings in $\mathcal{D}$	in *GTZAN*: {"blues", "blues", ..., "classical", ..., "country", ...}
analysis of an RMDS	answering whether an RMDS can meet the success criteria of a use case with relevant and valid experiments	designing, implementing, analyzing and interpreting experiments that validly answer, "Can RT GS [25] address the needs of user A?"
experiment	principally in service to answering a scientific question, the mapping of one or more RMDS to recordings of $\mathcal{D}$, and the making of measurements	apply RT GS to *GTZAN*, compare its output labels to "ground truth", and compute accuracy
figure of merit (FoM)	performance measurement of an RMDS from an experiment	classification accuracy of RT GS in *GTZAN*
real world performance of an RMDS	the figure of merit expected if an experiment with an RMDS uses all of $\mathcal{R}_\Omega$	classification accuracy of RT GS

Table 1. Summary of all elements defined in the formalization of the problem of music description, with examples.

5 seconds. The specific definition of "reliable" might include high accuracy, high precision in every class, or only in some classes.

Dannenberg et al. create an RMDS by using a training dataset of recordings curated from actual performances, as well as collected in a more controlled fashion in a laboratory. The ground truth of the dataset is created with input from performers. The feature extraction algorithm includes algorithms for pitch detection, MIDI conversion, and the computation of 13 low-level features from the MIDI data. One classification algorithm employed is maximum likelihood using a naive Bayesian model.

The system analysis performed by Dannenberg et al. involve experiments measuring the mean accuracy of all sys-

tems created and tested with 5-fold cross validation. Furthermore, they evaluate a specific RMDS they create in the context of a live music performance. From this they observe three things: 1) the execution time of the RMDS is under 1 ms; 2) the FoM of the RMDS found in the laboratory evaluation is too optimistic for its real world performance in the context of live performance; 3) using the confidence of the classifier and tuning a threshold parameter provides a means to improve the RMDS by reducing its number of false positives.

3.2 Turnbull et al. [24]

Turnbull et al. [24] propose several RMDS that work with a vocabulary consisting of 174 unique "musically relevant" words, such as "Genre–Brit_Pop", "Usage-Reading", and "NOT-Emotion–Bizarre_/_Weird". $A(s) := [|s| = 10 \wedge \forall i \neq j(v_i \neq v_j)]$, and so the elements of $\mathcal{S}_{\mathcal{V},A}$ are tuples of ten unique elements of $\mathcal{V}$. The music universe Ω consists of at least 502 songs (the size of the CAL500 dataset), such as "S.O.S." performed by ABBA, "Sweet Home Alabama" performed by Lynyrd Skynyrd, and "Fly Me to the Moon" sung by Frank Sinatra. The tangible music recording universe $\mathcal{R}_\Omega$ is composed of MP3-compressed recordings of entire music pieces. The RMDS sought by Turnbull et al. aims "[to be] good at predicting all the words [in $\mathcal{V}$]", or "produce sensible semantic annotations for an acoustically diverse set of songs." Since "good", "sensible" and "acoustically diverse" are not defined, the success criteria is ambiguous. Ω is also likely much larger than 502 songs.

The feature extraction algorithm in the RMDS of Turnbull et al. maps a music recording to a semantic universe built from a feature vocabulary $\mathbb{F} := \mathbb{R}^{39}$, and the semantic rule $A'(s) := [|s| = 10000]$. That is, the algorithm computes from an audio recording 13 MFCC coefficients on 23ms frames, concatenates the first and second derivatives in each frame, and randomly selects 10000 feature vectors from all those extracted. The classification algorithm in the RMDS uses a a maximum a posteriori decision criterion, with conditional probabilities of features modelled by a Gaussian mixture model (GMM) of a specified order. One RMDS uses expectation maximization to estimate the parameters of an 8-order GMM from a training dataset.

Turnbull et al. build an RMDS using a training dataset of 450 elements selected from CAL500. They apply this RMDS to the remaining elements of CAL500, and measure how its output compares to the ground truth. When the ground truth of a recording in CAL500 does not have 10 elements per the semantic rule of the semantic universe, Turnbull et al. randomly add unique elements of $\mathcal{V}$, or randomly remove elements from the ground truth of the recording until it has cardinality 10.

Turnbull et al. compute from an experiment FoM such as mean per-word precision. Per-word precision is, for a $v \in \mathcal{V}$ and when defined, the percentage of correct mappings of the system from the recordings in the test dataset to an element of the semantic universe that includes v. Mean per-word precision is thus the mean of the $|\mathcal{V}|$ per-word precisions. Turnbull et al. compare the FoM of the RMDS to other systems, such as a random classifier and a human. They conclude that their best RMDS is slightly worse than human performance on "more 'objective' semantic categories [like instrumentation and genre]" [24]. The evaluation, measuring the amount of ground truth reproduced by a system (human or not) and not the sensibility of the annotations, has questionable relevance and validity to the ambiguous use case.

4. CONCLUSION

Formalism can reveal when a problem is not adequately defined, and how to explicitly define it in no uncertain terms. An explicit definition of a problem shows how to evaluate solutions in relevant and valid ways. It is in this direction that we move with this paper for the problem of music description, the spirit of which is encapsulated by Table 1. The unambiguous definition of the use case is central for addressing an instance of the problem of music description.

We have discussed several published RMDS within this formalism. The work of Dannenberg et al. [7] provides a good model since its use case and analysis are clearly specified — both center on a specific music performer — and through evaluating the system in the real world they actually complete the research and development cycle to improve the system [27]. The use cases of the RMDS built by Tzanetakis and Cook [25] and Turnbull et al. [24] are not specified. In both cases, a labeled dataset is assumed to provide sufficient definition of the problem. Turnbull et al. suggest a success criterion of annotations being "sensible," but the evaluation only measures the amount of ground truth reproduced. Due to the lack of definition, we are thus unsure what problem either of these RMDS is actually addressing, or whether either of them is actually considering the music [23]. An analysis of an RMDS depends on an explicit use case. The definition of the use case in Dannenberg et al. [7] renders this question irrelevant: all that is needed is that the RMDS meets the success criteria of a given performer, which is tested by performing with it.

While we provide in this paper a formalization of the problem of music description, and a checklist of the components necessary to define an instance of such a problem, it does not describe how to solve any specific problem of music description. We do not derive restrictions on any of the components of the problem definition, or show how datasets should be constructed to guarantee an evaluation can result in good estimates of real world performance. Our future work aims in these directions. We will incoprorate the formalism of the design and analysis of comparative experiments [2,21], which will help define the notions of relevance and validity when it comes to analyzing RMDS. We seek to incorporate notions of learning and inference [13], e.g., to specify what constitutes the building of a "good" RMDS using a training dataset (where "good" depends on the use case). We also seek to explain more formally two paradoxes that have been observed. First, though an RMDS is evaluated in a test dataset to reproduce a large amount of ground truth, it appears to not be a result of the consideration of characteristics in the music universe [20]. Second, though artificial algorithms have

none of the extensive experience humans have in music listening, description, and culture, they can reproduce ground truth consisting of extremely subjective and culturally centered concepts like genre [11].

5. ACKNOWLEDGMENTS

The work of BLS and VE is supported in part by l'Institut français du Danemark, and ARCHIMEDE Labex (ANR-11-LABX- 0033).

6. REFERENCES

[1] J.-J. Aucouturier and E. Pampalk. Introduction – from genres to tags: A little epistemology of music information retrieval research. *J. New Music Research*, 37(2):87–92, 2008.

[2] R. A. Bailey. *Design of comparative experiments*. Cambridge University Press, 2008.

[3] M. Barthet, G. Fazekas, and M. Sandler. Multidisciplinary perspectives on music emotion recognition: Implications for content and context-based models. In *Proc. CMMR*, 2012.

[4] T. Bertin-Mahieux, D. Eck, and M. Mandel. Automatic tagging of audio: The state-of-the-art. In W. Wang, editor, *Machine Audition: Principles, Algorithms and Systems*. IGI Publishing, 2010.

[5] T. Bertin-Mahieux, D. Ellis, B. Whitman, and P. Lamere. The million song dataset. In *Proc. ISMIR*, 2011.

[6] A. Craft, G. A. Wiggins, and T. Crawford. How many beans make five? The consensus problem in music-genre classification and a new evaluation method for single-genre categorisation systems. In *Proc. ISMIR*, pages 73–76, 2007.

[7] R. B. Dannenberg, B. Thom, and D. Watson. A machine learning approach to musical style recognition. In *Proc. ICMC*, pages 344–347, 1997.

[8] E. R. Dougherty and L. A. Dalton. Scientific knowledge is possible with small-sample classification. *EURASIP J. Bioinformatics and Systems Biology*, 2013:10, 2013.

[9] J. Stephen Downie, Donald Byrd, and Tim Crawford. Ten years of ISMIR: Reflections on challenges and opportunities. In *Proc. ISMIR*, pages 13–18, 2009.

[10] A. Flexer. Statistical evaluation of music information retrieval experiments. *J. New Music Research*, 35(2):113–120, 2006.

[11] J. Frow. *Genre*. Routledge, New York, NY, USA, 2005.

[12] Z. Fu, G. Lu, K. M. Ting, and D. Zhang. A survey of audio-based music classification and annotation. *IEEE Trans. Multimedia*, 13(2):303–319, Apr. 2011.

[13] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag, 2 edition, 2009.

[14] A. W. Kimball. Errors of the third kind in statistical consulting. *J. American Statistical Assoc.*, 52(278):133–142, June 1957.

[15] E. Law, L. von Ahn, R. B. Dannenberg, and M. Crawford. Tagatune: A game for music and sound annotation. In *Proc. ISMIR*, pages 361–364, 2007.

[16] E. Pampalk, A. Flexer, and G. Widmer. Improvements of audio-based music similarity and genre classification. In *Proc. ISMIR*, pages 628–233, Sep. 2005.

[17] R. Sedgewick and K. Wayne. *Algorithms*. Addison-Wesley, Upper Saddle River, NJ, 4 edition, 2011.

[18] C. N. Silla, A. L. Koerich, and C. A. A. Kaestner. The Latin music database. In *Proc. ISMIR*, 2008.

[19] M. Slaney. Auditory toolbox. Technical report, Interval Research Corporation, 1998.

[20] B. L. Sturm. Classification accuracy is not enough: On the evaluation of music genre recognition systems. *J. Intell. Info. Systems*, 41(3):371–406, 2013.

[21] B. L. Sturm. Making explicit the formalism underlying evaluation in music information retrieval research: A look at the MIREX automatic mood classification task. In *Post-proc. Computer Music Modeling and Research*, 2014.

[22] B. L. Sturm. The state of the art ten years after a state of the art: Future research in music information retrieval. *J. New Music Research*, 43(2):147–172, 2014.

[23] B. L. Sturm. A simple method to determine if a music information retrieval system is a "horse". *IEEE Trans. Multimedia*, 2014 (in press).

[24] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet. Semantic annotation and retrieval of music and sound effects. *IEEE Trans. Audio, Speech, Lang. Process.*, 16, 2008.

[25] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Trans. Speech Audio Process.*, 10(5):293–302, July 2002.

[26] J. Urbano. *Evaluation in Audio Music Similarity*. PhD thesis, University Carlos III of Madrid, 2013.

[27] J. Urbano, M. Schedl, and X. Serra. Evaluation in music information retrieval. *J. Intell. Info. Systems*, 41(3):345–369, Dec. 2013.

[28] G. A. Wiggins. Semantic gap?? Schemantic schmap!! Methodological considerations in the scientific study of music. In *Proc. IEEE Int. Symp. Mulitmedia*, pages 477–482, Dec. 2009.

[29] Y.-H. Yang and H. H. Chen. *Music Emotion Recognition*. CRC Press, 2011.

AN ASSOCIATION-BASED APPROACH TO GENRE CLASSIFICATION IN MUSIC

Tom Arjannikov
University of Lethbridge
tom.arjannikov@uleth.ca

John Z. Zhang
University of Lethbridge
zhang@cs.uleth.ca

ABSTRACT

Music Information Retrieval (MIR) is a multi-disciplinary research area that aims to automate the access to large-volume music data, including browsing, retrieval, storage, etc. The work that we present in this paper tackles a non-trivial problem in the field, namely music genre classification, which is one of the core tasks in MIR. In our proposed approach, we make use of association analysis to study and predict music genres based on the acoustic features extracted directly from music. In essence, we build an associative classifier, which finds inherent associations between content-based features and individual genres and then uses them to predict the genre(s) of a new music piece. We demonstrate the feasibility of our approach through a series of experiments using two publicly available music datasets. One of them is the largest available in MIR and contains real world data, while the other has been widely used and provides a good benchmarking basis. We show the effectiveness of our approach and discuss various related issues. In addition, due to its associative nature, our classifier can assign multiple genres to a single music piece; hopefully this would offer insights into the prevalent multi-label situation in genre classification.

1. INTRODUCTION

The recent advances in technology, such as data storage and compression, data processing, information retrieval, and artificial intelligence, facilitate music recognition, music composition, music archiving, etc. The Internet is further promoting the enormous growth of digital music collections. Millions of songs previously in physical formats are now readily available through instant access, stimulating and motivating research efforts in meeting new challenges. Among them is *Music Information Retrieval (MIR)*, an interdisciplinary area that attracts practitioners from information retrieval, computer science, musicology, psychology, etc. One of the main tasks in MIR is the design and implementation of algorithmic approaches to managing large collections of digital music, including automatic tag annotation, recommendation, playlist generation, etc.

The work to be presented in this paper explores the feasibility of applying association analysis to music genre classification. Through our experience with music data, we have found that there are some inherent associations between audio characteristics and human assigned music genre labels. Accordingly, it would be desirable to see whether these associations, if found, can provide insight into genre classification of music. Our work in this paper is geared toward this target.

In a nutshell, our proposed approach uses music data itself by extracting useful information from it and conducting association analysis to make genre prediction. When we talk about the actual sound data of music, we refer to whatever is stored on various media, such as magnetic tapes and now in the digital format. We can extract useful information from this data via signal processing. This information represents the different characteristics of the actual sound stored on media [10]. We refer to it as *content-based features* and use it with our approach. To our knowledge, we are among the first to propose using association analysis for music genre classification in the MIR community.

2. PREVIOUS WORK

2.1 Classification in MIR

Classification is the process of organizing objects into predefined classes. It is a supervised type of learning, where we are given some labeled objects from which we form a computational model that can be used to classify new, previously unseen objects [15].

Classification is one of the core tasks in MIR, since it is usually the first step in many applications, such as on-line music retrieval, playlist recommendation, etc. In our work, we focus on genre classification, which is concerned with categorizing music audio into different genres. Tzanetakis and Cook [18] are among the first to work on this problem, where the task is to label an unknown piece of music with a correct genre name. They show that this is a difficult problem even for humans and report that college students achieve no more than 70% accuracy.

Previous works in MIR along this direction include the following. DeCoro *et al.* [5] use *Bayesian Model* to aid in hierarchical classification of music by aggregating the results of multiple independent classifiers and, thus, perform error correction and improve overall classification accu-

racy. Recent examples of using *Support Vector Machines (SVM)* for music genre classification include an investigation of Meng and Shawe-Taylor [13], where they explore different kernels used in a support vector classifier. Li and Sleep [9] extend normalized information distance into kernel distance for SVM and demonstrate classification accuracy comparable to others. In addition, recently, Anglade *et al.* [3] use *Decision Tree* for music genre classification by utilizing frequent chord sequences to induce context free definite clause grammars of music genres.

2.2 Association Analysis in MIR

Association analysis attempts to discover the inherent relations among data objects in an application domain. These relations are represented as association rules. An example of such application domain is the shopping basket analysis in supermarkets, where one tries to discover relations among the items purchased by customers. For example, the association rule $\{milk, eggs\} \rightarrow \{bread\}$ implies that, if *milk* and *eggs* are bought together by a customer, then *bread* is likely to be bought as well, i.e., they have some inherent statistical relationships [7].

We consider the so-called itemsets, such as $\{milk, eggs, bread\}$ in the above example, to be frequent if they appear in many transactions. The *support* of an itemset represents the percentage of transactions that contain the itemset and *minimum support* is the threshold that separates the frequent itemsets from the infrequent ones. A frequent itemset can produce an association rule of the form $A \rightarrow B$, where A and B are non-empty itemsets and $A \bigcap B = \phi$. An association rule holds for a dataset with some minimum support and *confidence*, which is the percentage of transactions containing A that also contain B [7].

A formal treatment of applying association analysis in MIR is in Section 3. Within the context of MIR, each track or music piece is represented using a set of content-based features derived from its digitized data. Together, a set of these features place the given track in a discrete location in the feature space. Intuitively, the tracks that are very similar to each other may share the same neighborhood. This could help with organizing music collections for effective data retrieval. When grouped together, the features contain some patterns. We would like to look for these patterns and use them for music genre classification.

Kuo *et al.* [8] propose a way to recommend music based on the emotion that it conveys and look for associations in data that contains information perceived only by humans. Similarly, Xiao *et al.* [19] use a parameterized statistical model to look for associations between timbre and perceived tempo. Liao *et al.* [12] use a dual-wing harmonium model to discover association patterns between MTV video clips and the music that accompanies those clips. Neubarth *et al.* [14] present a method of association rule mining with constraints and discover rules in the form of $A \rightarrow B$, telling that either region implies genre or genre implies region. Arjannikov *et al.* [4] use association analysis to verify tag annotation in music, though their approach is based on textual music tags and is not content-based. Our work to be presented below is different from the above and is among the initial efforts to apply association analysis to content-based music genre classification.

3. CLASSIFYING MUSIC INTO GENRES VIA ASSOCIATION ANALYSIS

Our work in this paper is focused on the music genre tags. As stated in [6, 10, 10], any discrete set of tags that are not correlated can be used as categories, or classes, into which we could split a collection of music pieces. Arjannikov *et al.* [4] show that association analysis reveals patterns in music textual tags. This motivates our investigation of association analysis in content-based music features.

3.1 Notation

Association analysis requires discrete items, however, most content-based music features are not. Thus, when given a set of features $F = \{f_1, f_2, f_3, \cdots, f_k\}$, we discretize each feature into a predetermined number of bins b, where $b > 1$, and derive a new feature set $F' = \{f'_{1_1}, f'_{1_2}, \cdots, f'_{1_b}, f'_{2_1}, f'_{2_2}, \cdots, f'_{2_b}, \cdots, f'_{k_1}, f'_{k_2}, \cdots, f'_{k_b}\}$. Then, from the set of music pieces M, we derive a transactional style dataset $D = \{d_1, d_2, \cdots, d_r\}$, where $r = |M|$. Each transaction $d_i = \{a_1, a_2, \cdots, a_k\}$ corresponds to a music piece and each a_j in d_i is a feature item in the literal form F_pB_q, where p corresponds to the feature number in F' and q corresponds to the bin number, into which the feature for the particular music piece falls. For example, if the first content-based feature is a number between 0 and 1, and it is discretized into 10 equidistant bins, then, given a particular music piece, whose first feature value is 0.125, its corresponding d_i would contain the label F_1B_2.

When we formulate our problem as described above, the music set M, becomes a transactional set D suitable for association mining.

3.2 Proposed Approach

We call our proposed approach *association-based music genre classifier (AMGC)*. Figure 1 depicts the whole process of using AMGC, which is detailed below.

3.2.1 AMGC

We start by preparing our data during the pre-processing stage. First, we acquire content-based features from music; in this paper, we use the features that have already been extracted and published for the purpose of comparing different classifiers on even ground [16, 17]. Then, we discretize any continuous features. It is worth noting that obtaining optimal discretization is an open problem in machine learning. In our work, we use feature discretization based on equal width of bins, for its simplicity, to avoid any possible bias based on class labels. Then we form transactional style datasets, as described in Section 3.1, and split the training dataset into subsets, one for each genre. Finally, we remove any items that appear in all transactions with a certain *frequency threshold* (FRQ), which is the percent of transactions containing the item.

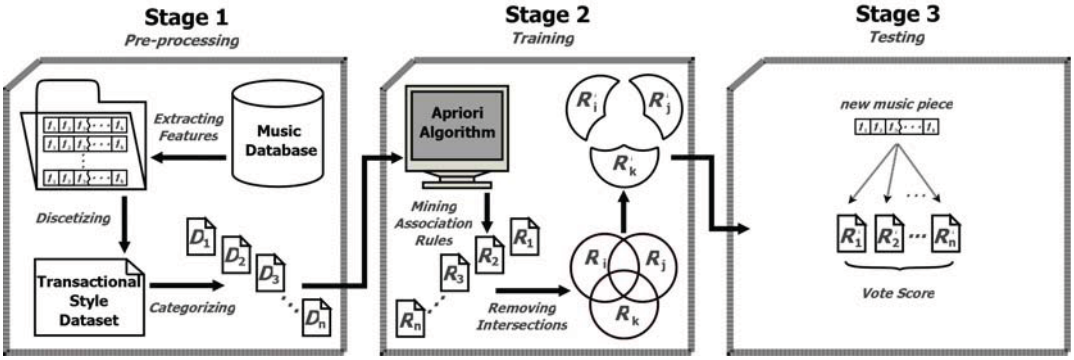


Figure 1. The three stages of our proposed association-based approach to classify music into genres.

During the training stage, we invoke the *Apriori* algorithm [1, 2] and mine frequent itemsets from each genre's sets of items at some minimum support. From these we generate classification rules of the form $A \rightarrow B$, where A is the frequent itemset and B is the genre associated with that itemset. Then, we remove any itemsets that appear in two or more genres. The resulting rules uniquely represent their respective genres and we use them for classification during the last stage.

3.2.2 Scoring Method

To obtain a classification score for each genre, we use the following four components. *Itemset Percentage* (IP) is the percent of itemsets that a given music piece matches for a given genre out of all itemsets matched from that genre. *Support Sum* (SS) is the sum of the matched itemsets' minimum support divided by the sum of all itemsets' minimum support for the given genre. *Confidence Sum* (CS) is the current genre's confidence sum of the matched itemsets divided by the sum of all itemsets' confidence. Finally, *Length Sum* (LS), the sum of cardinalities of the matched itemsets divided by the sum of cardinalities of all itemsets for the given genre.

We score each music piece against each genre's set of rules as following. First, we create a voting vector, whose cardinality is equal to the number of genres, and compute the corresponding component's value for each genre. Then, the genre with the highest value is voted as a candidate of that component, and its element in the voting vector is incremented by 1. Thus, the four components result in four votes and the genre with the highest number of votes is declared as winner and becomes the predicted genre of the given music piece.

3.2.3 Accuracy Evaluations

In our work, we use the following classification measures. *Recall*, also known as *sensitivity*, represents the percentage of correctly classified instances for that genre [7]. *Precision* reflects the percentage of correctly classified instances from all instances that are perceived as belonging to that genre by the classifier [7]. Finally, *accuracy* is calculated by dividing the number of all correctly classified instances for all genres by the total number of predictions made [7].

Because AMGC can assign multiple genre labels to a single music piece, we compute the *Multi-Labeling Rate* (MLR) by dividing the total number of predicted labels by the number of all test instances of a genre. MLR falls into the range between 1 and the total number of genres with frequent itemsets. The closer it is to 1, the fewer multi-label assignments were made, which indicates that AMGC is performing more like a single-label classifier. If MLR is equal to the total number of genres, then the results of classification are least useful. Furthermore, if MLR is below 1, then there are music pieces, whose genres could not be predicted.

3.3 Goals

Our aim is to test whether the classification rules obtained from music content-based features by AMGC can be used to categorize music into genres. For this, we designate three goals: (G_1) AMGC achieves a classification accuracy that is better than choosing genres at random; (G_2) AMGC is stable - when given similar datasets, it should achieve similar classification accuracy; (G_3) AMGC attains higher accuracy with better quality data and fewer genres.

4. EXPERIMENT RESULTS AND DISCUSSIONS

4.1 Data Preparation

The classification task at hand requires content-based features paired with genre tags and we find two datasets that fit this requirement.

The *Latin Music Database* [17], denoted as D_{LMD}, is popular in the music genre classification task despite its small size. There are many classification results available in the literature, which are based on a set of features that has already been extracted and circulated as part of D_{LMD}. Thus, we can test the feasibility of our approach without introducing variance based on difference in feature extraction techniques. Moreover, D_{LMD} usually results in high classification accuracy for many methods [17]. We use one of the three sets of features included with it, which is extracted from the beginning 30 seconds of each music piece.

The *Million Song Dataset Benchmarking* [16], denoted as D_{MSDB}, is much larger than D_{LMD} and boasts several sets of content-based features. We use five of these sets

and the genre labels, which were originally obtained from Allmusic [16]. Additionally, we restrict the number of tracks to 1000 per genre, in order to balance the number of training and testing examples among genres.

Dataset name	Number of songs	Number of genres	Number of features	Type of Features
D_{LMD}	3000	10	30	MFCC
$D_{\text{MSDB-1}}$	1500	15	10	MM
$D_{\text{MSDB-2}}$	1500	15	16	Spectral
$D_{\text{MSDB-3}}$	1500	15	20	LPC
$D_{\text{MSDB-4}}$	1500	15	20	AM
$D_{\text{MSDB-5}}$	1500	15	26	MFCC

Table 1. Music genre datasets and their statistics.

We include some statistical information about the datasets in Table 1 and label them accordingly. We split each one into two equal-sized partitions at random, while maintaining the genres balanced; each genre is represented by equal number of tracks in both partitions. One of the partitions becomes the training set and the other becomes the testing set. If there are too many music pieces belonging to one genre as compared to others, we remove the extra tracks at random. If a genre is represented by fewer pieces than 300 for D_{LMD} and 1000 for D_{MSDB}, then we do not use that genre in our experiments. This reduces the original D_{MSDB} dataset to 17 genres from 25. Moreover, during Stage 2 of our proposed approach, when we mine frequent itemsets, two of the genres produce none; therefore, only 15 genres persist, as reported in Table 1. D_{LMD} remains at 10 genres because it was originally balanced at 300 music pieces per genre.

In the following section, we demonstrate through our experiment results how we achieve the three goals formulated in Section 3.3.

4.2 Results and Discussions

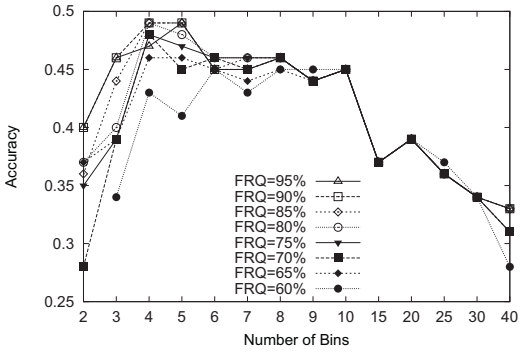


Figure 2. D_{LMD} at minimum support $= 20\%$.

During our experiments, we observe that our proposed parameters affect the classification accuracy, and thus, they are effective. It is evident from Figures 2 and 3 that the number of discretization bins affects the classification accuracy for both D_{LMD} and D_{MSDB}. Figure 4 demonstrates how the classification accuracy is affected by the minimum support parameter. We also note that AMGC performs much better than if we were to choose genres at random. Thus, we confirm that AMGC works for some parameter settings and conclude our work towards G_1.

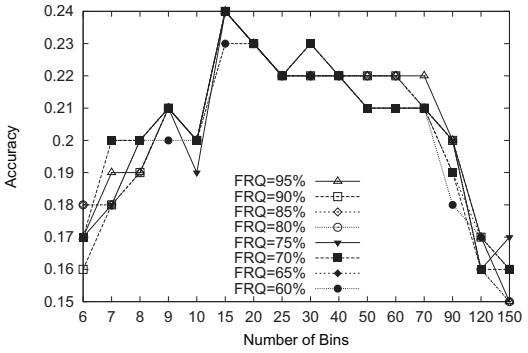


Figure 3. $D_{\text{MSDB-2}}$ at minimum support $= 2\%$.

As demonstrated in the literature, the classification accuracy usually increases when the number of classes is reduced [11]. Thus, we reduce the number of genres for both D_{LMD} and D_{MSDB} to 5 and observe that AMGC performs better. Therefore, we report only the results for the smaller set of genres in Figures 4 through 9. We also observe that D_{LMD} achieves higher accuracy than D_{MSDB} as can be seen in Figures 2 and 3. This concludes our work towards G_3, as AMGC performs better with a better quality dataset, moreover, it performs better on a reduced set of genres.

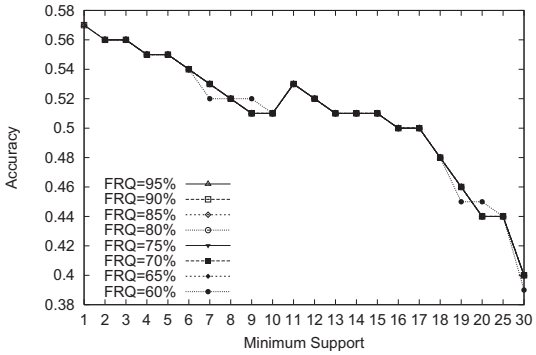


Figure 4. $D_{\text{MSDB-2}}$ with $number\ of\ bins = 20$.

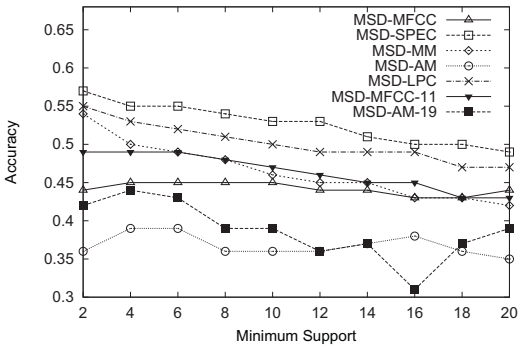


Figure 5. All five D_{MSDB} datasets compared, with $number\ of\ bins = 13$, unless otherwise specified.

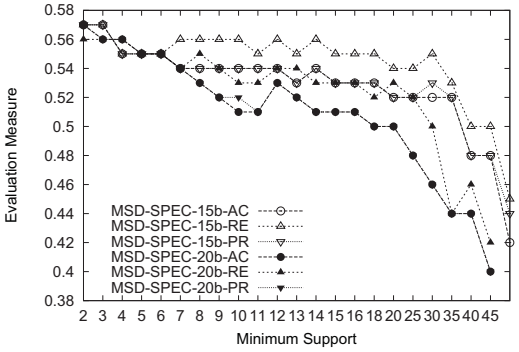


Figure 6. $D_{\text{MSDB-2}}$ across different minimum support.

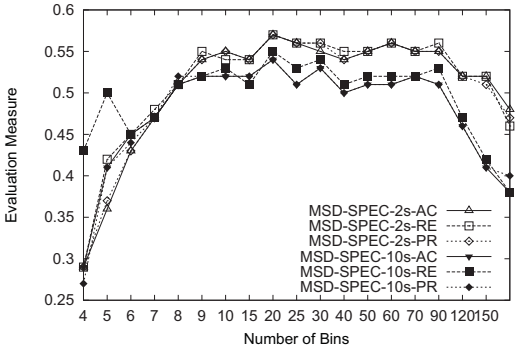


Figure 7. $D_{\text{MSDB-2}}$ across different number of bins.

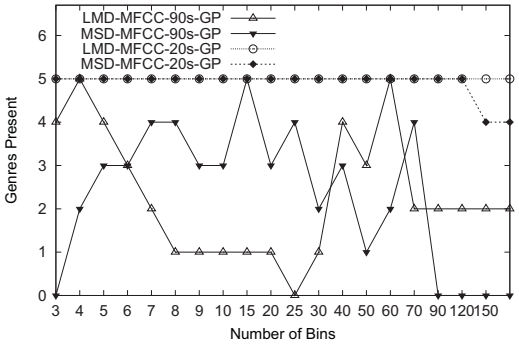


Figure 8. 5 genres of D_{LMD} and $D_{\text{MSDB-5}}$ at $FRQ = 95$.

It is clear from Figures 2, 3 and 4, that the FRQ parameter does not significantly affect the classification accuracy, although, it produces highest accuracy overall when set to 95%. We use this setting in all of the experiment results in Figures 5 through 9.

During our experiments, we observe that D_{MSDB} datasets perform best at lower minimum support and number of bins settings. We set the number of bins to 13 and perform a sweep across minimum support values between 2 and 20. As can be seen in Figure 5, among all five, $D_{\text{MSDB-2}}$ performs the best and $D_{\text{MSDB-4}}$ the worst. Three of the five datasets achieve their highest accuracy when the number of bins is set to 13; however, $D_{\text{MSDB-4}}$ performs better at 19 bins, and $D_{\text{MSDB-5}}$ at 11 bins. Thus, we include the corresponding results in Figure 5.

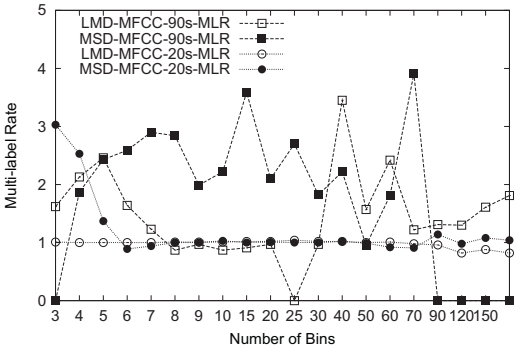


Figure 9. 5 genres of D_{LMD} and $D_{\text{MSDB-5}}$ at $FRQ = 95$.

We observe that all three evaluation measures, $recall$, $precision$, and $accuracy$, obtain very similar values to each other in our experiments, as can be seen in Figures 6 and 7. It can also be seen in Figures 2 through 7, that AMGC does not behave arbitrarily, when given different datasets or different parameter settings. This confirms that our approach is stable and concludes our work towards G_2.

During our experiments, we notice that for some values of minimum support and for some numbers of bins, AMGC performs much better than choosing genre assignment at random. However, with other values of these parameters, AMGC predicts majority of music to be of one genre. Moreover, sometimes it votes for all genres equally, where MLR becomes equal to the number of genres. Furthermore, we encountered certain parameter settings, when some or all genres were not represented by any classification rules. We investigate the behaviour of MLR and the number of genres present in both D_{LMD} and D_{MSDB} through further experiments and report our findings in Figures 8 and 9. Here, we set the minimum support to 20 and then to 90 for both datasets. As can be seen in Figure 8, at the higher minimum support, some genres are discarded, due to removal of intersections during Stage 2 of our approach. Meanwhile, Figure 9 illustrates that AMGC behaves as a single label classifier, because we remove rules that are found among any genre-pair, thus, the remaining rules are representative of a single genre.

When experimenting with our approach on music genre classification using different features in D_{MSDB}, we use the same genre assignment and alternate the features. This helps us confirm that difference in content-based features result in different classification performance. Hence, different features are more or less useful for the genre classification task, which is reflected by the *feature selection* task in MIR.

In our experiments, we notice that it may take a long time to pre-process the data and train the classifier. However, the resulting classification model is very fast, where its speed can be expressed as the number of classification rules multiplied by the number of music pieces to be classified.

5. CONCLUSION

In this paper, we introduce a novel approach to MIR, namely, using association analysis to help music genre classification. Association analysis looks for frequent patterns in music data, which represent the similarity of all music pieces in a given genre.

Through experiments, we demonstrate the effectiveness of our approach and confirm that association analysis can be applied to music data. However, there is still room for improvement, which includes feature extraction, feature selection and discretization. We believe that as they improve, our method will also improve. We can also take some immediate steps to improve our classifier by tuning the two parameters, minimum support for mining frequent items and the number of discretization bins. Our experiments demonstrate that these two parameters are directly related to the performance of our classifier, and they vary depending on the data. Hence, tuning these parameters to each specific dataset will improve the classification accuracy. We leave these to our future work.

6. REFERENCES

[1] R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, volume 22, pages 207–216. ACM, 1993.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, volume 1215, pages 487–499. Morgan Kaufmann Publishers Inc., 1994.

[3] A. Anglade, R. Ramirez, and S. Dixon. Genre classification using harmony rules induced from automatic chord transcriptions. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 669–674. ISMIR, 2009.

[4] T. Arjannikov, C. Sanden, and J. Z. Zhang. Verifying tag annotations through association analysis. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 195–200. ISMIR, 2013.

[5] C. DeCoro, Z. Barutcuoglu, and R. Fiebrink. Bayesian aggregation for hierarchical genre classification. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 77–80. ISMIR, 2007.

[6] Z. Fu, G. Lu, K. M. Ting, and D. Zhang. A survey of audio-based music classification and annotation. *IEEE Transactions on Multimedia*, 13(2):303–319, 2011.

[7] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., the second edition, 2006.

[8] F.-F. Kuo, M.-F. Chiang, M.-K. Shan, and S.-Y. Lee. Emotion-based music recommendation by association discovery from film music. In *Proceedings of the 13th Annual ACM International Conference on Multimedia*, pages 507–510. ACM, 2005.

[9] M. Li and R. Sleep. Genre classification via an lz78-based string kernel. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 252–259. ISMIR, 2005.

[10] T. Li, O. Mitsunori, and G. Tzanetakis, editors. *Music Data Mining*. CRC Press, 2012.

[11] T. Li, M. Ogihara, and Q. Li. A comparative study on content-based music genre classification. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, pages 282–289. ACM, 2003.

[12] C. Liao, P. Wang, and Y. Zhang. Mining association patterns between music and video clips in professional mtv. In B. Huet, A. Smeaton, K. Mayer-Patel, and Y. Avrithis, editors, *Advances in Multimedia Modelling*, volume 5371 of *Lecture Notes in Computer Science*, pages 401–412. Springer Berlin Heidelberg, 2009.

[13] A. Meng and J. Shawe-Taylor. An investigation of feature models for music genre classification using the support vector classifier. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 604–609. ISMIR, 2005.

[14] K. Neubarth, I. Goienetxea, C. Johnson, and D. Conklin. Association mining of folk music genres and toponyms. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 7–12. ISMIR, 2012.

[15] S. Russel and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson education Inc., the third edition, 2010.

[16] A. Schindler, R. Mayer, and A. Rauber. Facilitating comprehensive benchmarking experiments on the million song dataset. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 469–474. ISMIR, 2012.

[17] C. N. J. Silla, C. A. A. Kaestner, and A. L. Koerich. The latin music database. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 451–456. ISMIR, 2008.

[18] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, July 2002.

[19] L. Xiao, A. Tian, W. Li, and J. Zhou. Using a statistic model to capture the association between timber and perceived tempo. In *Proceedings of the International Society for Music Information Retrieval*, pages 659–662. ISMIR, 2008.

MULTIPLE VIEWPOINT MELODIC PREDICTION WITH FIXED-CONTEXT NEURAL NETWORKS

Srikanth Cherla[1,2], Tillman Weyde[1,2] and Artur d'Avila Garcez[2]
[1]Music Informatics Research Group, Department of Computer Science, City University London
[2] Machine Learning Group, Department of Computer Science, City University London
{srikanth.cherla.1, t.e.weyde, a.garcez}@city.ac.uk

ABSTRACT

The *multiple viewpoints* representation is an event-based representation of symbolic music data which offers a means for the analysis and generation of notated music. Previous work using this representation has predominantly relied on n-gram and variable order Markov models for music sequence modelling. Recently the efficacy of a class of distributed models, namely restricted Boltzmann machines, was demonstrated for this purpose. In this paper, we demonstrate the use of two neural network models which use fixed-length sequences of various viewpoint types as input to predict the pitch of the next note in the sequence. The predictive performance of each of these models is comparable to that of models previously evaluated on the same task. We then combine the predictions of individual models using an entropy-weighted combination scheme to improve the overall prediction performance, and compare this with the predictions of a single equivalent model which takes as input all the viewpoint types of each of the individual models in the combination.

1. INTRODUCTION

We are interested in the computational modelling of melodies available in symbolic music data formats such as MIDI and KERN. For this purpose, we chose to work with a representation of symbolic music first proposed in [9] in relation to *multiple viewpoints for music prediction* (which we refer to here as the "multiple viewpoints representation"). The multiple viewpoints representation is an event-based representation extracted from symbolic music data where a given piece of music is decomposed into parallel streams of features, known as *viewpoint types*. Each viewpoint type is either a directly observable musical dimension such as *pitch* and *note duration*, or an abstract one derived from them such as *inter-onset interval* or *pitch contour*. In order to analyse musical structure using this representation, one can train a machine learning model on sequences of viewpoint types and apply it to tasks such as music generation [6] and classification [3, 7]. This representation has also been the focus of more recent work related to music cognition [14,17]. The novelty of this approach is in its extension of previous work in language modelling to music with an information theoretic backing which facilitates an objective evaluation of models for music prediction. Approaches based on information theory have been of interest in musicology to understand structure and meaning in music in terms of its predictability [10, 11, 13].

In the original work on multiple viewpoints [9] and that which followed [15, 21], Markov models were exclusively employed for music modelling using this framework. While this is a reasonable choice, Markov models are often faced with a problem related to data sparsity known as the *curse of dimensionality* [2]. This refers to the exponential rise in the number of model parameters to be estimated with the length of the modelled sequences. Models which employ distributed architectures such as neural networks tend to avoid this problem, as they do not require enumerating all state transition probabilities, but rather the weights of the network encode only those dependencies necessary to minimize prediction error. It was demonstrated more recently in [4] how a distributed model — the restricted Boltzmann machine, is a suitable alternative in this context. It was also suggested in [8] that neural networks might be suitable alternatives to n-gram models for music modelling with multiple viewpoints but no actual research in this direction has ensued.

In this paper, we first present two neural networks for modelling sequences of musical pitch. The first is a simple feed-forward neural network [20], and the second is the musical extension of the Neural Probabilistic Language Model [2] — a deeper feed-forward network with an added weight-sharing layer between the input and hidden layers. The latter was originally proposed for learning distributed representations of words in language modelling. Both models predict a probability distribution over the possible values of the next pitch given a fixed-length context as input. Their predictive performance is comparable to or better than previously evaluated melody prediction models in [4, 16]. The second network is further extended to make use of additional viewpoint types extracted from the context, as inputs for the same task of predicting musical pitch. We then combine the predictions of individual models with different viewpoint types as their respective

inputs using an entropy-weighted combination scheme to improve the overall prediction performance, and compare this with the predictions of a single model which takes as input all the viewpoint types of each of the individual models in the combination.

We begin with an overview of the multiple viewpoints representation in Section 2. This is followed by a description of the two neural networks which are used with this representation, in Section 3. Section 4 presents an evaluation of the predictive performance of the two models along with a comparison to previous work. Finally, directions for future research are outlined in Section 5.

2. MULTIPLE VIEWPOINT SYSTEMS

In order to explain music prediction with multiple viewpoints, the analogy to natural language is used here. In statistical language modelling, the goal is to build a model that can estimate the joint probability distribution of subsequences of words occurring in a language L. A statistical language model (SLM) can be represented by the conditional probability of the next word w_T given all the previous ones $[w_1, \ldots, w_{(T-1)}]$ (written here as $w_1^{(T-1)}$), as

$$P(w_1^T) = \prod_{t=1}^{T} P(w_t | w_1^{(t-1)}) \; . \tag{1}$$

The most commonly used SLMs are *n-gram* models, which rely on the simplifying assumption that the probability of a word in a sequence depends only on the immediately preceding $(n-1)$ words [12]. This is known as the Markov assumption, and reduces (1) to

$$P(w_1^T) = \prod_{t=1}^{T} P(w_t | w_{(t-n+1)}^{(t-1)}) \; . \tag{2}$$

Following this approach, musical styles can be interpreted as vast and complex languages [9]. In predicting music, one is interested in learning the joint distribution of *musical event* sequences s_1^T in a *musical language* S. Much in the same way as an SLM, a system for music prediction models the conditional distribution $p(s_t | s_1^{(t-1)})$, or under the Markov assumption $p(s_t | s_{(t-n+1)}^{(t-1)})$. For each prediction, context information is obtained from the events $s_{(t-n+1)}^{(t-1)}$ immediately preceding s_t. Musical events have a rich internal structure and can be expressed in terms of directly observable or derived musical features such as pitch, note duration, inter-onset interval, or a combination of two or more such features. The framework of multiple viewpoint systems for music prediction [9] was proposed in order to efficiently handle this rich internal structure of music by exploiting information contained in these different musical feature sequences, while at the same time limiting the dimensionality of the models using these features. In the interest of brevity, we limit ourselves to an informal discussion of multiple viewpoint systems for monophonic music prediction and refer the reader to [9] for a more detailed explanation.

A musical event s refers to the occurrence of a note in a melody. A *viewpoint type* (or simply *type*) τ refers to any of a set of musical features that describe an event. The domain of a *type*, denoted by $[\tau]$ is the set of possible values of that type. A *basic type* is a directly observable or given feature such as *pitch, note duration, key-signature* or *time-signature*. A *derived type* can be derived from any of the basic types or other derived types. Two or more types can be "linked" by taking the Cartesian product of their respective domains, thus creating a *linked viewpoint type*. A *multiple viewpoints system* (MVS) is a set of models, each of which is trained on subsequences of one *type*, whose individual predictions are combined in some way to influence the prediction of the next event in a given event sequence. Given a context $s_{(t-n+1)}^{(t-1)}$ and an event s_t, each viewpoint τ in an MVS must compute the probability $p_\tau(s_t | s_{(t-n+1)}^{(t-1)})$.

In order to input the viewpoint type sequences to the neural network models, we first convert each input type value into a binary one-hot encoding. When a context event is missing or undefined, each element of the vector is initialized to $1/|S|$. When there is more than one input type, one-hot vectors corresponding to all the input types for a musical event are concatenated to obtain an input vector for that event. As we are dealing with models of fixed context-length l, the final input feature vector input to the model is a concatenation of l such vectors. In doing so, we are effectively bypassing the need to compute a Cartesian product to link viewpoint types before using them as input to a single model which has been the practice when using n-gram and variable order Markov models.

Each model in an MVS relies on a different source of information (its respective input types) to make a prediction about the target viewpoint type. The accuracy of the prediction depends on how informative these input types are of the target type. It is possible to combine the information provided by different input types for possibly better predictive performance. Here, we consider two ways of doing this - *implicitly* in a single model which is trained using a set of input types, and *explicitly* by combining the probability distributions of multiple models, each of which is trained separately on a mutually exclusive subset of these input types. While the former is only a special case of what has been described so far, we provide an explanation of the latter below in Section 2.1.

2.1 Combining Multiple Models

It was demonstrated in [9, 15] that an entropy-weighted combination of the predictions of two or more n-gram or variable order Markov models typically results in ensembles with better predictive performance than any of the individual models. As it is the predicted distributions which are combined, this approach is independent of the types of models involved. Here, we briefly describe two approaches for creating such ensembles. Let M be a set of models and $p_m(s)$ be the probability assigned to symbol $s \in [\tau_{tgt}]$ by model m, where $[\tau_{tgt}]$ is the domain of the target type. The first approach involves taking a weighted arithmetic mean of their respective predictions. This is the *mixture-*

of-experts combination, and is defined as

$$p(s) = \frac{\sum_{m \in M} w_m p_m(s)}{\sum_{m \in M} w_m}$$

where each of the weights w_m depends on the entropy of the distribution generated by the corresponding model m in the combination such that greater entropy (and hence uncertainty) is associated with a lower weight [5]. The weights are given by the expression $w_m = H_{rel}(p_m)^{-b}$, where the relative entropy $H_{rel}(p_m)$ is

$$H_{rel}(p_m) = \begin{cases} H(p_m)/H_{max}(p_m), & \text{if } H_{max}([\tau_{tgt}]) > 0 \\ 1, & \text{otherwise} \end{cases}$$

The best value of the bias b is determined through cross-validation. The quantities H and H_{max} are respectively the entropy of the prediction and the maximum entropy of predictions over the symbol space $[\tau_{tgt}]$, and are defined as

$$H(p) = -\sum_{s \in [\tau_{tgt}]} p(s) \log_2 p(s) \; . \tag{3}$$

$$H_{max}(p) = \log_2 |S|.$$

where $p(s \in [\tau_{tgt}]) = p(\chi = s)$ is the probability mass function of a random variable χ distributed over the discrete alphabet $[\tau_{tgt}]$ such that the individual probabilities are independent and sum to 1.

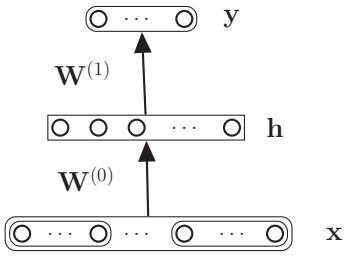
The second combination method — *product-of-experts*, is computed similarly as the weighted geometric mean of the probability distributions. This is given by

$$p(s) = \frac{1}{R} \left(\prod_{m \in M} p_m(s)^{w_m} \right)^{\frac{1}{\sum_{m \in M} w_m}}$$

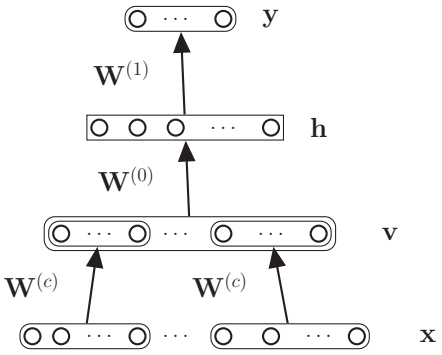
where R is a normalisation constant which ensures that the resulting distribution over S sums to unity. The weights w_m in this case are obtained in the same manner as for the mixture-of-experts case. It was observed in a previous application of these two combination methods to melody modelling [15], that product-of-experts resulted in a greater improvement in predictive performance.

3. FIXED-CONTEXT NEURAL NETWORKS

In this section, we provide a brief overview of the two fixed-context neural network models which we employed for the task of predicting the pitch of the next note in a melody, given a viewpoint type context which leads up to it. These are (1) a feed-forward neural network, and (2) a neural probabilistic melody model. The key difference between the two is the presence of an additional weight-sharing layer in the latter which transforms the binary representation of the viewpoint types into lower-dimensional real-valued vectors before passing these on as inputs to a feed-forward network (much like the former).



(a) Feed-forward Neural Network



(b) Neural Probabilistic Melody Model

Figure 1: The two models employed for multiple viewpoint melodic prediction in this paper (biases ignored in the illustration). A concatenation of the fixed-length input type context is presented to each model in its visible layer and the predictions are made in the output layer.

3.1 Feed-forward Neural Network

In its simplest form, a feed-forward neural network (Figure 1) consists of an input layer $\mathbf{x} \in \mathbb{R}^n$, a hidden layer $\mathbf{h} \in \mathbb{R}^m$ and an output layer $\mathbf{y} \in \mathbb{R}^l$. The input layer is connected to the hidden layer by a weight-matrix $W^{(0)}$ and likewise, the hidden layer to the output layer by a matrix $W^{(1)}$. Each unit in the hidden layer typically applies a non-linear function to the input it receives from the layer below it. Similarly, each unit of the output layer applies a function to the input it receives from the hidden layer immediately preceding it. In a network with a single hidden layer, this happens according to the following equations

$$\mathbf{u}^{(0)} = \mathbf{b}^{(0)} + W^{(0)}\mathbf{x} \tag{4}$$
$$\mathbf{h} = f^{(0)}(\mathbf{u}) \tag{5}$$
$$\mathbf{u}^{(1)} = \mathbf{b}^{(1)} + W^{(1)}\mathbf{h} \tag{6}$$
$$\mathbf{y} = f^{(1)}(\mathbf{v}) \tag{7}$$

where $\mathbf{b}^{(0)}$ and $\mathbf{b}^{(1)}$ are the hidden and output layer biases, $f^{(0)}$ and $f^{(1)}$ are functions applied to the input received by each node in the hidden and output layers respectively. Thus, for a given input $\mathbf{x}$, the output $\mathbf{y}$ is calculated as

$$\mathbf{y} = f^{(1)}(\mathbf{b}^{(1)} + W^{(1)} \cdot f^{(0)}(\mathbf{b}^{(0)} + W^{(0)}\mathbf{x})) \tag{8}$$

In the present case, $f^{(0)}$ is the logistic sigmoid function and $f^{(1)}$ is the softmax function. The network can

be trained in a supervised manner using the backpropagation algorithm [20]. This algorithm applies the chain rule of differentiation to propagate the error between the target output and the output of the model backwards into the network, and use these derivatives to appropriately update the model parameters (the network weights and biases).

3.2 Neural Probabilistic Melody Model

Next we consider the neural probabilistic melody model (NPMM), which was originally introduced in [2] as a language model for word sequences. It consists of a feed-forward network such as the one described in Section 3.1, with an additional *embedding* layer below it (Figure 1). This model takes as input a concatenation of binary viewpoint type vectors (*cf.* Section 3) which represent a fixed-length context. The first layer of the network maps each of these sparse binary vectors to lower-dimensional dense real-valued vectors which make up the input layer of what is essentially a feed-forward network above it. This mapping is determined by a shared weight matrix $W^{(c)}$ which is learned from data, and is given by

$$\mathbf{v} = W^{(c)}\mathbf{x}. \tag{9}$$

The hidden layer in the case of the NPMM consists of hyperbolic-tangent activation units. The output layer contains softmax units. The model is trained with backpropagation using gradient descent as in the case of a standard feed-forward neural network.

4. EVALUATION

The first goal of this paper is to demonstrate the suitability of fixed-context neural networks for multiple viewpoint melodic prediction. To this end, we compare the two models described in Section 3 with variable-order Markov models (VOMMs) and restricted Boltzmann machines (RBMs). It was observed that the predictive performance of each of the neural network models is either comparable to or better than that of the best VOMMs of both bounded and unbounded order [16], while slightly worse than the RBM of [4] (Figure 2). Second, we wish to compare the predictions of a single neural network which uses multiple input types with that of an ensemble of networks with smaller input dimensions, each of which uses a subset of the input types of the former, and combined with the entropy-based weighting scheme described in 2.1. We found that, while the addition of viewpoint types does improve the predictive performance in both cases, that of the single network is slightly worse than the ensemble (Figure 3). Moreover, the extent of this improvement diminishes with an increase in context length.

4.1 Dataset

Evaluation was carried out on a corpus of monophonic MIDI melodies that cover a range of musical styles. It consists of 4 datasets - Bach chorale melodies, and folk melodies from Canada, China and Germany, with a total of 37,229 musical events. These were also used to evaluate RBMs and variable order Markov models for music prediction in [4, 16]. To facilitate a direct comparison, the melodies are not transposed to a default key.

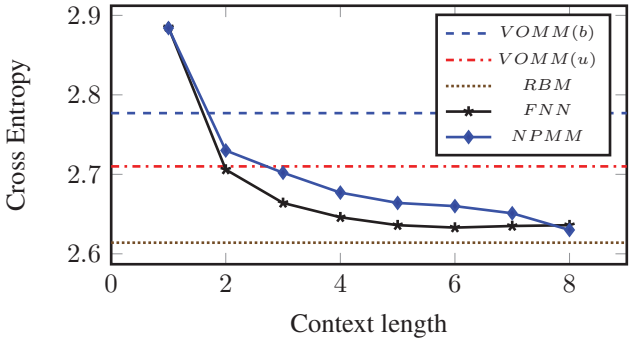


Figure 2: Comparison between the predictive performances of the best bounded and unbounded variable-order Markov models (VOMM(b) and VOMM(u) respectively), the best restricted Boltzmann machine (RBM), the feed-forward neural network (FNN) and the neural probabilistic melody model (NPMM) averaged over the datasets.

4.2 Evaluation Measure

In order to evaluate the proposed prediction models, we turn to a previous study of Markov models for music prediction in [16]. There, *cross entropy* was used to measure the information content of the models. This is a quantity related to *entropy* (3). The value of entropy, with reference to a prediction model, is a measure of the uncertainty of its predictions. A higher value reflects greater uncertainty. In practice, one rarely knows the true probability distribution of the stochastic process and uses a model to approximate the probabilities in (3). An estimate of the goodness of this approximation can be measured using cross entropy (H_c) which represents the divergence between the entropy calculated from the estimated probabilities and the source model. This quantity can be computed over all the subsequences of length n in the test data $\mathcal{D}_{test}$, as

$$H_c(p_{mod}, \mathcal{D}_{test}) = \frac{-\sum_{s_1^n \in \mathcal{D}_{test}} \log_2 p_{mod}(s_n | s_1^{(n-1)})}{|\mathcal{D}_{test}|} \tag{10}$$

where p_{mod} is the probability assigned by the model to the last pitch in the subsequence given its preceding context. Cross-entropy approaches the true entropy as the number of test samples ($|\mathcal{D}_{test}|$) increases.

4.3 Model Selection

Different neural network configurations were evaluated by a grid search over the learning rate $\eta = \{0.05, 0.1\}$, the number of hidden units $n_{hid} = \{25, 50, 100, 200, 400\}$, number of embedding units $n_{emb} = \{10, 20\}$ (only for the NPMM), and weight decay $w_{decay} = \{0.0000, 0.0001, 0.0005\}$. Each model was trained using mini-batch gradient descent up to a maximum of 1000 epochs with a batch size of 100 samples. Early-stopping [19] and weight-decay

were also incorporated to counter overfitting. The momentum parameter μ, was set to 0.5 during the first five epochs and then increased to 0.9 for the rest of the training. Each model was evaluated with 10-fold cross-validation, with folds identical to those used in [4, 16] for the sake of comparison.

4.4 Model Comparison

We carried out a comparison between the predictive performance of the two neural network models presented here, and models previously evaluated on the same datasets [4, 16]. It is to be noted that, since neither of our models is updated online during prediction, the comparison with the variable order Markov models of [16] is limited to their best performing Long-Term Models. These are of order bound 2 and unbounded order (labelled there as *C*I*). It is evident from Figure 2 that both the neural network models are able to take advantage of information in longer contexts than the bounded order n-gram models. This is also a feature of the RBM, whose best case of context-length 5 outperforms the rest of the models in the plot. The slight deterioration in the performance of the feed-forward network for longer contexts is possibly due to poor optimization of its parameters. This is considering the fact that weight-decay and early-stopping were implemented in the training algorithm to prevent overfitting. While it was not possible to incorporate further steps for better parameter optimization in this paper, the results are still illustrative of the networks' suitability at the given task and the improvement in performance with context consistent with each other and with that of the RBMs. Possible optimizations have been left as future work, and will be discussed in Section 5.

4.5 Model Combination

In order to evaluate the combination of viewpoint types, we selected one type which is related to the "what" in music — scale-degree ($intfref$), and another which is related to the "when" — inter-onset interval (ioi), from the several possible choices that exist. Furthermore, this experiment was performed using the NPMM and only on the Chinese folk melody dataset for the purpose of illustration, with the assumption that a similar trend would be observed with the other model and datasets. As our target viewpoint type i.e. the one being predicted, is musical pitch ($seqpitch$), the first model has the input types $seqpitch$ and $intfref$ and the second one $seqpitch$ and ioi. The additional viewpoints are incorporated as explained in Section 2. The predictions of these two models are combined *explicitly* using the mixture- and product-of-experts schemes. On the other hand, the *implicit* combination of these two is a single model whose input types are $seqpitch$, $intfref$ and ioi. Figure 3 compares the predictions of the pitch-only version of the NPMM and the three models using the additional input types. It can be seen that each of these three models has a better predictive performance than its pitch-only counterpart, thus confirming the relevance of the added viewpoint types to musical pitch prediction. Both the mixture- and product-of-experts combination schemes ($seqpitch \times (intfref +_m ioi)$ and $seqpitch \times (intfref +_p ioi)$ respectively in the plot) result in very similar predictive performance, with the latter working only slightly better for shorter context-lengths of 1, 2 and 3. Moreover, both these explicit combinations of viewpoint types perform better than the single implicit combination of types ($seqpitch \times intfref \times ioi$ in the plot). One will, however, notice that the cross entropy of the predictions slightly worsens at longer context-lengths, and that the discrepancy between the implicit and explicit combinations gradually increases in these cases. As mentioned earlier, we attribute this to the optimization of the network parameters, which is to be dealt with in future work.

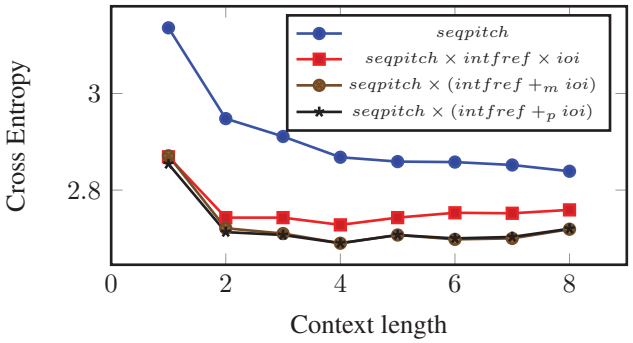


Figure 3: Comparison between the predictive performances, on the Chinese folk melody dataset, of the pitch-only NPMM, its extension which uses the *intfref* and *ioi* types as additional input, and ensembles each of which combines two models of input types (a) *seqpitch* and *intfref* (b) *seqpitch* and *ioi* using the mixture ($+_m$) and product ($+_p$) combination schemes.

5. CONCLUSIONS & FUTURE WORK

The two neural network models for melodic prediction presented here have been found to have a predictive performance comparable to or better than previously evaluated VOMMs, but slightly worse than that of RBMs. Predictive performance can be further improved by the addition of viewpoint types to the same model, or by combining multiple models using an entropy-weighted combination scheme. In our experiments, the latter tended to be better.

One open issue that remains is the parameter optimization in the two networks presented here. It was observed that, particularly when the input layer of a network is large and the dataset relatively small, the predictive performance does not improve as expected with context-length and the addition of viewpoint types. We note here that the results presented have been generated with models implemented in-house [1] for use with the Python machine learning library *scikit-learn* [18], and were thus limited in the various initialization and optimization strategies used in their learning algorithms. We also suspect this to be the reason for the limited success of the NPMM which exhibited relatively more promising results in its language

[1] Code available upon request.

modelling application in [2]. Many more measures to improve generalization and overall prediction accuracy (such as dropout, different weights initialization strategies and layer-wise pre-training) have been suggested in [1]. Incorporating these measures (or using an existing neural network library which does) can further improve the results.

Apart from this, there are three other aspects which are of immediate interest to us. The first is the incorporation of a short-term element in the prediction model which updates its parameters as data is presented to it, and has been shown to result in improved prediction performance and human-like predictions [15]. Secondly, while the number of parameters of the fixed-context models presented here increases linearly with the context-length (assuming a fixed number of hidden units), we are at present experimenting with recurrent networks where this problem does not arise due to their recurrent connections. And finally, the extension of the said models to polyphonic multiple viewpoints representations is also an open issue at the moment which we hope to address in the future.

6. ACKNOWLEDGEMENTS

Srikanth Cherla is supported by a PhD studentship from City University London. The authors would like to thank Marcus Pearce for his valuable advice, and the anonymous reviewers for their feedback on the submission.

7. REFERENCES

[1] Yoshua Bengio. Practical Recommendations for Gradient-Based Training of Deep Architectures. In *Neural Networks: Tricks of the Trade*, pages 437–478. 2012.

[2] Yoshua Bengio, Rejean Ducharme, Pascal Vincent, and Christian Jauvin. A Neural Probabilistic Language Model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

[3] Srikanth Cherla, Artur d'Avila Garcez, and Tillman Weyde. A neural probabilistic model for predicting melodic sequences. In *International Workshop on Machine Learning and Music*, 2013.

[4] Srikanth Cherla, Tillman Weyde, Artur d'Avila Garcez, and Marcus Pearce. A distributed model for multiple viewpoint melodic prediction. In *International Society for Music Information Retrieval Conference*, pages 15–20, 2013.

[5] Darrell Conklin. Prediction and entropy of music. 1990.

[6] Darrell Conklin. Music generation from statistical models. In *AISB Symposium on Artificial Intelligence and Creativity in the Arts and Sciences*, pages 30–35, 2003.

[7] Darrell Conklin. Multiple viewpoint systems for music classification. *Journal of New Music Research*, 42(1):19–26, 2013.

[8] Darrell Conklin and John G Cleary. Modelling and generating music using multiple viewpoints. 1988.

[9] Darrell Conklin and Ian H Witten. Multiple viewpoint systems for music prediction. *Journal of New Music Research*, 24(1):51–73, 1995.

[10] Greg Cox. On the relationship between entropy and meaning in music: An exploration with recurrent neural networks. *Proceedings of the 32nd Annual Cognitive Science Society. Austin TX: CSS*, 2010.

[11] David Brian Huron. *Sweet anticipation: Music and the psychology of expectation*. MIT press, 2006.

[12] Christopher D Manning and Hinrich Schütze. *Foundations of statistical natural language processing*. MIT press, 1999.

[13] Leonard B Meyer. Meaning in music and information theory. *The Journal of Aesthetics and Art Criticism*, 15(4):412–424, 1957.

[14] Diana Omigie, Marcus T Pearce, Victoria J Williamson, and Lauren Stewart. Electrophysiological correlates of melodic processing in congenital amusia. *Neuropsychologia*, 2013.

[15] Marcus T Pearce. *The construction and evaluation of statistical models of melodic structure in music perception and composition*. PhD thesis, City University London, 2005.

[16] Marcus T Pearce and Geraint A Wiggins. Improved methods for statistical modelling of monophonic music. *Journal of New Music Research*, 33(4):367–385, 2004.

[17] Marcus T Pearce and Geraint A Wiggins. Expectation in melody: The influence of context and learning. *Music Perception*, 23(5):377–405, June 2006.

[18] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research*, 12:2825–2830, 2011.

[19] Lutz Prechelt. Early Stopping But When? In *Neural Networks: Tricks of the Trade*, pages 55–69. 2012.

[20] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 1988.

[21] Raymond P Whorley. *The Construction and Evaluation of Statistical Models of Melody and Harmony*. PhD thesis, 2013.

VEROVIO: A LIBRARY FOR ENGRAVING MEI MUSIC NOTATION INTO SVG

Laurent Pugin
Swiss RISM Office
laurent.pugin@rism-ch.org

Rodolfo Zitellini
Swiss RISM Office
rodolfo.zitellini@rism-ch.org

Perry Roland
University of Virginia
pdr4h@virginia.edu

ABSTRACT

Rendering symbolic music notation is a common component of many MIR applications, and many tools are available for this task. There is, however, a need for a tool that can natively render the Music Encoding Initiative (MEI) notation encodings that are increasingly used in music research projects. In this paper, we present Verovio, a library and toolkit for rendering MEI. A significant advantage of Verovio is that it implements MEI's structure internally, making it the best suited solution for rendering features that make MEI unique. Verovio is designed as a fast, portable, lightweight tool written in pure standard C++ with no dependencies on third-party frameworks or libraries. It can be used as a command-line rendering tool, as a library, or it can be compiled to JavaScript using the Emscripten LLVM-to-JavaScript compiler. This last option is particularly interesting because it provides a complete in-browser music MEI typesetter. The SVG output from Verovio is organized in such a way that the MEI structure is preserved as much as possible. Since every graphic in SVG is an XML element that is easily addressable, Verovio is particularly well-suited for interactive applications, especially in web browsers. Verovio is available under the GPL open-source license.

1. INTRODUCTION

A few decades ago, rendering music notation by computer almost exclusively targeting printed output, most often in Postscript of PDF formats. Today, partly in response to the development of MIR applications, rendering of music notation can be necessary in a wide range of contexts, for example within standalone desktop applications, in server-side web application scenarios, or directly in a web browser. For example, music notation might need to be rendered for displaying search results or for visualizing analysis outputs. Another example is score-following applications, where the passage currently played needs to be displayed and possibly highlighted. Rendering music notation by computer, however, is a complex task. Powerful music notation rendering engines exist in commercial and open-source notation editors, but these are usually not very modular and cannot easily be integrated within other applications. Other rendering engines, such as LilyPond [13] or Mup [1], can be used; however, they usually require the encoding to be converted to a particular typesetting input syntax. Their architectures and dependencies also often limit the contexts in which their use is possible.

In recent years, the Music Encoding Initiative (MEI) has been increasingly adopted for music research projects [6]. Its large scope (MEI can be used to encode a wide range of music notations, from medieval neumes to common Western music notation), modularity, rich metadata header and numerous other features, including alignment with audio files or performance annotations, make it appropriate for a wide range of MIR applications. Unfortunately, most of the solutions currently available for rendering MEI involve a conversion to another format, either explicitly or internally in the software application used for rendering.

In this paper, we present the Verovio project, a library and toolkit for rendering MEI natively in SVG. Its purpose is to provide a self-contained typesetting engine that is capable of creating high-quality graphical output and that can also be used in different application contexts. In the following section, we describe previous work and existing solutions for rendering MEI and the use of SVG for music notation. We then introduce Verovio, describe the MEI structure on which it is built, outline its programming architecture, and highlight features currently available. We then present possible uses and output examples and conclude the paper with the future work that is planned for Verovio.

2. PREVIOUS WORK

One currently available option for rendering MEI is conversion to another format in order to use existing tools that do not support MEI. For software applications or rendering engines that support the import of the MusicXML interchange format, MEI can be converted with the mei2musicxml XSL stylesheet [9]. Another option is to convert MEI directly to a typesetting format, such as Mup. Mup is a C rendering engine that was made open-source in 2012. It uses its own typesetting syntax and produces high quality Postscript output. The conversion of MEI to Mup can be achieved in one step using the mei2mup XSL stylesheet [8]. A similar approach is pos-

sible for rendering MEI in a web browser, using a conversion to the ABC format. ABC is an encoding format primarily targeting material with fairly limited notational features, such as folk and traditional melodies. It can be rendered in a web browser with the abcjs renderer [15], and the conversion from MEI to ABC can be achieved with the mei2abc converter [5]. There is also a new JavaScript library, MEItoVexflow [18], that makes it possible to render MEI directly in web browsers using the Vexflow API [12]. Another tool for rendition of MEI online is Neon.js [3]. The tool not only renders, but also acts as a full online editor for neumatic medieval notation.

SVG for music notation has been used in several projects. One early attempt was made in 2003 for converting MusicXML to SVG using XSLT [14]. A framework with an editor was also developed for outputting SVG from GuidoXML notation as part of a dissertation thesis [2]. With MEI, SVG rendering was used for the first time in the DiMusEd project, a critical edition of songs of Hildegard von Bingen (1098-1179) [11]. In this web-based edition of neumatic notation, MEI rendering is performed on the server side with a custom rendering engine. There are also attempts to use XSLT to transform MEI to SVG directly in the browser. This approach is used in mono:di, the transcription software of the Corpus Monodicum editorial project sponsored by the Akademie der Wissenschaften und der Literatur in Mainz, also focused on medieval notation [4]. Finally, SVG is a possible back-end for the aforementioned Vexflow API in conjunction with the Raphael JavaScript library.

These solutions all have strengths and drawbacks in terms of compatibility, usability, speed, output quality, and music notation features available. Many of them, however, have limitations when the format to which MEI is converted for rendering does not support some features encoded in the MEI source or has a different structure, with the consequence that part of the encoding will be lost in conversion, or not rendered appropriately.

3. VEROVIO

3.1 MEI structure

The MEI schema provides multiple options for structuring the musical content. The most widely-used option is the score-based structure, where all the parts of a musical score are encoded together in the same XML sub-tree. The MEI schema also includes a part-based option, where each part is stored in a separate XML sub-tree. The choice between these options can depend not only on the type of document being encoded but also on the type of application. The Verovio library was designed as a direct implementation of the MEI structure. However, since it is rendering-focused, it is built around another content organization of MEI, a page-based customization more appropriate for graphical display. In a rendering task, the page (or more generically, the rendering surface) is a required high-level entity on which elements can be laid out by the rendering process. The page-based customization is a more fitting alternative data organization that provides a page top-level entity. It prioritizes the hierarchy that is treated as secondary when encoded with milestone elements `<pb>` in other MEI representations.

In the page-based customization, the content of the music is encoded in `<page>` elements that are themselves contained in a `<pages>` element within `<mdiv>` as shown in Figure 1. A `<page>` element contains `<system>` elements. From then on, the encoding is identical to standard MEI. That is, a `<system>` element will contain `<measure>` elements or `<staff>` elements that are both un-customized, depending on whether or not the music is measured or un-measured, respectively. Since the modifications introduced by the customization are very limited, the Verovio library can be used to render un-customized MEI files. When loading un-customized MEI documents, some MEI elements are loaded by Verovio and converted to a page-based representation. Typically, `<pb>` milestone elements are converted to `<page>` container elements. Conversely, `<section>` container elements are converted to `<secb>` milestone elements.

```
<body>
  <mdiv>
    <pages>
      <page page.width="2108" page.height="2970" page.leftmar="20">
        <system system.leftmar="50" system.rightmar="50">
          <scoreDef>
            <staffGrp symbol="line">
              <staffDef n="1" clef.line="2" clef.shape="G" />
            </staffGrp>
          </scoreDef>
          <measure n="1">
            <staff n="1">
              <layer n="1">
                <!-- ... -->
              </layer>
            </staff>
          </measure>
        </system>
      </page>
    </pages>
  </mdiv>
</body>
```

Figure 1. The page-based MEI structure used by Verovio. The `<mdiv>` element contains `<pages>`, `<page>` and `<system>` elements.

3.1.1 Layout and positioning

In addition to making rendering simpler and faster, the idea of the page-based customization is also to make it possible to encode the positioning of elements directly in the content tree without having to refer to the facsimile sub-tree. The latter traditional approach remains available with the page-based customization for more detailed and more complex referencing to facsimile images. However, the page-based customization introduces a lightweight positioning and referencing system that can be useful when the encoding represents a single source with one

image per page. This is typically the case with optical music recognition applications for which the encoding of the position of each encoded element is necessary. Another possible use is the creation of overlay images to be displayed on top of facsimile images where the position of each symbol also needs to be encoded. Verovio supports both positioned elements and automatic layout. Automatic layout will be executed when un-customized MEI files are rendered.

3.1.2 Additional supported formats

In addition to MEI, Verovio can render Plain and Easy (PAE) code [7] and DARMS code [16]. PAE and DARMS encodings are widely used for encoding incipits, including those for the Répertoire International des Sources Musicales (RISM) project. In Verovio, these formats are converted to MEI internally, which means that the toolkit can also be used to convert them to MEI for purposes other than rendering.

3.2 SVG output

One significant advantage of SVG rendering over other formats (e.g., Postscript or PDF) is that it is rendered natively in most modern web browsers with no plug-in required. Because SVG is XML, it has an advantage over raster image formats that every graphical element is addressable, making it well-suited for interactive applications. In a web environment, this makes it easy to highlight notes or symbols, for example. In addition, since SVG is a vector format, the output can also be used for high-quality printing.

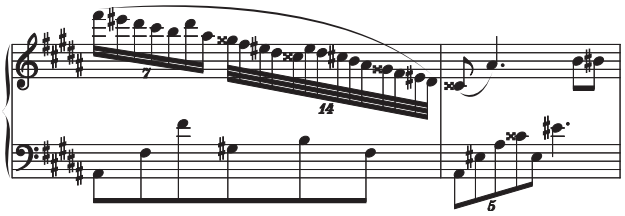
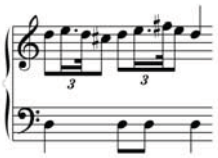


Figure 2. The output of Vervovio for two bars. The built-in layout engine of Verovio avoids symbol collisions as much as possible.

One interesting feature of Verovio is that the SVG is organized in such a way that the MEI structure is preserved as much as possible. For example, a `<note>` element with an `@xml:id` attribute in the MEI file will have a corresponding `<g>` element in the SVG with an `@class` attribute equal to `"note"` and an `@id` attribute corresponding to the `@xml:id` of the MEI note. This makes interaction with the SVG very easy. The hierarchy of the elements is also preserved. For example, in MEI, a `<beam>` can be the child element of a `<tuplet>`, but the opposite is also possible. The hierarchy is fully preserved in the SVG as shown in Figure 3.

```
<tuplet xml:id="t1" num="3" numbase="2">
  <beam xml:id="b1">
    <note xml:id="n1" pname="d" oct="5" dur="8" />
    <note xml:id="n2" pname="e" oct="5" dur="16" dots="1"/>
    <note xml:id="n3" pname="d" oct="5" dur="32" />
    <note xml:id="n4" pname="c" oct="5" dur="8" accid="s"/>
  </beam>
</tuplet>
<beam xml:id="b2">
  <tuplet xml:id="t2" num="3" numbase="2">
    <note xml:id="n5" pname="d" oct="5" dur="8" />
    <note xml:id="n6" pname="e" oct="5" dur="16" dots="1"/>
    <note xml:id="n7" pname="f" oct="5" dur="32" accid="s"/>
    <note xml:id="n8" pname="e" oct="5" dur="8"/>
  </tuplet>
</beam>
```



```
<g class="tuplet" id="t1" >
  <g class="beam" id="b1" >
    <g class="note" id="n1" >...</g>
    <g class="note" id="n2" >...</g>
    <g class="note" id="n3" >...</g>
    <g class="note" id="n4" >...</g>
  </g>
</g>
<g class="beam" id="b2" >
  <g class="tuplet" id="t2" >
    <g class="note" id="n5" >...</g>
    <g class="note" id="n6" >...</g>
    <g class="note" id="n7" >...</g>
    <g class="note" id="n8" >...</g>
  </g>
</g>
```

Figure 3. Comparison of MEI and SVG file structures. The hierarchy of the MEI is preserved in the SVG.

3.3 Programming architecture

Verovio is designed as a fast, portable, lightweight tool usable as a one-step conversion program. It is written in pure standard C++ with no dependencies on third-party frameworks and libraries. This ensures maximum portability of the codebase. Verovio implements its own rendering engine, which can produce SVG with all the musical symbols embedded in it. The musical glyphs are themselves SVG graphics that are included in the Verovio output. This means that no external font needs to be included in the SVG generated from Verovio, limiting dependencies and reducing as far as possible any potential compatibility issues between SVG rendering engines.

The Verovio rendering engine itself is defined as an abstract class, and the SVG output is the default concrete class. This makes it relatively easy to implement a rendering back-end different from SVG (e.g., PDF, or HTML Canvas), if necessary.

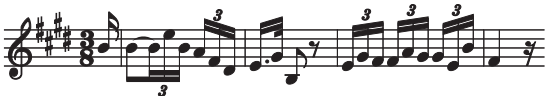
The Verovio toolkit has several options for controlling the output. These include options for defining the page size (i.e., the surface, or `<svg>` element size), for setting the amount of zoom, and for choosing whether layout information contained in the MEI file must be taken into account. When there is no layout information provided in the MEI file (no system or page breaks, for example), or when the option for ignoring them is selected, Verovio will extrapolate the necessary layout information.

3.4 Features

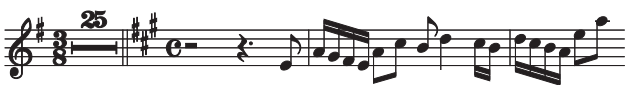
Verovio currently supports the basic features of simple common Western music notation and mensural notation.

Table 1 shows a list of music notation snippets rendered with Verovio. Figure 4 illustrates how the SVG output of Verovio can be used as facsimile overlay when the positioning feature of the MEI page-based customization is used. The example also illustrates mensural music notation support.

Beams and tuplets

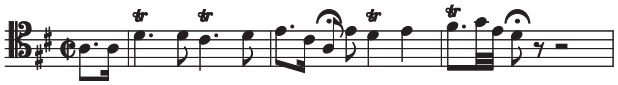


Measure rests and key and time signature changes



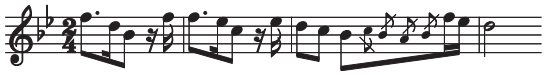
Clef changes

Trills and fermata



Ties

Grace notes (accaciature)



Grace notes (appogiature)

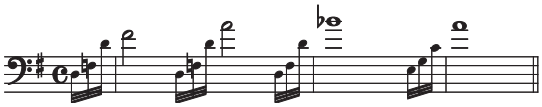


Table 1. A list of music notation snippets rendered with the Verovio toolkit. The basic features of simple common Western music notation are accounted for.

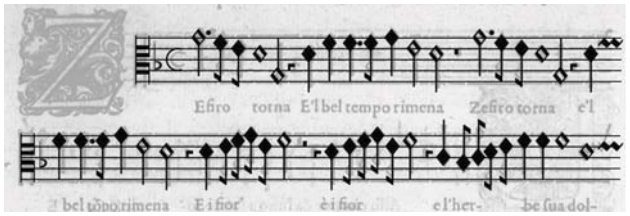


Figure 4. An example of the output of Verovio placed back on top of a facsimile image and acting as transcription overlay. In this case, positioning information was available in the page-based MEI encoding.

4. USE OF VEROVIO

4.1 C++ tools and library

Several use cases can be imagined for the Verovio toolkit. First of all, it can be built and used as a standalone command-line tool. This option is well-suited to scripting environments and applications. The command-line tool can be used to render music notation files (in MEI, PAE or DARMS) into SVG. It can also be used to convert DARMS or PAE to MEI. Another option is to use Verovio as a music notation rendering library that can be statically or dynamically linked to full applications. In such cases, it is also relatively easy to implement another drawing back-end for the corresponding C++ graphic environment for the music to be rendered directly on the screen. This is the case with the Aruspix optical music recognition software application where Verovio provides a screen rendering using a wxWidgets back-end instead of the standard SVG one. This approach is conceivable for any C++ graphical environments, be they cross-platform, like the Qt or JUCE toolkits, or platform specific.

4.2 JavaScript toolkit

The Verovio toolkit can also be compiled to JavaScript using the Emscripten LLVM-to-JavaScript compiler [19]. In this case, it behaves similarly to the command-line tool but in the web browser context. This approach is particularly interesting because it provides a complete in-browser music MEI typesetter that can be easily integrated into web-based applications.

Emscripten does not directly translate C++ into JavaScript. Instead it takes the LLVM (Low Level Virtual Machine) byte code generated by the Clang compiler from the C++ code as a base for the conversion to JavaScript. This has several advantages. Most importantly, the level of completeness in terms of C++ language feature support is extremely high since the idiomatic features of C++ did not have to be explicitly translated into JavaScript in the Emscripten compiler (only the translation from LLVM was necessary). In fact, for the Verovio toolkit, the Emscripten compiler is applied on exactly the same codebase as the C++ compiler, and no change to the code had to be done for this to work. Only the compilation makefile is different.

Another advantage of this approach is that the JavaScript produced is very fast because it benefits from all the code optimization performed by the Clang compiler when generating the LLVM byte code. Furthermore, in addition to standard JavaScript, Emscripten can also generate asm.js code, a subset of JavaScript that has the advantage of being highly optimizable. On web browsers that support asm.js (currently Firefox, Chrome and Opera), the execution speed is only up to about 1.6 times slower than with the native C++ executable. Table 2 shows the system time required to load an MEI file of 120 pages of

music (7 MB) and for displaying the first page with the native executable and with three web browsers. The figures are the median value of the operation repeated 100 times.

	Native	Firefox	Chrome	Safari
System time in sec.	0.657	1.054	1.364	1.811
Comparison to native	-	1.6	2.1	2.8

Table 2. The system time in seconds for loading an MEI files (120 pages, 7 MB) and for displaying the first page. The second line gives the ratio with the native executable time for the three web browsers used for comparison.

The JavaScript version of the Verovio toolkit is easy to use in web environments. It is packed in one single file which size is only about 1.2 MB. It is available as a JavaScript class, and all the options of the command-line version are supported in the toolkit. The options can be passed to the toolkit in JSON format, and the SVG output can be directly fed to HTML objects for display. The Figure 5 shows a HTML and Javascript code snippet for loading an MEI file using a jQuery HTTP GET request.

```
<script src="verovio-toolkit.js"></script>
<!-- The div where we are going to insert the SVG -->
<div id="output"/>
<script type="text/javascript">
  /* Create the Vevorio toolkit instance */
  var vrvToolkit = new verovio.toolkit();
  /* Load the file using HTTP GET */
  $.get( "mei-file.mei", function( data ) {
    /* Render the data */
    var svg = vrvToolkit.renderData(
                data + "\n",
                JSON.stringify({ inputFormat: 'mei' }) );
    /* Insert the data as content of the #output div */
    $("#output").html(svg);
  });
</script>
```

Figure 5. A JavaScript example for loading an MEI file. The toolkit parameters can be set using JSON.

The layout of the MEI data is performed on loading. Once the file is loaded into memory, it remains accessible in the toolkit instance. The class provides methods for getting the number of pages or for navigating through them, making it convenient to integrate the toolkit in a JavaScript application.

The Figure 6 shows a screenshot of a web application where the toolkit was turned into an online MEI file viewer. The application works on desktop computers but also on tablets and mobile devices. The JavaScript toolkit has been tested with recent versions of the most widely used web-browsers. Internet Explorer requires at least version 10.

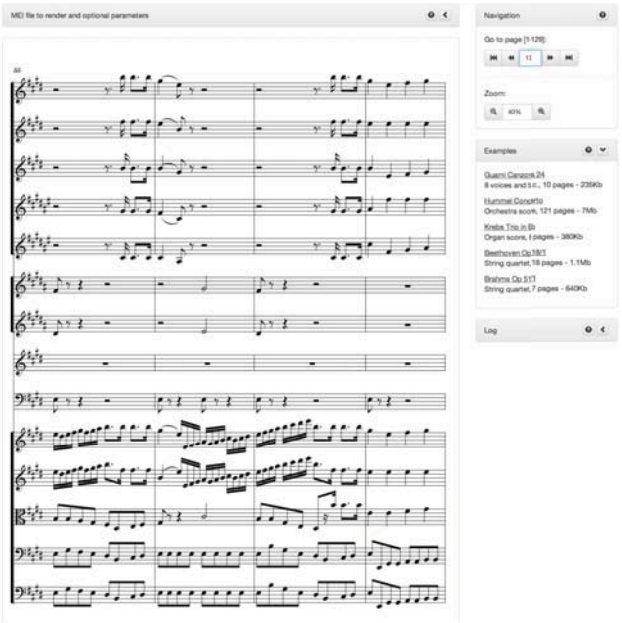


Figure 6. An example of a web-based MEI viewer built with the Verovio toolkit. Large MEI files can be loaded and displayed in the web browser in a very convenient way.

5. CONCLUSION AND FUTURE WORK

Verovio is a toolkit for rendering MEI in SVG that can be used in different application environments, including online. It is designed with MEI in mind, making it the right basis for implementing encoding features that are specific to MEI. It will avoid problematic situations that occur when using rendering engines based on other formats and that implement a different data structure. Even if at this stage, the supported features can be in some cases more limited than with other rendering options, Verovio already implements many important features for rendering both common Western music notation and mensural notation.

Current work on Verovio includes the adoption of the Standard Music Font Layout (SMuFL) [17] for supporting other fonts converted to SVG glyphs, the improvement of the SVG structure and adding support for additional MEI elements and attributes. The priority is given to features specific to MEI. The future work will include the development of a prototype for making Verovio a possible basis for an online MEI editor. It will also include the creation of an MEI application profile for Verovio using the TEI One Document Does-it-all (ODD) approach. The corresponding XSL stylesheets for converting to it other MEI profiles will also be provided. Adding the import of other encoding formats is also envisaged in the future.

6. AVAILABILITY

Verovio can be downloaded from http://www.verovio.org and is available under the GPLv3 open-source license. The website also includes documentation on currently available features.

7. REFERENCES

[1] Arkkra Enterprises, *Mup.* <http://www.arkkra.com>

[2] G. A. Bays: *ScoreSVG: A New Software Framework for Capturing the Semantic Meaning and Graphical Representation of Musical Scores Using Java2D, XML, and SVG.* Diss. Georgia State Univ., 2005.

[3] G. Burlet, A. Porter, A. Hankinson, and I. Fujinaga: "Neon.js: Neume Editor Online," *Proceedings of the 13th International Society on Music Information Retrieval Conference,* pp. 121–6, 2012.

[4] Corpus Monodicum, *mono:di.* <http://monodi.corpus-monodicum.de>

[5] Edirom, *mei2abc*. <https://github.com/edirom/mei2abc>

[6] A. Hankinson, P. Roland, and I. Fujinaga: "The Music Encoding Initiative as a document-encoding framework," *Proceedings of the 12th International Society on Music Information Retrieval Conference,* pp. 293–8, 2011.

[7] J. Howard: "Plaine and Easie code: A code for music bibliography," in Selfridge-Field, E. (Ed.), *Beyond MIDI: The Handbook of Musical Codes*. The MIT Press, Cambridge, pp. 362–72, 1997.

[8] MEI, *mei2mup*. <http://code.google.com/p/music-encoding/source/browse/trunk/tools/mei2mup>

[9] MEI, *mei2musicxml*. <https://code.google.com/p/music-encoding/source/browse/trunk/tools/mei2musicxml>

[10] MEI-incubator, *page-based customization*, <https://code.google.com/p/mei-incubator/source/browse/page-based>

[11] S. Morent: "Digitale Edition älterer Musik am Beispiel des Projekts TüBingen," in *Digitale Edition zwischen Experiment und Standardisierung. Musik – Text – Codierung*, pp. 89–109, 2009.

[12] M. Muthanna, *VexFlow.* <https://github.com/0xfe/vexflow>

[13] H. W. Nienhuys and J. Nieuwenhuizen: "LilyPond, a system for automated music engraving," *Proceedings of the XIV Colloquium on Musical Informatics (XIV CIM 2003)*, pp. 167–72, 2003.

[14] L. O'Shea: "Stirring XML: Visualizations in SVG: MusicML2SVG," *Proceedings of the SVGOpen2003 Conference,* pp. 2–6, 2003.

[15] P. Rosen, *abcjs*. <http://github.com/paulrosen/abcjs>

[16] E. Selfridge-Field: "DARMS, its dialects, its uses," in Selfridge-Field, E. (Ed.), *Beyond MIDI: The Handbook of Musical Codes*. The MIT Press, Cambridge, pp. 163–74, 1997.

[17] Steinberg, *Standard Music Font Layout.* <http://www.smufl.org>

[18] TEI Music SIG, *MEItoVexFlow*. <http://github.com/tei-music-sig/meitovexflow>

[19] A. Zakai: "Emscripten: an LLVM-to-JavaScript compiler," *Companion to the 26th Annual ACM OOPSLA Conference*, pp. 301–12, 2011.

MUSIC CLASSIFICATION BY TRANSDUCTIVE LEARNING USING BIPARTITE HETEROGENEOUS NETWORKS

Diego F. Silva, Rafael G. Rossi, Solange O. Rezende, Gustavo E. A. P. A. Batista
Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo
{diegofsilva,ragero,solange,gbatista}@icmc.usp.br

ABSTRACT

The popularization of music distribution in electronic format has increased the amount of music with incomplete metadata. The incompleteness of data can hamper some important tasks, such as music and artist recommendation. In this scenario, transductive classification can be used to classify the whole dataset considering just few labeled instances. Usually transductive classification is performed through label propagation, in which data are represented as networks and the examples propagate their labels through their connections. Similarity-based networks are usually applied to model data as network. However, this kind of representation requires the definition of parameters, which significantly affect the classification accuracy, and presents a high cost due to the computation of similarities among all dataset instances. In contrast, bipartite heterogeneous networks have appeared as an alternative to similarity-based networks in text mining applications. In these networks, the words are connected to the documents which they occur. Thus, there is no parameter or additional costs to generate such networks. In this paper, we propose the use of the bipartite network representation to perform transductive classification of music, using a bag-of-frames approach to describe music signals. We demonstrate that the proposed approach outperforms other music classification approaches when few labeled instances are available.

1. INTRODUCTION

The popularity of online music services has dramatically increased in the last decade. The revenue of online services, such as music streaming, has more than triplicate in the last three years. Online services already account for a significant 40% of the overall industry trade revenues [1]. However, the popularization of music and video clips distribution in electronic format has increased the amount of music with incomplete metadata. The incompleteness of the data can hamper some important tasks such as indexing, retrieval and recommendation.

For instance, users of music services commonly define their preferences based on genre information. A recommendation system can make use of such information to suggest other music conditional to the expressed preferences. The lack of genre information on music imposes limits to the capability of the recommendation systems to correctly identify consume patterns as well as to recommend a diverse set of music. Similar statements can be made for music indexing and retrieval.

Due to the academic and commercial importance of digital music, we have witnessed in the last decade a tremendous increase of interest for Music Information Retrieval (MIR) tasks. Most of the proposed MIR methods are based on supervised learning techniques. Supervised learning usually requires a substantial amount of correctly labeled data in order to induce accurate classifiers. Although labeled data can be obtained with human supervision, such process is usually expensive and time consuming. A more practical approach is to employ methods that can avail of both a small number of labeled instances and a large amount of unlabeled data.

Transductive learning directly estimates the labels of unlabeled instances without creating a classification model. A common approach to perform transductive classification is label propagation, in which the dataset is represented as a network and the labels of labeled instances are propagated to the unlabeled instances through the network connections. Similarity-based networks are usually applied to represent data as networks for label propagation [19]. However, they present a high cost due to the computation of the similarities among all dataset instances, and require the definition of several graph construction parameters that can significantly affect the classification accuracy [11].

Bipartite heterogeneous networks have appeared as an alternative to similarity-based networks in sparse domains, such as text mining [9, 10]. In these networks, words are connected to documents in which they occur. Thus, there are no parameters or additional costs to generate such networks. In a similar way, we can represent music collections as a bipartite network though the use of a bag-of-frames (BoF) representation. The BoF is a variation of bag-of-words (BoW) representation used in text analysis and has been applied in studies of genre recognition, music similarity, and others [12].

In this paper, we propose the use of the bipartite network representation to perform transductive classification of music, using a BoF approach to describe music signals. We demonstrate that the proposed approach outperforms other music classification approaches when few labeled instances are available.

2. BACKGROUND & RELATED WORK

Transductive classification is a useful way to classify all dataset instances when just few labeled instances are available [19]. Perhaps the most common and intuitive way to perform transductive classification is through label propagation, which is commonly made by using similarity-based networks to represent the data. Usual ways to generate similarity-based networks are [17–19]: (*i*) fully connected-network, in which every pair of instances are connected; (*ii*) k nearest neighbor, in which an instance is connected with its k most similar instances; and (*iii*) ϵ network, in which two instances are connected if their distance is above a threshold.

Bipartite networks have appeared as an alternative to similarity-based networks in sparse domains such as texts [9, 10]. The use of bipartite networks to represent text collections and the use of algorithms which perform label propagation in bipartite networks obtained results as good as the obtained by similarity-based networks [10]. However, the computation cost to generate similarity-based networks is $O(|I|^2 \times |A|)$, in which $|I|$ is the number of intances and $|A|$ is the number of attributes of a dataset, while the computational cost to generate bipartite networks is $O(|I| \times |A|)$. Morevorer, the gerenation of bipartite networks is parameter-free.

We can represent music collections as a bipartite network though the generation of a bag-of-frames (BoF). Methods using BoF has became common in different MIR tasks, including similarity, genre, emotion and cover song recognition [12]. Such strategies basically consist of three main steps: feature extraction, codebook generation and learning/classification.

Probably, the most simple and commonly used strategy for the codebook generation is the Vector Quantization (VQ). Basically, the VQ uses clustering algorithms on the frame-level features and consider the center of clusters as the words of a dictionary. The simple k-means is, probably, the most used algorithm in this step and showed to achieve similar results to other methods [8].

Recently, new tools have emerged for creating codebooks. Specifically, strategies based on Sparse Coding [5, 15] and Deep Belief Networks [3, 7] have been widely used. However, even though these strategies often improve the results in different domains, they can present similar performance to simple strategies such as VQ in certain tasks [7].

3. PROPOSED FRAMEWORK: MC-LPBN

In this paper we propose a framework called MC-LPBN (Music Classification through Label Propagation in Bipartite Networks) to perform transductive classification of musics. The proposed framework has three main steps: (*i*) codebook generation, (*ii*) network generation for transductive classification, and (*iii*) transductive classification using bipartite heterogeneous networks. In the next subsections we present the details of each step.

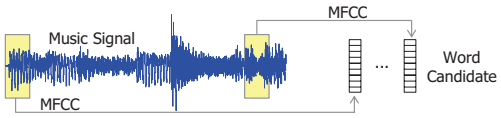


Figure 1. Word candidates generation process

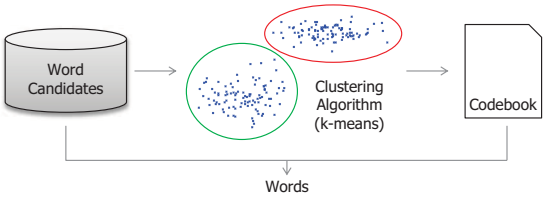


Figure 2. The word candidates are clustered and each centroid is elected as a codeword. The word frequency is directly related to the candidates count in each cluster

3.1 Codebook Generation and Bag-of-Frames

In order to represent a music collection as a BoF it is necessary to extract a set of words. Such procedure starts with the extraction of word candidates. A word candidate is a set of features extracted from a single window. As a sliding window swipes across the entire music signal, each music gives origin to a set of word candidates. In this work we use MFCC as feature extraction procedure. Figure 1 illustrates the word candidates generation process.

The next step is the creation of a codebook. A codebook is a set of codewords used to associate the word candidates to a finite set of words. The idea is to select the most representative codeword for each word candidate. To do this, we use a clustering algorithm with the word candidates and consider the center of each cluster as a codeword. So, each candidate is associated to the codeword that represents the cluster it belongs. In this step, we used the ubiquitous k-means algorithm, due to its simplicity and efficiency. Figure 2 illustrates this procedure.

Finally, there is a step to the generation of a BoF matrix. In such a matrix, each line corresponds to a music recording, each column corresponds to a word and the cells correspond to the frequency of occurrence of the word in the music. The BoF allows the generation of bipartite networks for transductive classification, as we discuss in the next subsection.

3.2 Network Generation for Transductive Classification

Formally, a network is defined by $N = \langle \mathcal{O}, \mathcal{E}, \mathcal{W} \rangle$, in which $\mathcal{O}$ represents the set of objects (entities) of a problem, $\mathcal{E}$ represents the set of connections among the objects and $\mathcal{W}$ represents the weights of the connections. When

$\mathcal{O}$ is composed by a single type of object, the network is called homogeneous network. When $\mathcal{O}$ is composed by h different types of objects, i.e., $\mathcal{O} = \mathcal{O}_1 \cup \cdots \cup \mathcal{O}_h$, the networks is called heterogeneous network [6].

The music collection can be represented by a bipartite heterogeneous network with $\mathcal{O} = \mathcal{M} \cup \mathcal{T}$, in which $\mathcal{M} = \{m_1, m_2, \ldots, m_n\}$ represents the set of music and $\mathcal{T} = \{t_1, t_2, \ldots, t_l\}$ represents the set of words. $\mathcal{M}$ is composed by labeled ($\mathcal{M}^L$) and unlabeled ($\mathcal{M}^U$) music, i.e., $\mathcal{M} = \mathcal{M}^L \cup \mathcal{M}^U$. A music $m_i \in \mathcal{M}$ and a word $t_j \in \mathcal{T}$ are connected if t_j occurs in m_i. The weight of the relation between m_i and t_j (w_{m_i,t_j}) is the frequency of t_j in m_i. Thus, only the words and their frequencies in the music are needed to generate the bipartite network.

For transductive classification based on networks, let $\mathcal{C} = \{c_1, c_2, \ldots, c_l\}$ be the set of possible labels, and let $\mathbf{f}_{o_i} = \{f_1, f_2, \ldots, f_{|\mathcal{C}|}\}^T$ be the weight vector of an object o_i, which determines its weight or relevance for each class. Hence, it is also referred as class information vector. The class information of an object $m_i \in \mathcal{M}$ or an object $t_j \in \mathcal{T}$ is denoted respectively by $\mathbf{f}_{m_i}$ and $\mathbf{f}_{t_j}$. All the class information of objects in $\mathcal{M}$ or $\mathcal{T}$ is denoted by the matrices $\mathbf{F}(\mathcal{M}) = \{\mathbf{f}_{m_1}, \mathbf{f}_{m_2}, \ldots, \mathbf{f}_{m_{|\mathcal{M}|}}\}^T$ and $\mathbf{F}(\mathcal{T}) = \{\mathbf{f}_{t_1}, \mathbf{f}_{t_2}, \ldots, \mathbf{f}_{t_{|\mathcal{T}|}}\}^T$. The class information of a labeled music m_i is stored in a vector $\mathbf{y}_{m_i} = \{y_1, y_2, \ldots, y_{|\mathcal{C}|}\}^T$, which has the value 1 to the corresponding class position and 0 to the others. The weights of connections among objects are stored in a matrix $\mathbf{W}$. A diagonal matrix $\mathbf{D}$ is used to store the degree of the objects, i.e., the sum of the connection weights of the objects. Thus the degree of a music m_i in a bipartite network is ($d_{m_i} = d_{i,i} = \sum_{t_j \in \mathcal{T}} w_{m_i,t_j}$).

3.3 Transductive Classification Using Bipartite Heterogeneous Networks

The main algorithms for transductive classification in data represented as networks are based on regularization [19], which have to satisfy two assumptions: (*i*) the class information of neighbors must be close; and (*ii*) the class information assigned during the classification process must be close to the real class information. In this paper we used three regularization-based algorithms: (*i*) Tag-based classification Model (TM) [16], (*ii*) Label Propagation based on Bipartite Heterogeneous Networks (LPBHN) [10], and (*iii*) GNetMine (GM) [6].

TM algorithm minimizes the differences among (*i*) the real class information and the class information assigned to music ($\mathcal{M}$), (*ii*) the real class information and the class information assigned to and objects from other domains that aid the classification ($\mathcal{A}$), and (*iii*) the class information among words ($\mathcal{T}$) and objects in ($\mathcal{M}$) or ($\mathcal{A}$), as presented in Equation 1.

$$Q(\mathbf{F}) = \alpha \sum_{a_i \in \mathcal{A}} ||\mathbf{f}_{a_i} - \mathbf{y}_{a_i}||^2 + \beta \sum_{m_i \in \mathcal{M}^L} ||\mathbf{f}_{m_i} - \mathbf{y}_{m_i}||^2 \quad (1)$$
$$+\gamma \sum_{m_i \in \mathcal{M}^U} ||\mathbf{f}_{m_i} - \mathbf{y}_{m_i}||^2 + \sum_{o_i \in \mathcal{M} \cup \mathcal{A}} \sum_{t_j \in \mathcal{T}} w_{o_i,t_j} ||\mathbf{f}_{o_i} - \mathbf{f}_{t_j}||^2$$

The parameters α, β and γ control the importance given for each assumption of TM. Objects are classified using class mass normalization [18].

LPBHN is a parameter-free algorithm based on the Gaussian Fields and Harmonic Functions (GFHF) algorithm [18], which performs label propagation in homogeneous networks. The difference is that LPBHN considers the relations among different types of objects. The function to be minimized by LPBHN is:

$$Q(\mathbf{F}) = \frac{1}{2} \sum_{m_i \in \mathcal{M}} \sum_{t_j \in \mathcal{T}} w_{m_i,t_j} ||\mathbf{f}_{m_i} - \mathbf{f}_{t_j}||^2 \quad (2)$$
$$+ \lim_{\mu \to \infty} \mu \sum_{m_i \in \mathcal{M}^L} ||\mathbf{f}_{m_i} - \mathbf{y}_{m_i}||^2,$$

in which μ tending to infinity means that $\mathbf{f}_{m_i} \equiv \mathbf{y}_{m_i}$, i.e., the information class of labeled musics do not change.

The GM framework is based on the Learning with Local and Global Consistency (LLGC) algorithm [17], which performs label propagation in homogeneous networks. The difference between the algorithms is that GM considers the different types of relations among the different types of objects. For the problem of music classification using bipartite networks, GM minimizes the differences of (*i*) the class information among music and words and (*ii*) the class information assigned to labeled music during the classification and their real class information. The function to be minimized by GM is:

$$Q(\mathbf{F}) = \sum_{m_i \in \mathcal{M}} \sum_{t_j \in \mathcal{T}} w_{m_i,t_j} \left|\left| \frac{\mathbf{f}_{m_i}}{\sqrt{d_{m_i}}} - \frac{\mathbf{f}_{t_j}}{\sqrt{d_{t_j}}} \right|\right|^2 \quad (3)$$
$$+ \sum_{m_i \in \mathcal{M}} \mu ||\mathbf{f}_{m_i} - \mathbf{y}_{m_i}||^2,$$

in which $0 < \mu < 1$.

We highlight that all the algorithms presented above have iterative solutions to minimize the respective equations. This allows to obtain similar results to the closed solutions with a lower computational time.

4. EXPERIMENTAL EVALUATION

To illustrate the generality of our approach, we evaluate our framework in two different scenarios. In this section, we describe the tasks we considered, the experimental setup used in our experiments, as well as the results obtained and a short discussion about them.

4.1 Tasks Description

We evaluate our framework in genre recognition and cover song recognition scenarios. The remaining of this section contains a brief description of each task and the datasets used to each end.

4.1.1 Genre Recognition

Genre recognition is an important task in several applications. Genre is a quality created by the human beings to

intuitively characterize music [14]. For humans, the classification of music by genre is relatively simple task, and can be done by listening to a short excerpt of a music.

Therefore, most of the existing data for this task considers a short duration excerpt for each recording. In this work, we use the GTZAN [1] and Homburg [2] datasets. The first has $1,000$ snippets of 30 seconds of ten different genres. The number of instances of each class is equally distributed. The Homburg dataset, in turn, has ten seconds sections of $1,886$ recordings, belonging to nine genres. In this case, the genre with fewer instances has only 47 examples, while the largest has 504.

4.1.2 Cover Song Recognition

Cover song may be defined as distinct performances of the same music with differences in tempo, instrumentation, style or other characteristics [4]. Finding reinterpreted music is an important task mainly to commercial ends. For example, it can be used to ensure copyright in websites which allow users to create content.

In this paper, we evaluate our framework in a task similar to the cover song recognition. But, instead find the original recording of a query music, we consider all different interpretations of the same music as the same class.

To evaluate our proposal we used the Mazurkas Project data [3], in which each music has several versions. This dataset contains 2914 recordings of 49 Chopin's mazurkas for piano (from 43 to 97 versions per class).

4.2 Experimental Setup

We evaluated our framework considering different configurations for the 1^{st} and 3^{rd} steps. For the 1^{st} step, we consider variations of parameters of the feature extraction and codebook generation phases. In this work, we use 20 MFCC as frame-level features. This number is a common choice in MIR papers [2]. We use 5 different window sizes, with an overlap of 50% between them: 0.0625, 0.125, 0.25, 0.5 and 0.75 seconds. Finally, we applied the k-means using $k \in \{100, 200, 400, 800, 1600, 3200\}$.

For the 3^{rd}, we consider the algorithms presented in Section 3.3: Label Propagation using Bipartite Heterogeneous Networks (LPBHN), Tag-based Model (TM), and GNetMine (GM). For GM we use the parameter $\alpha \in \{0.1, 0.3,\ 0.5, 0.7, 0.9\}$. For TM we use $\alpha = 0$, since there are no objects from different domains, $\beta \in \{0.1, 1.0,\ 10, 100, 1000\}$, and $\gamma \in \{0.1, 1.0, 10.0, 100.0,\ 1000.0\}$. The iterative solutions proposed by the respective authors were used for all the algorithms. The maximum number of iterations is set to 1000 since this is a common limit value for iterative solutions.

We also carried out experiments using two algorithms for label propagation in similarity-based networks, Learning with Local and Global Consistency (LLGC) [17] and Gaussian Fields and Harmonic Functions (GFHF) [18], and two classical supervised learning algorithms for comparison with the proposed approach, k Nearest Neighbors (kNN) and Support Vector Machines (SVM) [13].

We build similarity-based networks using the fully connected approach with $\sigma = 0.5$ [19] and we set $\alpha = \{0.1, 0.3, 0.5, 0.7, 0.9\}$ for LLGC algorithm. For kNN we set $k = 7$ and weighted vote, and for SVM we used linear kernel and $C = 1$ [9].

The metric used for comparison was the classification accuracy, i.e., the percentage of correctly classified music recordings. The accuracies are obtained considering the average accuracies of 10 runs. In each run we randomize the dataset and select x examples as labeled examples. The remaining $|\mathcal{M}| - x$ examples are used for measuring the accuracy. We carried out experiments using $x = \{1, 10, 20, 30, 40, 50\}$ to analyze the trade-off between the number of labeled documents and classification accuracy. The best accuracies obtained by some set of parameters of the algorithms are used for comparison.

4.3 Results and Discussion

Given the large amount of results obtained in this work, their complete presentation becomes impossible due to space constraints. Thus, we developed a website for this work, where detailed results can be found [4]. In this section, we summarize the results from different points of view.

4.3.1 Influence of Parameters Variation

Our first analysis consists in evaluating the influence of the variation of the codebook generation step parameters. Figure 3 presents the variation of accuracy for both genre recognition dataset and each window size according to a different number of words in the dictionary. To do this, we fixed the number of labeled examples in 10. This number represents a good trade-off between the classification accuracy of the algorithms and the human effort to label music. But, we note that the behaviors are similar to other numbers of properly labeled examples.

The results show that the transductive learning methods can achieve similar or even superior results than the obtained by using inductive models. In the case of GTZAN data, there is a clear increasing pattern when the number of words varies. Using higher values to it, both strategies perform well, but the transductive learning obtained the higher accuracies. The results obtained by similarity-based networks were slightly better in most of configurations. But, as mentioned before, similarity-based networks has a high cost to calculate the similarities between all the examples and require the setting of several parameters to construct the network. In the Homburg dataset, transductive learning is better independently of the parameter configuration. In this case, there are no significant differences between bipartite network approaches, but they performed better than the similarity-based networks in most of cases.

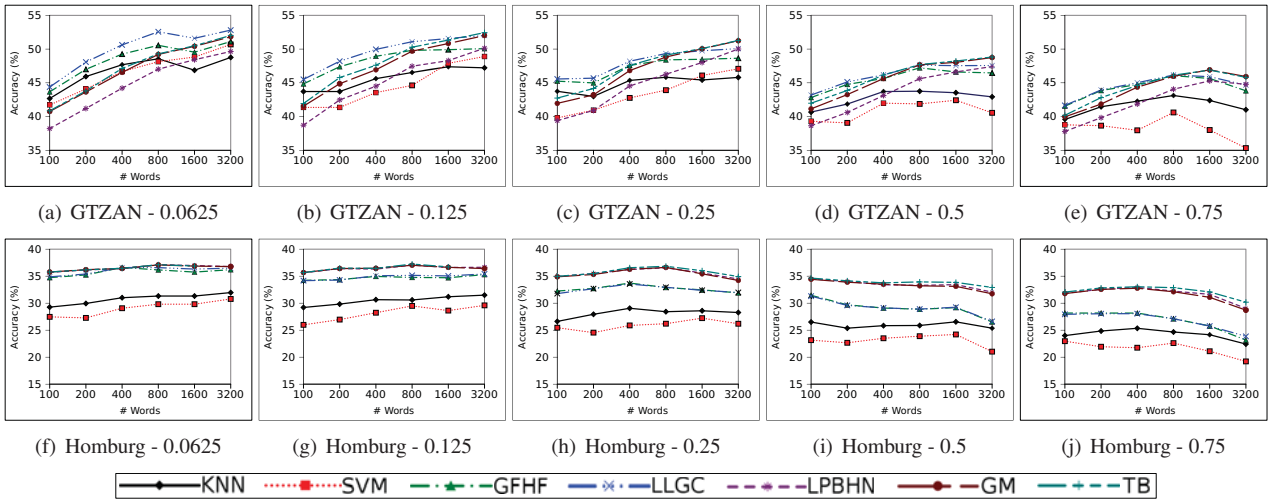
In order to evaluate our framework in the cover song recognition, we used the LPBHN algorithm, that achieved

[1] http://marsyas.info/download/data_sets/

[2] http://www-ai.cs.uni-dortmund.de/audio.html

[3] http://www.mazurka.org.uk/

[4] http://sites.labic.icmc.usp.br/dfs/ismir2014



(a) GTZAN - 0.0625 (b) GTZAN - 0.125 (c) GTZAN - 0.25 (d) GTZAN - 0.5 (e) GTZAN - 0.75

(f) Homburg - 0.0625 (g) Homburg - 0.125 (h) Homburg - 0.25 (i) Homburg - 0.5 (j) Homburg - 0.75

Figure 3. Accuracy for genre recognition by varying the number of words in the codebook. The number of labeled examples per class is fixed in 10.

similar result to other transductive methods and has the advantage of being parameter-free, and both, SVM and kNN, inductive approaches. The results show a high increasing pattern when transductive learning is used, and a more stable pattern to supervised methods. Figure 4 shows the accuracy achieved by fixing the number of labeled examples in 10. We fixed the window size to 0.75 seconds, since Mazurkas is the larger dataset used in this work and this is the fastest configuration to the feature extraction phase. We believe that the performance of transductive learning can overcome the SVM if we increase the number of words.

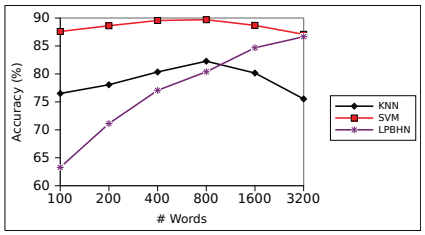


Figure 4. Accuracy for Mazurkas by varying the number of words in the codebook. The number of labeled examples is fixed in 10 and the window length in 0.75 seconds.

4.3.2 Number of Labeled Examples Variation

The evaluation of the performance variation according to the number of labeled examples is an important analysis in the context of transductive learning. Figure 5 shows the behavior of accuracy in genre recognition task when there is a variation in the number of labeled examples that belong to each class. The results were obtained by fixing the number of words in 3200, in which good results were achieved in several configurations, and a window size of the middle value of 0.25 seconds, since the results were similar than the obtained with other values to this parameter.

To analyze these graphs, it is interesting to know the proportion that the number of labeled examples represents in each dataset. For example, in the case of GTZAN set, 50 examples correspond to exactly 50% of the examples in each classes. In the case of Homburg, it represents 100% of the minority class, but less than 10% of the majority.

In both cases, the behavior of the accuracies was similar. As the number of labeled samples increases, the performance becomes better. The transductive learning methods performed better across the curve. As the proportion of the number of labeled instances increases, the tendency is that the performance of inductive algorithms approaches the performance of transductive algorithms.

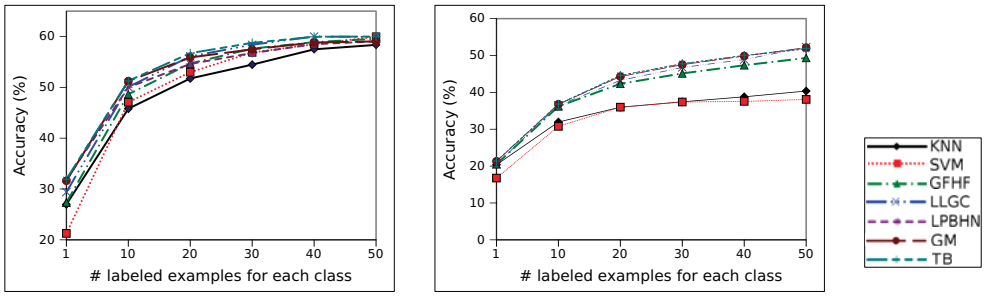
For sake of space limitations, we omitted the results for the cover song recognition task. However, we point out that the behavior of accuracy rates were very similar to obtained in the other task.

5. CONCLUSION

In this paper, we presented a framework for transductive classification of music using bipartite heteregeneous networks. We show that we can have a better performance by using this approach instead the traditional inductive learning. Our results were close or superior to the obtained by similarity-based networks. This kind of network, however, requires several parameters and has a high cost due to the calculation of the similarity between the instances.

We should note that the accuracy rates achieved in this paper are worse than some results presented in related works. For example, there are some papers that achieved accuracy higher than 80% to the GTZAN dataset using BoF approaches. However, these results were probably obtained due to the choice of specific features and parameters. Moreover, these papers used inductive learning approaches, with labels for the entire dataset. Nevertheless, we demonstrated that, for the same parameter set, the use of bipartite heterogeneous network achieved the best results.

As future work we will investigate a better feature configuration and different codebook generation strategies.



(a) GTZAN (b) Homburg

Figure 5. Accuracy of genre recognition by varying the numbers of labeled examples for each class. The number of words and the windows length are fixed in 3200 and 0.25 seconds, respectively.

ACKNOWLEDGEMENTS: We would like to thank the financial supports by the grants 2011/12823-6, 2012/50714-7, 2013/26151-5, and 2014/08996-0, Sao Paulo Research Foundation (FAPESP).

6. REFERENCES

[1] Recording industry in numbers. Technical report, International Federation of the Phonographic Industry, 2014.

[2] M. A. Casey, R. Veltkamp, M. Goto, M. Leman, C. Rhodes, and M. Slaney. Content-based music information retrieval: Current directions and future challenges. *Proceedings of the IEEE*, 96(4):668–696, 2008.

[3] S. Dieleman, P. Brakel, and B. Schrauwen. Audio-based music classification with a pretrained convolutional network. In *ISMIR*, pages 669–674, 2011.

[4] D. P. W. Ellis and T. Bertin-Mahieux. Large-scale cover song recognition using the 2d fourier transform magnitude. In *ISMIR*, pages 241–246, 2012.

[5] M. Henaff, K. Jarrett, K. Kavukcuoglu, and Y. LeCun. Unsupervised learning of sparse features for scalable audio classification. In *ISMIR*, pages 681–686, 2011.

[6] M. Ji, Y. Sun, M. Danilevsky, J. Han, and J. Gao. Graph regularized transductive classification on heterogeneous information networks. In *Proc. Eur. Conf. on Machine Learning and Knowledge Discovery in Databases*, pages 570–586. Springer-Verlag, 2010.

[7] J. Nam, J. Herrera, M. Slaney, and J. O. Smith. Learning sparse feature representations for music annotation and retrieval. In *ISMIR*, pages 565–570, 2012.

[8] M. Riley, E. Heinen, and J. Ghosh. A text retrieval approach to content-based audio retrieval. In *ISMIR*, pages 295–300, 2008.

[9] R. G. Rossi, de T. P. Faleiros, de A. A. Lopes, and S. O. Rezende. Inductive model generation for text categorization using a bipartite heterogeneous network. In *Proc. Intl. Conf. on Data Mining*, pages 1086–1091. IEEE, 2012.

[10] R. G. Rossi, A. A. Lopes, and S. O. Rezende. A parameter-free label propagation algorithm using bipartite heterogeneous networks for text classification. In *Proc. Symp. on Applied Computing*, pages 79–84. ACM, 2014.

[11] C. A. R. Sousa, S. O. Rezende, and G. E. A. P. A. Batista. Influence of graph construction on semi-supervised learning. In *Proc. Eur. Conf. Machine Learning and Knowledge Discovery in Databases*, pages 160–175. Springer-Verlag, 2013.

[12] L. Su, C.-C.M. Yeh, J.-Y. Liu, J.-C. Wang, and Y.-H. Yang. A systematic evaluation of the bag-of-frames representation for music information retrieval. *IEEE Transactions on Multimedia*, 16(5):1188–1200, Aug 2014.

[13] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison-Wesley, 2005.

[14] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE transactions on Speech and Audio Processing*, 10(5):293–302, 2002.

[15] C. C. M. Yeh, L. Su, and Y. H. Yang. Dual-layer bag-of-frames model for music genre classification. In *Intl. Conf. on Acoustics, Speech and Signal Processing*, 2013.

[16] Z. Yin, R. Li, Q. Mei, and J. Han. Exploring social tagging graph for web object classification. In *Proc. Intl. Conf. on Knowledge Discovery and Data Mining*, pages 957–966, 2009.

[17] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems*, volume 16, pages 321–328, 2004.

[18] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proc. Intl. Conf. on Machine Learning*, pages 912–919. AAAI Press, 2003.

[19] X. Zhu, A. B. Goldberg, R. Brachman, and T. Dietterich. *Introduction to Semi-Supervised Learning*. Morgan and Claypool Publishers, 2009.

AUTOMATIC MELODY TRANSCRIPTION BASED ON CHORD TRANSCRIPTION

Antti Laaksonen
Department of Computer Science
University of Helsinki
ahslaaks@cs.helsinki.fi

ABSTRACT

This paper focuses on automatic melody transcription in a situation where a chord transcription is already available. Given an excerpt of music in audio form and a chord transcription in symbolic form, the task is to create a symbolic melody transcription that consists of note onset times and pitches. We present an algorithm that divides the audio into segments based on the chord transcription, and then matches potential melody patterns to each segment. The algorithm uses chord information to favor melody patterns that are probable in the given harmony context. To evaluate the algorithm, we present a new ground truth dataset that consists of 1,5 hours of audio excerpts together with hand-made melody and chord transcriptions.

1. INTRODUCTION

Melody and chords have a strong connection in Western music. The purpose of this paper is to exploit this connection in automatic melody transcription. Given a chord transcription, we can use it in melody transcription to constrain the set of possible melodies. Both the rhythm and the pitches of the melody should match the chords in a successful melody transcription.

For example, let us consider the melody in Figure 1. The melody consists of 16 bars, each annotated with a chord symbol. The first observation is that chord boundaries divide the melody into segments of approximately equal length. Each of the segments has a simple rhythmical structure. In this case the chord boundaries exactly match the bar lines, and each segment contains up to three melody notes. Of course, many melodies are more complex than this, but the underlying segmentation is still usually apparent.

Let us now consider the pitches of the melody. The key of the melody mainly determines what pitches typically occur in the melody. In this example the key of the melody is C major, and almost all melody pitches belong to the C major scale. However, there are two exceptions: the G# note in the second bar and the C# note in the sixth bar. Thus, although the melody follows the C major scale, the individual chords also have an effect on the pitches. In this case the major thirds of E major and A major chords are so predominant that the melody adapts to the harmony.

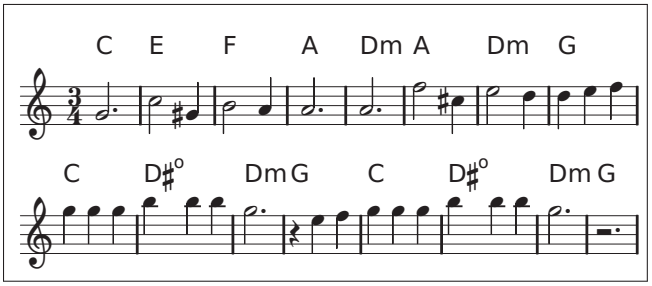


Figure 1: A melody from Disney's *Snow White and the Seven Dwarfs*. The chord transcription consists of 16 chords, and the melody transcription consists of 30 notes.

Human transcribers routinely use this kind of musical knowledge in music transcription. If the melody does not match the chords, or the chords do not match the melody, the transcription cannot be correct. However, in automatic music transcription, chord extraction and melody extraction have been studied separately for the most part.

Currently, the best automatic systems for chord transcription produce promising results, while melody transcription seems to be a more challenging problem. For this reason, we approach automatic melody transcription with the assumption that a chord transcription has already been done. We present an algorithm that divides the audio data into segments based on the chord boundaries. After this, the algorithm assigns each segment a melody pattern that matches both the audio data and the chord information.

1.1 Problem statement

Given an excerpt of music in audio form and a chord transcription in symbolic form, the task is to produce a melody transcription in symbolic form. We concentrate on typical Western music, and assume that the pitches of the notes are given in semitones.

We assume that the audio data A is divided into n_A frames of equal length using some preprocessing method. For each audio frame k ($1 \leq k \leq n_A$), we are given values $A[k].begin$ and $A[k].end$ that are time values in seconds

when the frame begins and ends. In addition, for each possible melody note q we are given a real number $A[k][q]$ in the range $[0, 1]$. This value estimates the strength of the note q in frame k.

The chord transcription C consists of n_C chord changes. For each chord change k ($1 \leq k \leq n_C$), we are given a value $C[k].time$ that is the time when the chord changes. In addition, we are given a value $C[k].chord$ that is the name of the chord. We restrict ourselves to triads (major, minor, diminished and augmented chords that consist of three notes), which results in a total of 48 possible chords.

Finally, the outcome of the algorithm should be a melody transcription M that consists of n_M melody notes. For each melody note k ($1 \leq k \leq n_M$), the algorithm should produce values $M[k].time$ and $M[k].pitch$ that denote the onset time of the note and the pitch of the note.

Throughout the paper, we use MIDI note numbers to refer to the pitches. Thus, every pitch has a unique integer value and the interval of pitches a and b is $|a-b|$ semitones. Pitch C4 (261.6 Hz) is associated with MIDI value 60.

1.2 Related work

Automatic melody transcription has been studied actively during the last decade. Detailed reviews of the proposed methods can be found in [16] and [20].

The usual first step in automatic melody transcription is to detect potential melody notes in the audio signal. The most popular method for this is to calculate a salience function for the audio frames using the discrete Fourier transform or a similar technique (e.g. [2, 6, 15, 19]). Other proposed approaches for audio data processing include signal source separation [3] and audio frame classification [5].

After this, the final melody is selected according to some criterion. One technique for this is to construct a hidden Markov model (HMM) for note transitions and use the Viterbi algorithm for tracking the most probable melodic line [3, 5, 19]. An alternative to this is to use a set of local rules that describe typical properties of melody notes and outlier notes [7, 15, 20]. In addition, some systems [2, 6] feature agents that follow potential melody lines.

The idea of providing additional information to facilitate the melody transcription has also been considered in previous studies. A usual approach for this is to gather information from users. For example, users can determine which instruments are present [8], help in the source separation process [4] or create an initial version of the transcription [9]. The drawback of these systems is, of course, that the transcription is not fully automatic.

There are also some previous studies that combine key, chord and pitch estimation. In [18], the key and the chords of the piece are estimated simultaneously. Multiple pitch transcription systems that exploit key and chord information in pitch estimation include [1] and [17].

Most of the previous work on automatic melody transcription focuses on a slightly different problem than the topic of this paper, namely how to determine the melody frequency in the audio signal frame-by-frame. In [19] and [22], the output of the algorithm is similar to ours.

2. ALGORITHM

In this section we present our melody transcription algorithm that uses a chord transcription as a starting point for the transcription. The algorithm first divides the audio data into segments so that the boundaries of the segments correspond to the boundaries of the chords in the chord transcription. After this, the key of the piece is estimated using the chord transcription. Finally, the algorithm assigns each segment a pattern of notes that matches both the audio data and the chord transcription.

The input and the output of the algorithm are as described in Section 1.1. Thus, the algorithm is given n_A audio frames in array A and a chord transcription of n_C chord changes in array C, and the algorithm produces a melody transcription of n_M notes in array M.

2.1 Segmentation

The first step in the algorithm is to divide the audio data into segments. The segments will be processed separately in a later phase in the algorithm. The idea is to choose the boundaries of the segments so that the harmony in each segment is stable. This is accomplished using the chord boundaries in the chord transcription.

Let

$$l(k) = C[k+1].time - C[k].time$$

for each k where $1 \leq k \leq n_C - 1$ and

$$f(k, x) = (l(k)/x)/\lfloor l(k)/x \rfloor$$

where x is a real value. Thus, $l(k)$ is the length of the segment between chord changes k and $k+1$, and $f(k, x)$ is an estimate how evenly x divides that segment into smaller segments. Finally, let

$$g(k, x) = \begin{cases} 1 & \text{if } f(k, x) \leq 1 + \epsilon \\ 0 & \text{otherwise} \end{cases}$$

and

$$s(x) = \sum_{i=1}^{n_C - 1} g(i, x).$$

If $f(k, x) \leq 1 + \epsilon$ for some small ϵ, our interpretation is that x divides the segment evenly. Thus, $g(k, x)$ indicates if the segment is divided evenly, and $s(x)$ is the number of segments that are divided evenly if x was chosen. In this paper we use value $\epsilon = 0.1$.

The algorithm chooses a value of x such that x is in the range $[min_x, max_x]$ and $s(x)$ is as large as possible. The value x will be used as a unit length in the segmentation. The values min_x and max_x denote the minimum and maximum unit length; in this paper we use values $min_x = 0.5$ and $max_x = 3$.

Finally, the algorithm produces a segment division S of n_S segments by dividing each chord segment k into $l(k)/x$ smaller segments of equal length (the number of segments is rounded to the nearest integer). For each new segment u ($1 \leq u \leq n_S$) the algorithm assigns the values $S[u].begin$ and $S[u].end$ as described above, and $S[u].chord$ denotes the name of the chord in the segment.

2.2 Key estimation

After determining the segments, the algorithm estimates the key of the piece. The estimated key will be used later in the algorithm to favor melody notes that agree with the key. The key estimation is done using a simple method that is based on the chord information.

The algorithm goes through all segments in S and maintains a counter for each pitch class (a total of 12 counters). Initially, all the counters are zero. For each segment k, the algorithm increases the value of each counter that corresponds to $S[k].chord$. For example, if $S[k].chord$ is G major, the algorithm increases counters that correspond to notes G, B and D.

Finally, the algorithm determines the key using the counters as follows. There are 24 possible keys, 12 major keys and 12 minor keys. For each key, the algorithm calculates the sum of counters that correspond to tonic, mediant and dominant in that key. The key whose sum is the highest is selected as the key of the piece.

This method for key estimation is more simple than methods used in previous studies involving chord and key estimation from music audio [11,18]. However, this method produces results that are considered accurate enough for this purpose.

2.3 Pattern matching

For each segment, the algorithm chooses a melody pattern that matches both the audio data within the segment and the chord and key information. Each segment is processed independently, and the final melody transcription consists of all melody patterns in the segments.

The algorithm divides each segment into d note slots where d is a preselected constant for all segments. Each note slot can contain either one melody note or rest in the melody pattern. The idea is to select d so that most rhythms can be represented using d note slots, but at the same time d is small enough to restrict the number of melody notes. In practice, small integers that are divisible by 2 and/or 3 should be good choices for d.

An optimal melody pattern is selected according to a scoring function. The scoring function should give high scores for melody patterns that are probable choices for the segment. Depending on the scoring function, there are three ways to construct the optimal melody pattern:

- Greedy construction: If the melody slots are independent of one another, we can select the optimal melody note for each slot and combine the results to get the optimal melody pattern.
- Dynamic programming: If the melody slots are not independent but the score of a slot only depends on the previous slot, we can use dynamic programming to construct the optimal pattern.
- Complete search: If the score of a melody pattern cannot be calculated before all melody notes are selected, we have to go through all possible note patterns and select the optimal one.

The methods involving greedy construction and dynamic programming are efficient in all practical situations. However, in complete search we need to check q^d melody patterns where q is the number of possible choices for a melody slot. In practice, $q \approx 50$, so complete search can be used only when $d \leq 4$ to keep the algorithm efficient.

2.4 Scoring function

The scoring function that we use in this paper is primarily based on the key information and favors melody notes that match the estimated key of the excerpt. In addition, the notes that belong to the chord of the segment have an increased probability to be selected to the melody. Thus, if an E major chord occurs in the C major key, the note G# is a strong candidate for the melody note even if it does not belong to the C major scale.

Let $s(k, a, q)$ denote the score for an event where a'th note slot of segment k contains note q. We calculate the score using the formula

$$s(k, a, q) = z \cdot b(k, a, q) - x \cdot c(k, a, q)$$

where $b(k, a, q)$ is a base score calculated from the audio data, $c(k, a, q)$ is a penalty for selecting a note that does not appear in the audio data, and z and x are parameters that control the balance of the base score and the penalty.

Let I be a set that contains the indices of all audio frames that are inside the current note slot. Now we define

$$b(k, a, q) = \sum_{i \in I} A[i][q]$$

and

$$c(k, a, q) = \sum_{i \in I} e(i, q)$$

where

$$e(i, q) = \begin{cases} 1 & \text{if } A[i][q] = 0 \\ 0 & \text{otherwise.} \end{cases}$$

The parameter z favors melody notes that match the chord transcription, and it should depend on the key of the excerpt and the chord in segment k. We set $z = 2$ if q belongs to the current chord, $z = 1$ if q belongs to the key of the excerpt and otherwise $z = 0$. The parameter x controls the effect of adding a note to the melody that does not appear strongly in the audio data, and we study the effect of that constant in Section 3.

Finally, we select the note pattern greedily so that each note slot will be assigned the note that maximizes the score for that slot. If no note produces a score greater than 0, we leave that slot empty.

We also experimented with dynamic programming scoring functions that favor small intervals between consecutive notes, but the results remained almost unchanged. Consequently, we chose the more simple greedy construction approach.

3. EVALUATION

In this section we present results concerning the accuracy of the transcriptions produced by our algorithm, using real-world inputs. We evaluated our algorithm using a dataset

of Western popular music. We used both hand-made and automatic chord transcriptions as additional input for the algorithm.

Audio Melody Extraction task is an established part of the MIREX evaluation [13]. However, in the MIREX evaluation each audio frame is assigned a melody note frequency, and those results cannot be compared with our melody transcriptions that consist of note onset times and pitches in semitones.

3.1 Dataset

The evaluation dataset consists of 1,5 hours of audio excerpts from Western popular music. The length of each excerpt in the dataset is between 20 and 60 seconds. For each excerpt, we manually created time-aligned melody and chord transcriptions. We chose the excerpts so that the content of each excerpt is unique i.e. repetitions of verses and choruses are not included in the dataset.

The dataset can be found on our web site [1] , and is available for free for use in research. For each excerpt, the dataset includes an audio file in WAV format, and chord and melody transcriptions in text format. Each chord transcription is a list of chord change times and chord symbols, and each melody transcription is a list of note onset times and pitches. Thus, the transcriptions in the dataset correspond to the definitions in Section 1.1.

3.2 Evaluation method

To evaluate a melody transcription, we calculate two values: the precision and the recall. Precision is the ratio of the number of correct notes in the transcription and the total number of notes in the transcription. Recall is the ratio of the number of correct notes in the transcription and the total number of notes in the ground truth.

Let X be a melody transcription of n_X notes created by the algorithm, and let G be the corresponding melody transcription of n_G notes in the ground truth. Both transcriptions consists of a list of melody note onset times and pitches as described in Section 1.1.

To evaluate the precision and the recall of X, we first align the transcriptions. Let n_D be the maximum integer value such that we can create lists D_X and D_G as follows. List D_X consists of n_D note indices in X, and list D_G consists of n_D note indices in G. In addition, for each k $(1 \leq k \leq n_D)$ $X[D_X[k]].pitch = G[D_G[k]].pitch$ and $|X[D_X[k]].time - G[D_G[k]].time| \leq \alpha$ where α is a small contant. In other words, we require that lists D_X and D_G align a set of notes where all pitches match each other and the onset times of the notes do not differ more than α from each other.

Finally, let

$$\text{precision}(X, G) = n_D/n_X$$

and

$$\text{recall}(X, G) = n_D/n_G.$$

[1] http://cs.helsinki.fi/u/ahslaaks/fpds/

In practice, we calculate the value n_D efficiently using dynamic programming. The technique is similar to calculating the Levenshtein distance between two strings [14].

This evaluation method corresponds with that used in [19] and [22], however, the previous papers do not specify how the notes in the two transcriptions are aligned.

3.3 Experiments

We implemented our algorithm as described in Section 2. For calculating array A we used an algorithm by Salamon and Gómez that estimates potential melody contours in the audio signal. We used the Vamp plugin implementation of the algorithm (”all pitch contours”). Note that this algorithm already restricts the set of possible melodies considerably. We converted each pitch frequency to a MIDI note number assuming that the frequency of A4 is 440 Hz.

We used four chord transcriptions in the evaluation:

- A random chord transcription where the time between two chord changes is a random real number in the range $[0.5, 2]$ and each chord is randomly selected from the set of 48 possible triads.
- A simple automatic chord transcription created by our own algorithm. We used the standard technique of constructing a hidden Markov model and tracking the optimal path using the Viterbi algorithm [21].
- An advanced automatic chord transcription created using the Chordino tool [12].
- The chord transcription in the ground truth.

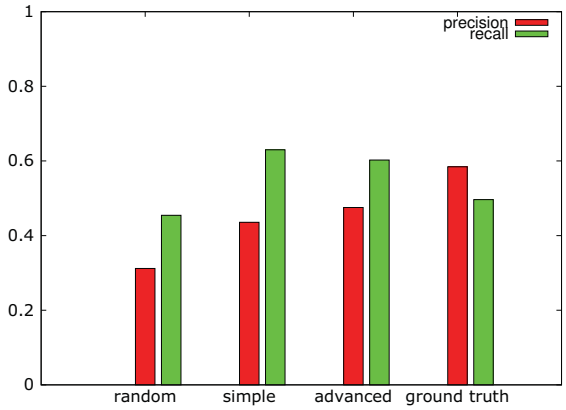
Random chord transcriptions were used in an effort to understand the actual role of the chord information and how the algorithm works if the chord information does not make sense at all.

Finally, there are three parameters that we varied during the evaluation:

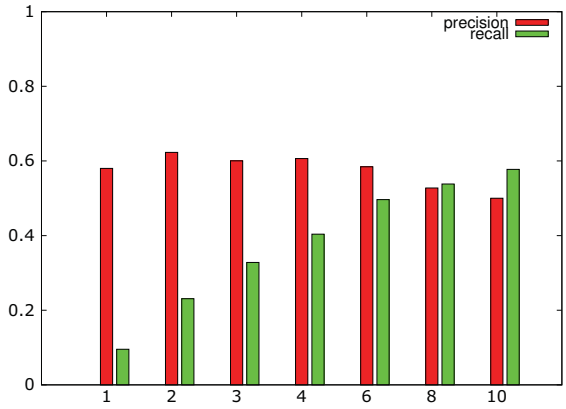
- d: the number of note slots in a segment as described in Section 2.3 (default value: 6),
- x: the cost for assigning a note to a frame without a note as described in Section 2.4 (default value: 1.00),
- α: the maximum onset time difference in the evaluation as described in Section 3.2 (default value: 0.25).

The default values were chosen so that they produce good results on the evaluation dataset.

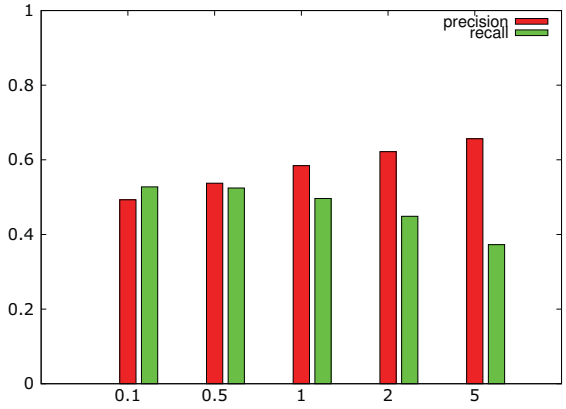
In each experiment in the evaluation, we varied one parameter and kept the remaining parameters unchanged. We used the ground truth chord transcription as the default chord transcription. We created melody transcriptions for all excerpts in the dataset and calculated average precision and recall values.



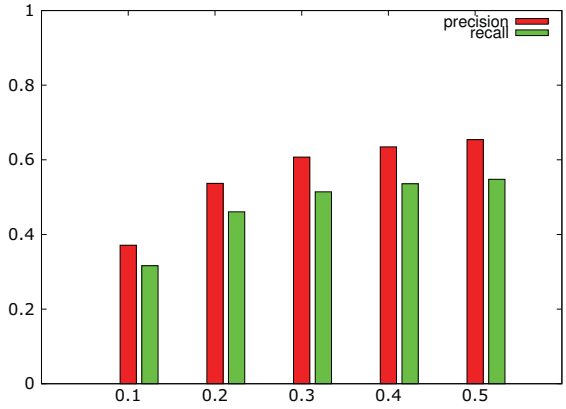
(a) The results using random chord transcription, simple automatic transcription, advanced automatic transcription, and ground truth transcription.



(b) The results varying the parameter d: the number of available note slots in a segment.



(c) The results varying the parameter x: the cost for assigning a note to a frame without a note.



(d) The results varying the parameter α: the maximum onset time difference in seconds in the evaluation.

Figure 2: The results of the experiment.

3.4 Results

In the first experiment (Figure 2a) we studied how the quality of the chord transcription affects the results. As expected, the better the chord transcription, the better the precision of the melody transcription. However, recall was highest when using automatic chord transcriptions. One possible reason for this is that there were more chord changes in automatic transcriptions than in the ground truth transcription. Therefore more melody notes were selected using the automatic chord transcriptions.

In the second experiment (Figure 2b) we varied the parameter d: the number of note slots in a segment. Our findings suggest that 6 note slots is a good trade-off between the precision and the recall. This can be explained by the fact that 6 is divisible by both 2 and 3, and thus segments of 6 note slots are suitable for both 3/4 time and 4/4 time music. Interestingly, when $d \leq 6$ the precision of the transcription remained nearly unchanged.

In the third experiment (Figure 2c) we varied the parameter x: the cost for assigning a note to a frame without a note. This was an important parameter, and the results were as expected. Increasing the parameter x improves the precision because melody notes are only selected if they appear strongly in the audio data. At the same time, this decreases the recall because fewer uncertain notes are included in the melody transcription.

Finally, in the fourth experiment (Figure 2d) we varied the parameter α: the maximum note onset time difference in the evaluation. Of course, the greater the parameter α, the better the results. Interestingly, after reaching a value of approximately 0.25, increasing α did not affect the results considerably. The probable reason for this is that if the melody note pitches in the transcription are not correct, the situation cannot be rescued by allowing more error for the onset times.

Previous studies also present some results about the precision and the accuracy of the algorithms. However, findings from these studies cannot be directly compared with the new results because the evaluation dataset is different in each study. In [19] precision 0.49 and recall 0.61 was reported using a database of 84 popular songs. In [22] the melody transcription was evaluated using a small set of 11 songs with precision 0.68 and recall 0.63.

4. CONCLUSIONS

In this paper we presented an automatic melody transcription algorithm that uses a chord transcription for selecting melodies that match the harmony of the music. We evaluated the algorithm using a collection of popular music excerpts, and the results of the evaluation suggest that the chord information can be successfully used in melody transcription of real-world inputs.

Our new evaluation dataset consists of 1,5 hours of audio excerpts of popular music together with melody and chord annotations. The dataset can be used at no cost for research purposes, for example as evaluation material for other chord transcription and melody transcription systems.

Our future work aims to use the harmony information provided by the chord transcription more extensively in melody transcription. Currently our algorithm uses only information about chord notes to constrain the pitches of melody notes, but using more advanced musical knowledge should yield better results.

5. ACKNOWLEDGEMENTS

This work has been supported by the Helsinki Doctoral Programme in Computer Science and the Academy of Finland (grant number 118653).

6. REFERENCES

[1] E. Benetos, A. Jansson and T. Weyde: "Improving automatic music transcription through key detection," *AES 53rd International Conference on Semantic Audio*, 2014.

[2] K. Dressler: "An auditory streaming approach for melody extraction from polyphonic music," *12th International Society for Music Information Retrieval Conference*, 19–24, 2011.

[3] J.-L. Durrieu et al: "Source/filter model for unsupervised main melody extraction from polyphonic audio signals," *IEEE Transactions on Audio, Speech, and Language Processing*, 18(3), 564–575, 2010.

[4] J.-L. Durrieu and J.-P. Thiran: "Musical audio source separation based on user-selected F0 track," *10th International Conference on Latent Variable Analysis and Signal Separation*, 2012.

[5] D. Ellis and G. Poliner: "Classification-based melody transcription," *Machine Learning*, 65(2–3), 439–456, 2006.

[6] M. Goto: "A real-time music scene description system: predominant-F0 estimation for detecting melody and bass lines in real-world audio signals," *Speech Communication*, 43(4), 311–329, 2004.

[7] S. Joo, S. Park, S. Jo and C. Yoo: "Melody extraction based on harmonic coded structure," *12th International Society for Music Information Retrieval Conference*, 227–232, 2011.

[8] H. Kirchhoff, S. Dixon and A. Klapuri: "Shift-variant non-negative matrix deconvolution for music transcription," *37th International Conference on Acoustics, Speech and Signal Processing*, 2012.

[9] A. Laaksonen: "Semi-automatic melody extraction using note onset time and pitch information from users," *SMC Sound and Music Computing Conference*, 689–694, 2013.

[10] M. Lagrange et al: "Normalized cuts for predominant melodic source separation," *IEEE Transactions on Audio, Speech, and Language Processing*, 16(2), 278–290, 2008.

[11] K. Lee and M. Slaney: "A unified system for chord transcription and key extraction using hidden Markov models," *8th International Conference on Music Information Retriavel*, 245–250, 2007

[12] M. Mauch and S. Dixon: "Approximate note transcription for the improved identification of difficult chords," *11th International Society for Music Information Retrieval Conference*, 135–140, 2010

[13] MIREX Wiki: Audio Melody Extraction task, http://www.music-ir.org/mirex/wiki/

[14] G. Navarro: "A guided tour to approximate string matching," *ACM Computing Surveys*, 33(1): 31–88, 2001

[15] R. Paiva, T. Mendes and A. Cardoso: "Melody detection in polyphonic musical signals: exploiting perceptual rules, note salience, and melodic smoothness," *Computer Music Journal*, 30(4), 80–98, 2006.

[16] G. Poliner et al: "Melody transcription from music audio: approaches and evaluation," *IEEE Transactions on Audio, Speech, and Language Processing*, 15(4), 1247–1256, 2007.

[17] S. Raczyński, E. Vincent, F. Bimbot and S. Sagayama: "Multiple pitch transcription using DBN-based musicological models," *11th International Society for Music Information Retrieval Conference*, 363–368, 2010

[18] T. Rocher et al: "Concurrent estimation of chords and keys from audio," *11th International Society for Music Information Retrieval Conference*, 141–146, 2010

[19] M. Ryynänen and A. Klapuri: "Automatic transcription of melody, bass line, and chords in polyphonic music," *Computer Music Journal*, 32(3), 72–86, 2008.

[20] J. Salamon and E. Gómez: "Melody extraction from polyphonic music signals using pitch contour characteristics," *IEEE Transactions on Audio, Speech, and Language Processing*, 20(6), 1759–1770, 2012.

[21] A. Sheh and D. Ellis: "Chord segmentation and recognition using EM-trained hidden Markov models," *4th International Conference on Music Information Retrieval*, 183–189, 2003

[22] J. Weil et al: "Automatic generation of lead sheets from polyphonic music signals," *10th International Society for Music Information Retrieval Conference*, 603–608, 2009

AUDIO-TO-SCORE ALIGNMENT AT NOTE LEVEL FOR ORCHESTRAL RECORDINGS

Marius Miron, Julio José Carabias-Orti, Jordi Janer
Music Technology Group, Universitat Pompeu Fabra
marius.miron,julio.carabias,jordi.janer@upf.edu

ABSTRACT

In this paper we propose an offline method for refining audio-to-score alignment at the note level in the context of orchestral recordings. State-of-the-art score alignment systems estimate note onsets with a low time resolution, and without detecting note offsets. For applications such as score-informed source separation we need a precise alignment at note level. Thus, we propose a novel method that refines alignment by determining the note onsets and offsets in complex orchestral mixtures by combining audio and image processing techniques. First, we introduce a note-wise pitch salience function that weighs the harmonic contribution according to the notes present in the score. Second, we perform image binarization and blob detection based on connectivity rules. Then, we pick the best combination of blobs, using dynamic programming. We finally obtain onset and offset times from the boundaries of the most salient blob. We evaluate our method on a dataset of Bach chorales, showing that the proposed approach can accurately estimate note onsets and offsets.

1. INTRODUCTION

Audio-to-score alignment concerns synchronizing the notes in a musical score with the corresponding audio rendition. An additional step, alignment at the note level, aims at adjusting the note onsets, in order to further minimize the error between the score and audio. In the context of orchestral music, this task is challenging; first, because of the complex polyphonies, and, second, because of the timing expressivity of classical music.

As possible applications of note alignment, deriving the exact locations of the note onsets and offsets could improve tasks as score-informed source separation [6], [2], [7].

State-of-the-art score alignment methods use Non-negative matrix factorization (NMF) [14], [11], template adaptation through expectation maximization [9], dynamic time warping (DTW) [3], and Hidden Markov Models (HMM) [4, 6]. The method described in [11, p. 103] is the only one addressing explicitly the topic of fine note alignment as a post-processing step. A factorization is performed to obtain the onsets of the anchor notes. The basis vectors are trained with piano pitches models, and the onsets are obtained from the activations matrix. Furthermore, an additional step is performed in order to look for onsets between anchors.

However, the methods listed above have certain limitations. First, accurately detecting the offset of the note is a challenging problem and none of these methods claim to solve it. Second, the scope of the NMF-based systems is solely piano recordings. Third, except [11], the algorithms consider a large window to evaluate detected onsets. Note that the MIREX Real-time Audio-to-Score Alignment task considers a 2000 ms window size.

With respect to image processing techniques deployed in music information research, a system to link audio and scores for makam music is presented in [13]. In this case, Hough transform is used for picking the line corresponding to the most likely path from a binarized distance matrix. Additionally, the same transform is used in [1] to find repeating patterns for audio thumbnailing.

In this paper we propose a novel method for audio-to-score alignment at the note level, which combines audio and image processing techniques. In comparison to classical audio-to-score alignment methods, we aim to detect the offset of the note, along with its onset. Additionally, we do not assume a constant delay between score and audio, thus we do not use any information regarding the beats, tempo or note duration, in order to adjust the onsets. Therefore, our method can align notes when dealing with variable delays, as the ones resulting from automatic score alignment or the ones yielded by manually aligning the score at the beat level.

The proposed method is based on two stages. First, the audio processing stage involves filtering the spectral peaks in time and frequency for every note. Consequently, the filtering occurs in the time interval restricted for each note and in the frequency bands of the harmonic partials corresponding to its fundamental frequency. Furthermore, we decrease the magnitudes of the peaks which are overlapping in time and frequency with the peaks from other notes. Using the filtered spectral peaks, we compute the pitch salience for each note using the harmonic summation algorithm described in [10]. Second, we detect the boundaries of the note using an image processing algorithm. The pitch salience matrix associated to each note is binarized. Then, blobs, namely boundaries and shapes, are detected using

the connectivity rules described in [12, p. 248]. From all the blobs candidates associated to every note, we pick the best combination of consecutive blobs using dynamic programming. The image processing part has the advantage that the blob boundaries will define the note onsets along with the corresponding offsets.

The remainder of this paper is structured as follows. In the first section we describe the note-wise pitch salience computation, followed by the blob selection using image processing methods. Then, we evaluate our algorithm on a dataset of Bach chorales [6] and we discuss the results.

2. METHOD

The proposed method aims to detect the onsets and offsets of the notes from a monaural audio recording, where the score is assumed to be automatically or manually aligned a priori, assuming an error up to 200 ms.

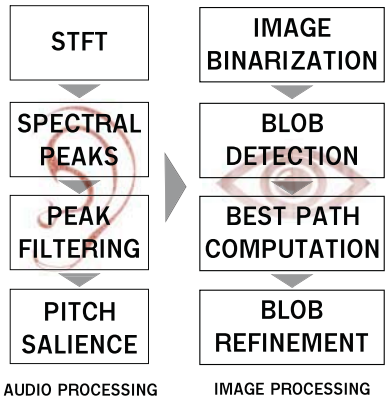


Figure 1. *The two main sections of our method: audio and image processing, and the corresponding steps.*

Figure 2 shows the block diagram of the proposed method. As can be seen, the method is subdivided in two stages. First, in the audio processing stage, a filtered pitch salience matrix is obtained for each of the notes in the score, and for every instrument. Second, in the image processing stage, the pitch salience matrix is regarded as a greyscale image, and blobs are detected in the binarized image. Moreover, we construct a graph with all the blobs and we pick the best combination of blobs by using Dijkstra's algorithm to find the best path in the graph. Finally, we refine the time boundaries for the blobs that overlap, using an adaptive threshold binarization.

2.1 Note-wise pitch salience computation

For each input signal, we first compute the Short time Fourier transform (STFT) and we extract the spectral peaks. Then, we analyze each single note in the score and we select only the spectral peaks in the frames around its approximate time location and the frequency bands associated to its harmonic partials (i.e. multiples of the fundamental frequency). Finally, we compute the pitch salience, using the harmonic summation algorithm described in [10].

To select the time intervals at which we are going to look for the note onsets and offsets, we analyze the pre-aligned score that we want to refine. We start from the assumption that the note onsets are played with an error lower than 200 ms from the actual onset in the score. In other words, we set the search interval to ± 200 ms from the note onset at the score. Additionally, in the case of the offset, we extend the possible duration of a note in the score by 200 ms or until another note in the score appears. In the rest of the paper, this search interval will be referred to as $T_{on}(n)$ and $T_{off}(n)$.

Then, we analyze the spectral peaks within the time interval defined for each note, and we filter them according to the harmonic frequencies of the MIDI note $\hat{F}_n(i)$, where $\hat{F}_n(0)$ is the fundamental frequency of note n. Namely, we take the first 16 of the harmonic partials of this frequency, $\hat{F}_n(i)$ with $i \in [0, ..., 15]$. Taking into account vibratos, we set a 1.4 semitone interval around each of the harmonic partials. Consequently, we select a set of candidate peaks $P_n(k)$ and the associated amplitudes $A_n(k)$ for note n at frame k such that $P_n(k) \in [\hat{F}_n(i) - \hat{L}_n(i), ..., \hat{F}_n(i) + \hat{L}_n(i)]$, where $\hat{L}_n(i)$ is a frequency band equivalent to 0.7 of a semitone.

As a drawback, some of the selected peaks could overlap in time and frequency. To overcome this problem, we distribute the amplitude $A_n(k)$ of the overlapped peaks $P_n(k)$ using a factor $g_i(P_n(k), P_m(k))$, where n and m are the overlapped notes, g_i is a gaussian centered at the corresponding frequency $\hat{F}_n(i)$ of the note n and the harmonic partial i. The standard deviation equals to $\frac{\hat{L}_n(i)}{2}$, thus:

$$g_i(x) = w * 0.8^i * e^{-\left(x-\hat{F}_n(i)\right)^2 \Big/ \frac{\hat{L}_n(i)}{2}^2} \quad (1)$$

Note that the magnitude of the gaussian decreases with the order of the harmonic, i, and is proportional to w, the weight of the rest of the instruments in current audio file, or the coefficient extracted from a pre-existing mixing matrix. For example, if we align using solely a monaural signal in which all four instruments have the same weight, 0.25 for all four instruments, the coefficient will be $w = 0.75$.

The factor g_i penalizes frequencies which are in the allowed bands but are further away from the central frequencies. In this way, we eliminate transitions to other notes or energy which can add up noise later on in the blob detection stage.

Finally, for each note n and its associated $P_n(k)$ and $A_n(k)$ where $k \in [T_{on}(n), ...T_{off}(n)]$, we use the pitch salience function described in [10]. The algorithm calculates a salience measure for each pitch candidate, starting at $\hat{F}_n(0) - \hat{L}_n(0)$, based on the presence of its harmonics and sub-harmonics partials, and the corresponding magnitudes. Finally, the salience function for each time window is quantized into cent bins, thus the resulting matrix S_n has the dimensions $(T_{off}(n) - T_{on}(n), Q)$, where Q is the number of frequency bins for the six octaves. In our case, we experimentally choose $Q = 600$ bins.

2.2 Blob selection using image processing

The goals of the image processing stage are to obtain the location of the note onset and offset by binarizing the note-wise pitch salience, and to detect shapes and contours in the binarized image.

Accounting that the image binarization is not a robust process [12], different results are expected as a function of the amount of time overlap between notes, the salience of the pitch and its fundamental frequency. Therefore, as the shape and contour detection heavily relies on this step, we need a robust binarization, which would finally give us the best information for detecting the boundaries of the note.

Previous approaches to improve binarization rely on background subtraction or local binarization [12]. Therefore, we propose a binarization method similar to the local binarization, but adapted to our context: the pitch salience matrix. On the assumption that the bins closer to the fundamental frequency, $\hat{F}_n(0)$, are more salient than the ones at higher frequencies, we split the binarization areas in sub-areas related to the harmonic partials $\hat{F}_n(i)$. Thus, the salience matrix $\mathbf{S}_n$ is binarized gradually and locally, obtaining a binary matrix $\mathbf{B}_n$. Moreover, we consider l as the binarization step, moving gradually from 50 to 600 in steps of 50 bins.

Furthermore, we compute $\mathbf{B}_n$ in l steps, each time only for the columns in the interval $[l-50...l]$.

$$\mathbf{B}_n(i,j) = \begin{cases} 0, \mathbf{S}_n(i,j) < mean(\mathbf{S}_n^l) \\ 1, \mathbf{S}_n(i,j) \geq mean(\mathbf{S}_n^l) \end{cases} \quad (2)$$

where $i \in [T_{on}(n), ..., T_{off}(n)]$, $j \in [l-50...l]$, and $\mathbf{S}_n^l$ is a submatrix of $\mathbf{S}_n$, obtained by extracting the columns of $\mathbf{S}_n$ in the interval [0..l].

As an example, a pitch salience matrix $\mathbf{S}_n$ for a bassoon note is plotted in the Figure 2A. The green rectangles mark the submatrices $\mathbf{S}_n^l$ for various values of l. The resulting binarized image is depicted in Figure 2B.

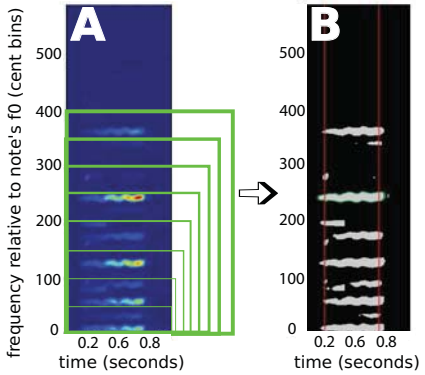


Figure 2. *Binarizing the spectral salience matrix (figure A) and detecting the blobs in the resulting image (figure B). Binarization is done gradually and locally, relative to the green squaresáreas in figure A. The ground truth onset and offset of the note are marked by vertical red lines.*

The next step is detecting boundaries and shapes on the binarized image. We use the connectivity rules described in [12, p. 248] in order to detect regions and the boundaries of these regions, namely the blobs. Thus, we want to label each pixel of the matrix $\mathbf{B}_n$ with a number from 0 to r, where r is the total number of detected blobs.

Having a pixel (i,j) with $i \in [T_{on}(n), ..., T_{off}(n)]$ and $j \in [0, ..., Q]$, where Q is the number of frequency bins, we need to consider all the neighboring pixels and we have to take into account their connectivity with the current pixel. The 4-way connectivity rules account for the immediate neighbors, as compared to 8-way connectivity which account for all the surrounding pixels. Because we are not interested in modeling transitions between notes, we discard diagonal shapes by using the 4-way connectivity rules. Hence, the connectivity matrix, which determines the neighborhood of the pixel (i,j), can be written as:

$$\mathbf{M} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

For the matrix $\mathbf{M}$, the central pixel with the coordinates (2,2) represents the origin pixel (i,j), and all the other non-zero pixels are the considered positions for the neighbors.

The algorithm, described in [12, p. 251], takes one pixel at a time and visits its non-zero neighbors. Then, we move sequentially from one pixel to its neighbors, setting boundaries for the pixels having neighbors equal to zero. Finally, the shape is enclosed when the algorithm reaches the pixel of origin.

Furthermore, once we have detected a set of blobs b_n for each note n, we need to compute the best combination of the blobs for all notes. Because search intervals for consecutive notes can overlap in time, choosing the best combination of blobs is not as trivial as picking the best blob in terms of area or salience, and the decisions that we take for a current note, should take into account the decisions we take for the previous and the next note. This kind of problem, which chains up a set of decisions can be solved with dynamic programming.

Consequently, we consider the blobs to be the vertices of an oriented graph, in which the edges are assigned a cost depending on the area of the two blobs and the overlapping between them, as seen in Figure 3. Basically, blobs with bigger area and little overlapping will have a lower cost, which makes them ideal candidates when we find the best path in the graph. Additionally, we can have an edge only between blobs of consecutive notes, and we can remove the edges between blobs which overlap more than 50% in time.

Therefore, we compute the area of each blob of the note n by summing up the values in the binarized matrix $\mathbf{B}_n$, enclosed by the corresponding blob contours. Additionally, we exclude the blobs which have the duration less than 100 ms, and the ones starting after the allowed interval for the attack time.

The normalized area of blob i for the note n is $H(b_n^i)$ and is a value inversely proportional with the actual area, because we want the larger blobs to have a lower cost,

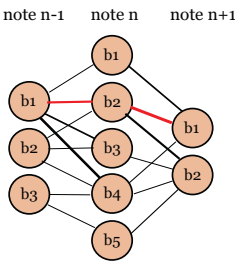


Figure 3. *A sample of the graph between three consecutive notes.* $b_{[1..5]}$ *are the blobs detected for each note. Thicker lines represent lower costs. The red line represents the best path in the graph.*

when picking the best path. In the same manner, we must increase the cost as the overlapping between the blobs increases. Thus, for two adjacent notes n and $n+1$, $O(b_n^i, b_{n+1}^i)$ has cost 1 if there is no overlapping, and an increased value summing up the ratio of the the area of the two overlapping blobs. For instance, if 20% of the area of the first blobs overlaps with 70% of the area of the second blob, $O = 1 + 0.2 + 0.7 = 1.9$.

Thus, the cost for the edges has the expression

$$cost(b_i^n, b_{i+1}^n) = O(b_n^i, b_{n+1}^i) * (H(b_n^i) + H(b_{n+1}^i))$$

In order to find the shortest path between the vertices of the first note in the score and the last one, we use Dijkstra's algorithm described in [5]. The algorithm finds the shortest path for a graph with non-negative edges by assigning a tentative distance to each of the vertices and progressively advancing by visiting the neighboring nodes.

Additionally, after the best path is computed, we can face the situation where two consecutive blobs overlap in time due to the inaccuracy in binarization and the fact that the minimum cost path does not guarantee no overlapping. Because the melody for a particular instrument is considered to be monophonic, we do not allow overlapping between two consecutive notes. Thus, we ought to find a splitting point between the starting point of the blob associated with the next note and the ending point of the blob associated with the current note.

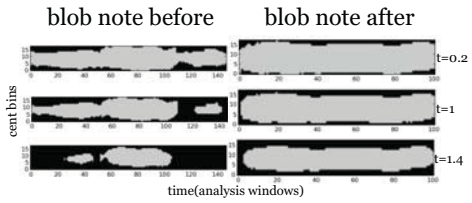


Figure 4. *Blob refinement using adaptive threshold binarization of two consecutive overlapping blobs in the best path. The minimum overlapping is achieved for threshold* $t = 1.4$

Having two consecutive blobs from the best path, b_n and b_{n+1}, we take the image patches surrounding their boundaries and we adaptively increase the threshold of binarization until the minimum overlapping is achieved. Consequently, we consider the submatrices $\hat{\mathbf{S}}_n$ and $\hat{\mathbf{S}}_{n+1}$ of the corresponding pitch salience matrices $\mathbf{S}_n$ and $\mathbf{S}_{n+1}$, and for a variable threshold $t = [0.2..2]$, we compute the binary matrices $\hat{\mathbf{B}}_n^t$ and $\hat{\mathbf{B}}_{n+1}^t$.

$$\hat{\mathbf{B}}_n^t(i,j) = \begin{cases} 0, \hat{\mathbf{S}}_n(i,j) < t * mean(\hat{\mathbf{S}}_n) \\ 1, \hat{\mathbf{S}}_n(i,j) \geq t * mean(\hat{\mathbf{S}}_n) \end{cases} \quad (3)$$

As seen in Figure 4, the higher the threshold t, the less pixels are be assigned to value 1 in the binary matrices, thus we increase the threshold gradually until no overlapping is achieved.

Finally, the note onset and offset are extracted from the leftmost and the rightmost pixels of the refined blobs in the best path.

3. EVALUATION

3.1 Experimental setup

The dataset used to evaluate our proposal consists of 10 human played J.S. Bach four-part chorales, and is commonly known as Bach10 . The audio files are sampled from real music performances recorded at 44.1 kHz that are 30 seconds in length per file. Each piece is performed by a quartet of instruments: violin, clarinet, tenor saxophone and bassoon. Each musician's part was recorded in isolation. Individual lines were then mixed to create 10 performances with four-part polyphony. More information about this dataset can be found in [6].

We observe that the dataset has a few particularities. First, every recording presents fermatas, where the final duration of the note is left at the discretion of the performer or the conductor, making it more difficult to detect the onset and offsets of the notes. Second, the chorales have a peculiar homophonic texture. Third, the annotated note onsets and offsets in the ground truth can have more or less notes than the actual score. We discovered that this mismatch comes from repeating notes, which in the original score are represented by a single larger note. This step also makes the detection of the note offsets more difficult.

In order to perform alignment at the note level, we generate a misaligned score by introducing onset and offset time deviations for all the notes and all the instruments in the ground-truth score. The deviations are randomly and uniformly distributed in the intervals $[-200, ..., -100]$ and $[100, ..., 200]$ ms. Moreover, we aim at refining the alignment of the algorithm proposed by [3]. Thus, we correct the onset times and we detect the offsets around the beginning of the next note. For both of these tasks we consider the interval $[-200, ..., 200]$ ms.

Furthermore, the STFT is computed using a Blackman-Harris 92dB window with a size of 128 ms and, a hop size of 6 ms. Additionally, we zero-pad the window by three times its length. Moreover, frequencies and magnitudes of the spectral peaks are extracted with the algorithm described in [8], which uses parabolic interpolation to accurately detect positive slopes in the spectrum computed at the previous step.

3.2 Results

We aim at correctly aligning the onsets and offsets of the misaligned score described in Section 3.1 and we add up 200 ms before and after the note boundaries in order to search for the exact starting and ending point of the note. Thus, our algorithm can have up to 400 ms in error for the onsets, and a larger error for the offset, because we are not constraining the duration of the note to any interval.

For each piece, aligned rate (AR) or precision is defined as the proportion of correctly aligned notes in the score and ranges from 0 to 1. A note is said to be correctly aligned if its onset does not deviate more than a threshold from the reference alignment. To test the reliability of our method, we tried different threshold values ranging form 15 to 140 ms. Other measures as the average offset (i.e. average absolute-valued time offset between a reported note onset by the score follower and its real onset in the reference file) and the std offset (i.e. standard deviation of sign-valued time offset) are also considered.

As illustrated in figure 5, the proposed system is able to accurately align more than the 30% of the onsets with a detection threshold lower than 15 ms. Furthermore, more than 80% of the onsets are accurately detected with a threshold of 60 ms. Because the search time interval for the note allows for error larger than 200 ms, the AR for the onset does not reach 100% in $t = 200ms$, as less than 2% of the onsets have larger errors.

Furthermore observe that we less accurate in detecting the offsets, particularly when we do not know the approximate note offset and we estimate it around the onset of the next note, as when we take as input the alignment of the algorithm proposed by [3]. The drop in performance of the offset detection can also be explained by the fact that the energy of a note can decay below a threshold, thus excluding it when binarization is performed.

Figure 6 shows boxplots of the average offset and the std error for each instrument, and for the note onset and offset, for the misaligned dataset. The lower and upper lines of each box show 25th and 75th percentiles of the sample. The line in the middle of each box is the average offset. The lines extending above and below each box show the extent of the rest of the samples, excluding outliers. Outliers are defined as points over 1.5 times the interquartile range from the sample median and are shown as crosses.

We observe that performance is lower for violin compared to the other instrument. This can be explained by the fact that for this dataset the violin has noisier or soft attacks, which do not yield a high enough value in terms of pitch salience, and is lost when binarizing the image.

Moreover, the fact that we are able to detect most of the onsets in the interval 0.06 seconds, which is an acceptable interval for the attack of the instruments aligned, point us on some limitation on using the pitch salience function, which is not able to be accurate enough with noisier attacks, as it happened for the violin.

Furthermore, we want more insight on how the errors are distributed across the time range. Thus, we plot the 2-d histogram of the onset errors, as seen in Figure 7. We

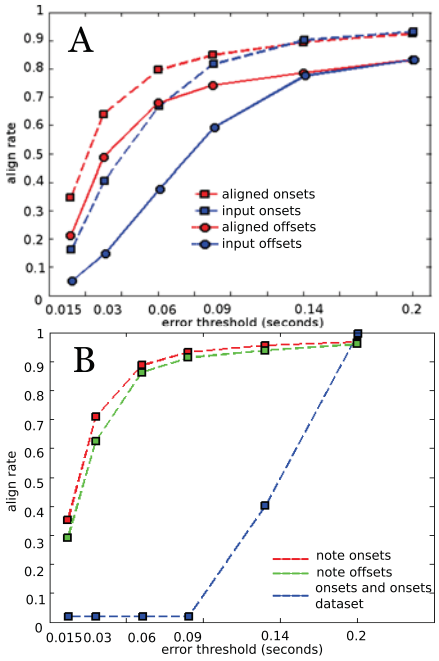


Figure 5. *The proposed system improves the align rate of (A) the system proposed by [3] and of (B) the misaligned dataset, for onset errors, as well as offset errors*

observe that even though the original dataset had large errors, our method was able to detect the note onsets within a small time frame, as most of the errors are in the bin centered at zero.

Moreover, our method is better at fixing the delays in the note onsets. In comparison, we can commit more errors if the onset of the note is thought to be before the actual onset, because the window in which we have to look for it overlaps more with the previous note, hence we have more interference.

Additionally, for every note and every instrument, we compute the percentage of correctly detected frames with respect to ground truth. Our algorithm is able to correctly detect 89% of the frames of the ground truth notes. In comparison, the notes in the misaligned dataset have a degree of 66% correctly detected frames.

Finally, we compute the percentage of frames which are erroneously detected as part of the notes. We observe that solely 0.07% of frames from the notes we refine are outside the boundaries of the ground truth notes, compared to the misaligned dataset, for which 34% of the frames are displaced outside the time boundaries of the notes.

Therefore, our algorithm is more likely to shorten the notes, rather than making erroneous decisions regarding their time frame. This is due to the way we are picking the best sequence of blobs, which penalizes the overlapping, thus picking blobs which have a smaller area but less overlapping with the blobs from neighboring notes.

4. CONCLUSIONS

We proposed a method to refine the alignment of onsets and offsets in orchestral recordings, using audio and im-

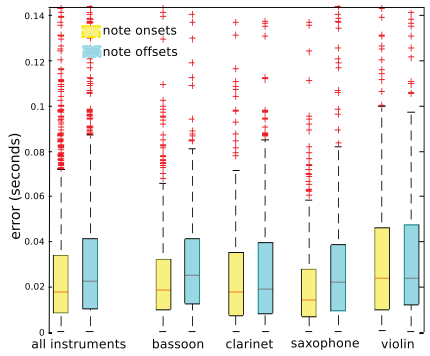


Figure 6. *The average offset and the std offset in terms of 25th and 75th percentile of the proposed system for bassoon, clarinet saxophone, and violin, for note onsets, as well as note offsets*

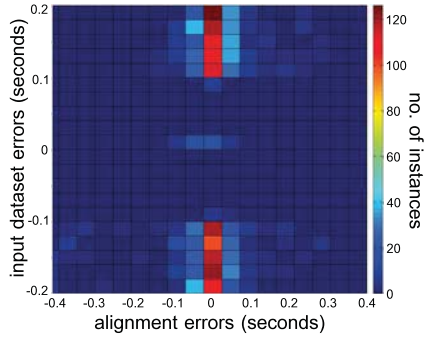


Figure 7. *The histogram of error distribution in the onset alignment*

age processing techniques. We compute a note-wise pitch salience function and we binarize it. Moreover, we detect blobs in the binarized image, and we pick the best blob candidate for each note by finding the best path in the associated graph. Furthermore, as offset detection is regarded as a more difficult problem, the proposed method addresses this issue by detecting image blobs to simultaneously label note onsets and offsets.

The evaluation shows that our method is able to refine the alignment in a misaligned dataset, having detected more than 80% of the onsets with an error of 60 ms. Moreover, we analyzed the performance across all four instruments, and we discovered that the accuracy drops for a violin, as being higher for the other instruments. Thus, as a future step, we need to analyze what limitation has the algorithm regarding certain instrument classes. Additionally, the proposed method should be tested with another dataset, with more complex polyphonies and tempo variations.

Furthermore, our method can be improved by using timbre models when filtering the spectral peaks and decreasing their magnitude. Additionally, choosing the best sequence of blobs can be improved by using a better cost function for the Dijkstra's algorithm. In addition, one could use image processing with other data obtained by audio processing means, as the spectrogram or come with a more robust approach than the pitch salience which does not capture noisy note attacks or noisy spectrum.

Finally, the note refinement can be used to improve the performance of score informed source separation, in the situation where the score is not well aligned with the audio.

5. ACKNOWLEDGEMENTS

This work was supported by the European Commission, FP7 (Seventh Framework Programme), STREP project, ICT-2011.8.2 ICT for access to cultural resources, grant agreement No 601166. Phenicx Project

6. REFERENCES

[1] J.-J. Aucouturier and M. Sandler. Finding repeating patterns in acoustic musical signals. *VIRTUAL, SYNTHETIC, AND ENTERTAINMENT AUDIO*, pages 412–421, 2002.

[2] J.J. Bosch, K. Kondo, R. Marxer, and J. Janer. Score-informed and timbre independent lead instrument separation in real-world scenarios. In *Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European*, pages 2417–2421, Aug 2012.

[3] J.J. Carabias-Orti, P. Vera-Candeas, F.J. Rodriguez-Serrano, and F.J. Canadas-Quesada. A RealTime Audio to Score Alignment System using Spectral Factorization and Online Time Warping. *IEEE Transactions on Multimedia(submitted)*, 2014.

[4] A. Cont. A coupled duration-focused architecture for real-time music-to-score alignment. *Pattern Anal. Mach. Intell. IEEE ...*, 32:974–987, 2010.

[5] E. W. Dijkstra. A note on two problems in connexion with graphs. *NUMERISCHE MATHEMATIK*, 1(1):269–271, 1959.

[6] Z. Duan and B. Pardo. Soundprism: An online system for score-informed source separation of music audio. *Selected Topics in Signal Processing, IEEE ...*, pages 1–12, 2011.

[7] S. Ewert and M. Muller. Using score-informed constraints for NMF-based source separation. *Acoustics, Speech and Signal Processing (...*, 2012.

[8] J. O. Smith Iii and X. Serra. Parshl: An analysis/synthesis program for non-harmonic sounds based on a sinusoidal representation . 1987.

[9] C. Joder and B. Schuller. Off-line refinement of audio-to-score alignment by observation template adaptation. *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 206–210, 2013.

[10] A. Klapuri. Multiple fundamental frequency estimation by summing harmonic amplitudes. In *in ISMIR*, pages 216–221, 2006.

[11] B. Niedermayer. *Accurate Audio-to-Score Alignment Data Acquisition in the Context of Computational Musicology*. PhD thesis, Johannes Kepler Universität, 2012.

[12] M. Nixon. *Feature Extraction and Image Processing*. Elsevier Science, 2002.

[13] S. Senturk, A. Holzapfel, and X. Serra. Linking Scores and Audio Recordings in Makam Music of Turkey. *Journal of New Music Research*, pages 35–53, 2014.

[14] T.M. Wang, P.Y. Tsai, and A.W.Y. Su. Score-informed pitch-wise alignment using score-driven non-negative matrix factorization. In *Audio, Language and Image Processing (ICALIP), 2012 International Conference on*, pages 206–211, July 2012.

A COMPOSITIONAL HIERARCHICAL MODEL FOR MUSIC INFORMATION RETRIEVAL

Matevž Pesek
University of Ljubljana
Faculty of computer
and information science
matevz.pesek@fri.uni-lj.si

Aleš Leonardis
Centre for Computational
Neuroscience and Cognitive Robotics
School of Computer Science
University of Birmingham
ales.leonardis@fri.uni-lj.si

Matija Marolt
University of Ljubljana
Faculty of computer
and information science
matija.marolt@fri.uni-lj.si

ABSTRACT

This paper presents a biologically-inspired compositional hierarchical model for MIR. The model can be treated as a deep learning model, and poses an alternative to deep architectures based on neural networks. Its main features are generativeness and transparency that allow clear insight into concepts learned from the input music signals. The model consists of multiple layers, each is composed of a number of parts. The hierarchical nature of the model corresponds well with the hierarchical structures in music. Parts in lower layers correspond to low-level concepts (e.g. tone partials), while parts in higher layers combine lower-level representations into more complex concepts (tones, chords). The layers are unsupervisedly learned one-by-one from music signals. Parts in each layer are compositions of parts from previous layers based on statistical co-occurrences as the driving force of the learning process. We present the model's structure and compare it to other deep architectures. A preliminary evaluation of the model's usefulness for automated chord estimation and multiple fundamental frequency estimation tasks is provided. Additionally, we show how the model can be extended to event-based music processing, which is our final goal.

1. INTRODUCTION

The field of music information retrieval (MIR) has reached a significant expansion in tasks and solutions in the short timespan of its existence [3, 10]. The tasks include extraction of high-level music descriptors from music, such as melody, chords and rhythm, as well as highly perceptual tasks involving mood estimation, genre recognition and artist influence. Solutions have not come to a perfect one for any of the described tasks yet; however, numerous approaches proposed each year are improving the state-of-the-art rapidly. Recently, deep belief networks as an alternative single model for a variety of tasks, have been successfully introduced to the field.

This paper presents a biologically-inspired compositional hierarchical model for music information retrieval. The proposed model poses an alternative to recent deep learning architecture approaches [6, 9]. Its main difference from the latter is in its transparent structure, thus allowing representation and interpretation of the signal's information extracted on different levels. We show the usefulness of our proposed approach in a preliminary evaluation of the model for the tasks of automated chord estimation and multiple fundamental frequency estimation. We also show how the model can be extended to event-based music processing, and point out how the model's transparency enables other applications of the model, e.g. for music analysis, synthesis and visualization.

2. DEEP ARCHITECTURES FOR MIR

The concept of deep learning has grown in popularity in the fields of signal processing [15], audio processing [9] and MIR. Lee [7] presented one of the first attempts of using deep belief networks (DBNs) on audio signals, where convolutional DBNs were applied to the speaker identification task. A DBN was used as a feature extractor, and a support vector machine for classification.

Later, Hamel and Eck [5], evaluated DBNs for genre recognition using a five-layer DBN with three hidden layers for feature extraction. The support vector machine was used for classification, where as raw spectral data was used as input to the DBN. DBNs show great potential for many tasks that involve high-level feature extraction, such as emotion recognition, since there is usually no trivial spectral or temporal feature that could be used to model the high-level representation in question. Schmidt and Kim [13] showed promising results by using a 5-layer DBN for extraction of emotion-based acoustic features. Other approaches modeled temporal aspects of the audio signal. Conditional DBNs were used by Battenberg and Wessel [1] for drum pattern analysis. Schmidt [12] took a step further and showed that DBNs can be trained for discriminating rhythm and melody.

Overall, recent research has shown great interest and

success in using features learned from music signals, in contrast to previously used hand-crafted features. The research reviewed in this subsection took place only in the last few years; thus, there is a vast expansion of deep learning in MIR to be expected, as anticipated by Humphrey [6].

3. THE COMPOSITIONAL HIERARCHICAL MODEL

3.1 Motivation and concept

DBNs brought an improvement to many MIR tasks with their unsupervised learning of features and generative modeling. However, they require a large set of hidden units per layer, and consequently, large training sets. Also, the hidden nature of units offers no clear explanation of the undergoing feature extraction process and the meaning of extracted features. It is our goal to overcome these limitations by developing a white-box compositional hierarchical model with shareable parts, thus reducing the number of parts and learning data needed, as well as reaching transparency in terms of interpretable internal structure of the model.

The proposed model provides a hierarchical representation of the audio signal, from the signal components on the lowest level, up to individual musical events on the highest levels. It is built on the assumption that a complex signal can be decomposed into a hierarchy of building blocks - *parts*. These parts exist at various levels of granularity and represent sets of entities describing the signal. According to their complexity, parts can be structured across several layers from less to the more complex. Parts on higher layers are expressed as compositions of parts on lower layers (e.g.: a chord is composed of several pitches, each pitch of several harmonics etc.). A part can therefore describe individual frequencies in a signal, their combinations, as well as pitches, chords and temporal patterns, such as chord progressions.

The structure of our model is inspired by work in computer vision, specifically the hierarchical compositional model presented by Leonardis and Fidler [8]. Their model represents objects in images in a hierarchical manner, structured in layers from simple to complex image parts. The model is learned from the statistics of natural images and can be employed as a robust statistical engine for object categorization and other computer vision tasks. We believe that such approach can also be used for music representation and analysis, however the transformation of the model to a different domain is not trivial.

3.2 Model structure

The compositional hierarchical model consists of several layers. Each layer contains a set of parts. A part is a composition of two or more parts from a lower layer and may itself be part of any number of compositions on a higher layer. Thus, the compositional model forms a hierarchy of parts, where each part represents a composition of lower-layer parts, as seen in Figure 1. Connections in the figure represent compositions of parts.

3.2.1 Input layer

The input layer of the model is derived from the time-frequency representation of the music signal. We denote this layer as layer $\mathcal{L}_0$. It contains a single atomic part, which is activated (produces output) at locations of all frequency components in the signal at a given time instance. An example is given in Figure 1, although not all activations are shown for clarity. More formally, a part's activation is defined by two values: location L_P that corresponds to frequency, and magnitude A_P, that corresponds to magnitude of the frequency component.

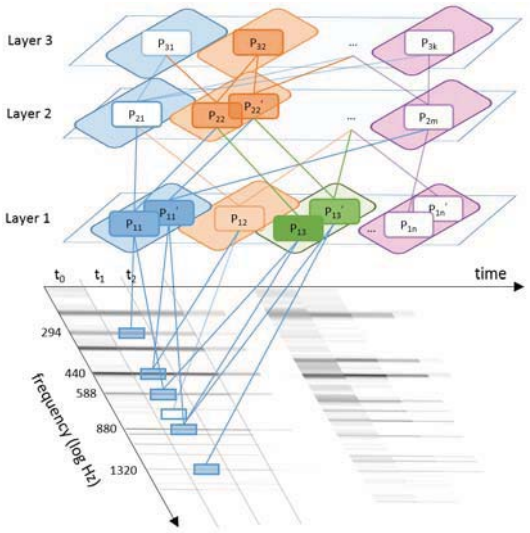


Figure 1. Compositional hierarchical model. Parts on the input layer correspond to signal components in the time-frequency representation. Parts on higher layers are compositions of lower-layer parts (denoted as links in the figure). A part may be contained in several compositions, e.g. P_{11} on the first layer is part of compositions P_{21}, P_{22} and P_{2m} on the second layer. Several depictions of the same part (e.g. part instances P_{11} and P'_{11}) denote several activations of the part on different locations (all instances of a part on a layer are marked with the same outlined color). Parts activated in t_1 are shown filled with color.

Any time-frequency representation can be used for the input layer, although logarithmic frequency spacing produces more compact models due to the relative nature of part compositions on higher layers (as described further on).

3.2.2 Subsequent layers

Higher layers of the model $\mathcal{L}_n$ contain sets of *compositions* - parts composed of parts from lower layers. Each composition can contain any number of parts from the lower layers (for clarity we only use two-part compositions to explain the model). A composition can be part of any number of compositions on higher layers. Compositions are denoted as links between parts in Figure 1.

Composition i on layer $\mathcal{L}_n$ can be formally defined as a structure containing parts from a layer below: a central part C, and a secondary part S. We name the parts forming

a composition subparts. A composition can be defined as:

$$P_{n,i} = \{C_{n-1,j}, S_{n-1,k}, (\mu_{n,i}, \sigma_{n,i})\}, \quad (1)$$

where $C_{n-1,j}$ and $S_{n-1,k}$ are the central and secondary subparts from layer $n-1$, while $\mu_{n,i}$ and $\sigma_{n,i}$ define a Gaussian limiting the difference between locations of subpart activations (see definition of activation below). For clarity, we shall omit subscripts in the following equations and use P, C, S, μ and σ to denote a part and its components.

A composition is *activated* (propagates output to higher layers) when all of its subparts are activated. This strict condition can be softened with hallucination, as explained in section 3.3. Part activation is composed of two values: activation location L_P, which represents the location (frequency) at which the part is activated, and activation magnitude A_P, which represents the strength of activation. The location of part's activation is defined simply as the location of activation of its central subpart:

$$L_P = L_C. \quad (2)$$

Thus, central parts of compositions on different layers propagate their locations upwards through the hierarchy. The magnitude of activation is defined as:

$$A_P = tanh[G(L_C - L_S, \mu, \sigma) \cdot (A_C + A_S)], \quad (3)$$

where *tanh* stands for the hyperbolic tangent function that limits the magnitude to [0,1) and *G* represent the Gaussian that limits the difference in locations of the central part and the subpart according to μ and σ. As an example, $P_{2,2}$ in Figure 1 is defined as

$$P_{2,2} = \{P_{1,1}, P_{1,3}, (1200, 25)\}, \quad (4)$$

where μ and σ are given in cents. Therefore, it will be activated whenever $P_{1,1}$ and $P_{1,3}$ will be activated at locations approximately one octave (1200 cents) apart. Two such activations are shown in the figure, one at 294 Hz and one at 440 Hz.

3.3 Inference

The model can be used as a feature extractor over any desired dataset. An audio signal, transformed into a time-frequency representation, serves as input for layer $\mathcal{L}_0$. Activations are then calculated layer-by-layer according to Equations 2 and 3. Additionally, two biologically-inspired mechanisms govern the inference process and increase robustness of the model: *hallucination* and *inhibition*.

Before we define both mechanisms, we need to introduce the concept of coverage. Coverage $c(P, L_P)$ of part P active at location L_P represents all signal information (frequency components) covered by the part and its subtree of parts. It is calculated top-down from an active part to $\mathcal{L}_0$ as:

$$c(P, L_P) = \bigcup \{c(C, L_P), c(S, L_P + \mu)\}. \quad (5)$$

For the $\mathcal{L}_0$ layer, coverage is defined as the set of parts with positive activations $A_P > 0$, thus representing the set of covered frequency components. An example from Figure 1: the coverage of $P_{2,2}$ active at 294 Hz is the set of frequencies: $\{294Hz, 588Hz, 880Hz\}$.

3.3.1 Hallucination

Hallucination deals with filling-in the missing or damaged information in the signal and is implemented by enabling part activation in presence of incomplete input. The missing information in the signal can be replaced with knowledge encoded in the model during learning by allowing activations of parts most fittingly covering the information present. This allows the model to produce hypotheses in situations with no straight result. Hallucination also boosts alternative explanations of input data, thus increasing its explanation power and robustness.

Hallucination is governed by parameter τ_1 which can be defined per layer and modified during the inference. It changes the conditions under which a part may be activated. The default condition, as explained in section 3.2, is that activation of a part is possible when all of its subparts are active. With hallucination, a part P may be activated at location L_P, when the number of frequency components it covers $|c(P, L_P)|$, divided by the maximal number of components it may cover is larger than τ_1. For example, a τ_1 of 0.75 means that $\frac{3}{4}$ of all possible frequency components must be covered by the part for it to be activated. A τ_1 of 1 represents the default behavior.

3.3.2 Inhibition

The second biologically-inspired mechanism provides a balancing factor by reducing redundant activations, similar to lateral inhibition performed by the human auditory system. Inhibition refines the set of parts that yield competing hypotheses of the same fragments of information in the input signal. Parts with greater activation magnitudes are retained and weaker activations inhibited. Inhibition also reduces activations that result from noise in the signal.

Activation of part P at L_P is inhibited, when another part Q with activation L_Q on the same layer (or a set of parts) covers the same fragments of information in the input signal, but with higher activation. The condition can be expressed as:

$$\exists Q : \frac{|c(P, L_P) \backslash c(Q, L_Q)|}{|c(P, L_P)|} < \tau_2 \wedge A_Q > A_P, \quad (6)$$

where τ_2 defines the amount of inhibition. For example, a value of 0.5 means that activation of P is inhibited if half of its coverage is already covered by another, stronger part.

To sum up: inference yields a set of activations on all model layers by calculating activations considering hallucination and inhibition over all layers in a bottom-up order and over all time-frames of the input signal. Resulting activations represent model features and can be directly interpreted or used as inputs for discriminative tasks.

3.4 Learning

The model is learned in an unsupervised manner on a set of input signals. It is constructed layer-by-layer, similar

to other deep architectures. The learning process relies on statistics of part activations, thus signal regularities are the driving force of the learning process.

When building layer $\mathcal{L}_n$, co-occurrences of activations of parts on $\mathcal{L}_{n-1}$ are observed. Compositions are formed from parts that frequently activate together at similar distances. All such parts are joined into compositions and added to the set of candidate compositions $\mathcal{P}$. When forming a composition of two frequently co-occurring parts, the part at the lower location represents the central part of the composition, while parameters μ and σ are estimated from all co-occurring activations of the two parts.

To reduce the number of compositions on each layer and keep only the most informative ones, the set of candidates $\mathcal{P}$ is refined. The goal of refinement is to reduce the number of compositions in the learned layer while maintaining sufficient coverage of information in the learning set.

Refinement is implemented with a greedy approach, where in each iteration, a part that contributes most to the coverage of information in the learning set, is selected and added to the layer. Refinement is concluded when one of the following two criteria are reached: a sufficient percentage of information in the learning set is covered (according to threshold τ_3), or no part remaining in the candidate set adds to the cumulative coverage of information. Algorithm 1 outlines the described approach.

Algorithm 1 Greedy approach for selection of compositions from the candidate set $\mathcal{P}$. Parts that add most to the coverage of information in the learning set are preferred. Function $perc$ calculates the percentage of information covered in the learning set by the given set of parts.

procedure REFINE($\mathcal{P}$)
  $prevCov \leftarrow 0$
  $coverages \leftarrow \emptyset$
  $\mathcal{L}_n \leftarrow \emptyset$
  repeat
    for $P \in \mathcal{P}$ **do**
      $coverages[P] \leftarrow perc(\mathcal{L}_n \cup P)$
    $Chosen \leftarrow \underset{P}{\operatorname{argmax}}(coverages)$
    $\mathcal{L}_n \leftarrow \mathcal{L}_n \cup Chosen$
    $\mathcal{P} \leftarrow \mathcal{P} \setminus Chosen$
    if $coverages[Chosen] = prevCov$ **then**
      break *//No added coverage - finish*
    $prevCov \leftarrow coverages[Chosen]$
  until $prevCov > \tau_3 \vee \mathcal{P} = \emptyset$

3.5 Time

The model presented so far is time-independent. It operates on a frame-by-frame basis, where each time frame in the time-frequency representation is treated independently from others. Music, however, evolves in time and models that operate on such bases often fail to reflect the evolution of sound properly.

The proposed model can be naturally extended to include the time dimension. Our first step towards extending the model for time-dependent processing was to implement a short-time automatic gain control mechanism, similar to the automatic gain control contrast mechanism in human and other animal perceptual systems. The mechanism integrates part activations at similar locations over time. When a new part activation appears and persists, its value is initially boosted to accentuate the onset and later suppressed towards a stable value.

The mechanism operates on all layers, and has a short-term effect on lower layers, and longer-term effect on higher layers due to the upward propagation of activations. Its end effect is that it stabilizes activations, reduces noise, produces smoother model output and boosts event onsets.

3.6 Relation to Deep Architectures

The compositional hierarchical model shares a great deal of similarities with other deep learning architectures. The structure of the model is similar in terms of learning a variety of signal abstractions on several layers of granularity. The model is learned in an unsupervised generative manner, thus, no annotated data is needed. The learning procedure is similar: the structure is built layer-by-layer. The proposed model can also be used for discriminative tasks by observing activations of parts on multiple layers.

We see the biggest advantage of the proposed compositional hierarchical model over other established deep architectures in its transparency. As parts are compositions of subparts, their activations are directly observable and interpretable. This opens the model up for a variety of interesting usages, as it not only produces features that can be used, but features that can be interpreted and explained. In addition, the inhibition and hallucination mechanisms make it possible to produce alternative explanations of the input by suppressing the winning explanation and search for alternatives. In comparison to DBNs, where the outputs of each layer can only be interpreted during the evaluation, the proposed model offers a deeper analysis of results by tracing the higher layer activations over all layers and investigating the impact of each subpart.

Another difference in comparison to DBNs is the shareability and *relativeness* of parts, which both lead to a small number of parts needed to represent complex signals. A part in the proposed model is defined by the relative distance between its subparts and can thus be activated on different locations along the frequency axis. Thus, the large amount of layer units that DBNs need to cover the entire spectrum is not necessary and is replaced by reusing the available parts. This relativeness is accompanied with the concept of part shareability: parts on a layer may be shared by many compositions on higher layers. For example, a chord is composed of at least three pitches which may be identical in their representation in our model.

We show the usefulness of the described model's features in the evaluation section, where the model is used as both feature extractor and a classifier. Other possible applications exploiting the the model's structure are presented in section 5.

4. EVALUATION OF THE MODEL

The presented model is applicable to different MIR tasks. To present the model's usefulness, we built a three-layer

model and evaluated it on two tasks: automated chord estimation and multiple fundamental frequency estimation.

The input layer was the same for both tasks. A constant-Q transform was used to transform music signals onto 345 frequency bins between 55 and 8000 Hz, with a step size of 50 ms and maximal window size of 100 ms. Two layers of compositions $\mathcal{L}_1$ and $\mathcal{L}_2$ were learnt as described previously. Due to the shareability of parts, the they contain only 23 and 12 parts respectively. The small number of parts in the model should mean that the model could be trained on a small learning set. We tested this hypothesis and trained the model on large and small datasets, and observed few differences. We were therefore able to build the model by using only a small set of 88 piano key samples as our learning set. We used the $\mathcal{L}_2$ layer for the task of multiple fundamental frequency estimation. For the task of automated chord estimation, we provided an additional $\mathcal{L}_3$ *octave-invariant* layer. The latter consists of 48 parts, where $\mathcal{L}_3$ activations correspond to octave-invariant activations of the $\mathcal{L}_2$.

4.1 Automated Chord Estimation

The time-independent model was tested for the task of automated chord estimation on the standard *Beatles* dataset, kindly provided by C. Harte. We used activations of the octave-invariant $\mathcal{L}_3$ layer as features and made the classification by using a hidden Markov model (HMM) with 24 states, each representing a chord, as described by [2]. We used cross-validation for evaluation; one album was used for HMM training and the rest of the dataset for estimation.

Our per-frame classification accuracy on the given dataset was 67.14 % with 0.1525 standard deviation. Compared to other per-frame approaches, we find our results slightly lower than for example [11], which also used per-frame technique for feature extraction. Nevertheless, we performed the evaluation as a proof of concept with time-independent feature extraction and no fine-tuning of the model, its learning, nor tuning of HMM parameters. We anticipate significant results increase by extending the model to time-dependent evaluation, using the whole hierarchy for classification and parameter tuning.

4.2 Multiple fundamental frequency estimation

The model was also tested for the task of multiple fundamental frequency estimation (MFEE) on the two subsets of MAPS (MIDI Aligned Piano Sounds) dataset, provided by [4]. Activations of layer $\mathcal{L}_2$ were directly used as fundamental frequency estimations with no further processing.

The following metrics were used for evaluation: per-frame precision and recall, precision and recall without penalising for octave errors, and pitch-class precision and recall. Results are shown in Table 1. Our results are significantly lower when compared to recent approaches, e.g. [14] which reported 77.1% classification accuracy on the subsets. However, the mentioned approach differs significantly from ours, as a severely larger dataset (approx. 4 times larger than the test sets) was used for training the support vector machine (SVM) classifier. In comparison, our model was trained only on a small set of piano key samples, so no parts of the MAPS dataset were used for training. It is also worth to mention that for this task, our model was used as a feature extractor and a classifier at the same time. We expect that accuracy would be improved if a classifier such as a SVM would be added on top of our model and would take features extracted on all layers for inputs. Our intention for this paper, however, is to present the general applicability of the model for multiple tasks and to avoid fine-tuning.

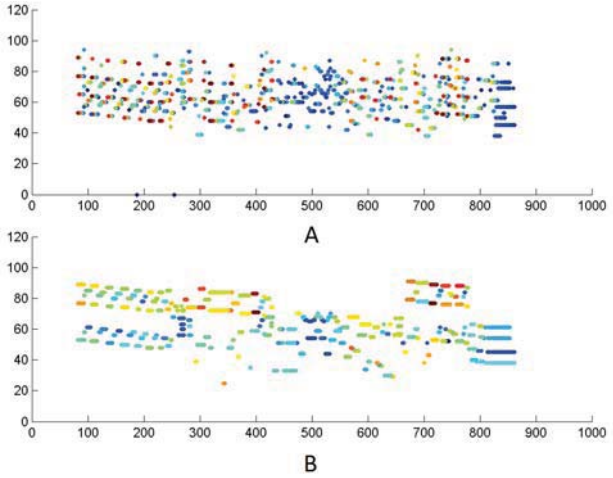


Figure 2. Hypotheses produced by our model for the task of multiple fundamental frequency estimation (A) and the ground truth (B). X axis represent time (in frames), and y midi pitches. Although the model produces many possible hypotheses per frame, only the ones with the highest magnitudes are used for comparison. Colors represent the magnitudes of activations in Fig. A or the MIDI velocity in Fig. B.

Table 1. Classification accuracy (CA) using all hypotheses provided by the model, precision (Pr) and recall (Re) values over a part of the MAPS dataset. Results without penalising octave errors and considering only pitch classes are marked with O and PC subscripts respectively.

Folder name	CA	Pr	Re
$AkPnBcht$	56.53 %	19.40 %	55.69 %
$AkPnBsdf$	66.17 %	22.05 %	61.27 %
$AkPnBcht_O$	67.08 %	35.37 %	64.55 %
$AkPnBsdf_O$	71.16 %	46.10 %	68.83 %
$AkPnBcht_{PC}$	86.20 %	51.83 %	86.59 %
$AkPnBsdf_{PC}$	88.23 %	58.68 %	70.99 %

5. OTHER APPLICATIONS OF THE MODEL

Our intention with developing the proposed model is to make an interpretable model that overcomes some of the limitations of DBNs and can be used for tackling various MIR tasks. Its transparency, however, also makes other

uses of the model possible.

The hierarchical approach presented in this paper fits well with the hierarchical structure of music in frequency as well as in time domains. Each part of the model represents an explainable entity (e.g. tone partial, pitch, chord). In contrast to the DBNs, each part of the model can be visualized. Visualization not only exposes the layered structure of the model, but also discloses information processed by the observed part and its influence on other parts and their activations. This insight into the music signal can be used in several scenarios — music visualization, music analysis and music synthesis.

We have developed a real-time visualization of the model, enabling deeper understanding of the processed information. When observing an inferred audio signal, the output of all layers of the model is presented by visualizing activations of parts. This insight enables detailed analysis of each event in the music signal and may bring additional event details to light. For example, a chord inversion can be observed by looking into the activated subtree of the chord from top layers to bottom-ones. Thus, visualization of our model offers an innovative user interface for music analysis.

The transparency of the model can also be exploited for music processing and synthesis. Parts across all layers form a variety of harmonic structures, and can be used for signal manipulation and synthesis. By activating a set of parts at different locations, a new spectral representation is produced. Although the interface may not provide a sufficient amount of features for a standalone music performance, it can be used as a sound generator in a combination with a music instrument, e.g. a MIDI keyboard. The interface thus serves as an advanced tool for spectral modification, while the instrument provides the interface for performance.

6. CONCLUSION AND FUTURE WORK

This paper presents a compositional hierarchical model as an alternative to deep learning architectures based on neural networks. The model shares a great deal of similarities with other deep architectures, including a multi-layer structure, unsupervised generative learning and suitability for discriminative tasks. Furthermore, the white-box structure of the model offers new utilizations of the model. We highlighted three possible applications: feature extraction for MIR tasks, music visualization and music analysis/synthesis.

The model's internals rely on findings in the fields of neurobiology and cognitive sciences. By implementing biologically-inspired mechanisms into the model, we made an attempt to build a model which partially resembles a subset of functions of the human auditory system. We intend to retain and further develop this aspect of the model with an intention to bring the computational modeling closer to human auditory perception.

The paper presents an initial development of our model. We plan to further extend it with the focus on temporal modeling. Parts can namely be extended into the time domain, thus bringing their activations closer to event-based modeling. We also plan to tackle temporal tasks, such as onset detection, as well as beat tracking and tempo estimation. The proposed model is also going to be evaluated for pattern analysis of symbolic data, including discovery of repeated themes, and symbolic melodic similarity.

7. REFERENCES

[1] Eric Battenberg and David Wessel. Analyzing Drum Patterns using Conditional Deep Belief Networks. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, pages 37–42, 2012.

[2] Juan P. Bello and Jeremy Pickens. A robust mid-level representation for harmonic content in music signals. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, pages 304–311, London, 2005.

[3] J. Stephen Downie, Andreas F. Ehmann, Mert Bay, and M. Cameron Jones. The Music Information Retrieval Evaluation eXchange: Some Observations and Insights. In Wieczorkowska A.A. and Ras Z.W., editors, *Advances in Music Information Retrieval*, pages 93–115. Springer-Verlag, Berlin, 2010.

[4] Valentin Emiya, Roland Badeau, and Bertrand David. Multipitch Estimation of Piano Sounds Using a New Probabilistic Spectral Smoothness Principle. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(6):1643–1654, August 2010.

[5] Philippe Hamel and Douglas Eck. Learning Features from Music Audio with Deep Belief Networks. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, pages 339–344, 2010.

[6] Eric J. Humphrey, Juan P. Bello, and Yann LeCun. Moving beyond feature design: deep architectures and automatic feature learning in music informatics. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, Porto, 2012.

[7] Honglak Lee, Peter Pham, Yan Largman, and Andrew Y. Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in Neural Information Processing Systems*, pages 1096–1104, 2009.

[8] Aleš Leonardis and Sanja Fidler. Towards scalable representations of object categories: Learning a hierarchy of parts. *Computer Vision and Pattern Recognition, IEEE*, pages 1–8, 2007.

[9] Abdel-rahman Mohamed, George E. Dahl, and Geoffrey Hinton. Acoustic Modeling using Deep Belief Networks. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):14–22, 2010.

[10] Nicola Orio. Music Retrieval: A Tutorial and Review. *Foundations and Trends® in Information Retrieval*, 1(1):1–90, 2006.

[11] Helene Papadopoulos and Geoffroy Peeters. Large-case Study of Chord Estimation Algorithms Based on Chroma Representation and HMM. *Content-Based Multimedia Indexing*, 53-60, 2007.

[12] Eric M. Schmidt and Youngmoo E. Kim. Learning Rhythm and Melody Features with Deep Belief Networks. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, pages 21–26, 2013.

[13] Erik M. Schmidt and Youngmoo E. Kim. Learning emotion-based acoustic features with deep belief networks. In *2011 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 65–68. IEEE, October 2011.

[14] Felix Weninger, Christian Kirst, Bjorn Schuller, and Hans-Joachim Bungartz. A discriminative approach to polyphonic piano note transcription using supervised non-negative matrix factorization. In *Proceedings of International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 6–10, Vancouver, 2013.

[15] Dong Yu and Li Deng. Deep Learning and Its Applications to Signal and Information Processing [Exploratory DSP. *IEEE Signal Processing Magazine*, 28(1):145–154, January 2011.

AN ANALYSIS AND EVALUATION OF AUDIO FEATURES FOR MULTITRACK MUSIC MIXTURES

Brecht De Man[1], Brett Leonard[2, 3], Richard King[2, 3], Joshua D. Reiss[1]
[1]Centre for Digital Music, Queen Mary University of London
[2]The Graduate Program in Sound Recording, Schulich School of Music, McGill University
[3]Centre for Interdisciplinary Research in Music Media and Technology
b.deman@qmul.ac.uk, brett.leonard@mail.mcgill.ca,
richard.king@mcgill.ca, joshua.reiss@qmul.ac.uk

ABSTRACT

Mixing multitrack music is an expert task where characteristics of the individual elements and their sum are manipulated in terms of balance, timbre and positioning, to resolve technical issues and to meet the creative vision of the artist or engineer. In this paper we conduct a mixing experiment where eight songs are each mixed by eight different engineers. We consider a range of features describing the dynamic, spatial and spectral characteristics of each track, and perform a multidimensional analysis of variance to assess whether the instrument, song and/or engineer is the determining factor that explains the resulting variance, trend, or consistency in mixing methodology. A number of assumed mixing rules from literature are discussed in the light of this data, and implications regarding the automation of various mixing processes are explored. Part of the data used in this work is published in a new online multitrack dataset through which public domain recordings, mixes, and mix settings (DAW projects) can be shared.

1. INTRODUCTION

The production of recorded music involves a range of expert signal processing techniques applied to recorded musical material. Each instrument or element thereof exists on a separate audio 'track', and this process of manipulating and combining these tracks is normally referred to as mixing. Strictly creative processes aside, each process can generally be classified as manipulating the dynamic (balance and dynamic range compression), spatial (stereo or surround panning and reverberation), and spectral (equalisation) features of the source material, or a combination thereof [1, 4, 8, 15].

Recent years have seen a steep increase in research on automatic mixing, where some of the tedious, routine tasks in audio production are automated to the benefit of the inexperienced amateur or the time constrained professional. Most research is concerned with the validation of a mixing rule based on knowledge derived from practical literature or expert interviews [2, 6, 7, 9], usually through an experiment where a method based on this assumption is compared to a set of alternative methods. Furthermore, some research has been done on machine learning systems for balancing and panning of tracks [13]. In spite of these efforts, the relation between the characteristics of the source material and the chosen processing parameters, as well as the importance of subjective input of the individual versus objective or generally accepted target features, is still poorly understood. Recurring challenges in this field include a lack of research data, such as high-quality mixes in a realistic but sufficiently controlled setting, and tackling the inherently high cross-adaptivity of the mixing problem, as the value of each processing parameter for any given track is usually dependent on features and chosen processing parameters associated with other tracks as well.

In this work, we conduct an experiment where a group of mixing engineers mix the same material in a realistic setting, with relatively few constraints, and analyse the manipulation of the signals and their features. We test the validity of the signal-dependent, instrument-independent model that is often used in automatic mixing research [6, 7], and try to identify which types of processing are largely dependent on instrument type, the song (or source material), or the individual mixing engineer. Consequently, we also identify which types of processing are not clearly defined as a function of these parameters, and thus warrant further research to understand their relation to low-level (readily extracted) features or high-level properties (instrument, genre, desired effect) of the source audio. We discuss the relevance of a number of audio features for the assessment of music production and the underlying processes as described above. This experiment also provides an opportunity to validate some of the most common assumptions in autonomous mixing research.

2. EXPERIMENT

The mixing engineers in this experiment were students of the MMus in Sound Recording at the Schulich School of Music at McGill University. They were divided in two groups of eight, where each group corresponds with a class

from a different year in the two-year programme, and each group was assigned a different set of four songs to mix. Each mixing engineer allocated up to 6 hours to each of their four mix assignments, and was allowed to use Avid's Pro Tools including built-in effects (with automation) and the Lexicon PCM Native Reverb Plug-In Bundle, a set of tools they were familiar with.

Four out of eight songs are available on a new multitrack testbed including raw tracks, the rendered mixes and the complete Pro Tools project files, allowing others to reproduce or extend the research. The testbed can be found on `c4dm.eecs.qmul.ac.uk/multitrack`. The authors welcome all appropriately licensed contributions consisting of shareable raw, multitrack audio, DAW project files, rendered mixes, or a subset thereof. Due to copyright restrictions, the other songs could not be shared.

We consider three types of instruments - drums, bass, and lead vocal - as they are featured in all test songs in this research, and as they are common elements in contemporary music in general. Furthermore, we split up the drums in the elements kick drum, snare drum, and 'rest'. Three out of eight songs had a male lead vocalist, and half of the songs featured a double bass (in one case part bowed) while the other half had a bass guitar for the bass part.

For the purpose of this investigation, we consider a fragment of the song only, consisting of the second verse and chorus, as all considered sources (drums, bass and lead vocal) are active here.

Whereas the audio was recorded and mixed at a sampling ratio of 96 kHz, we converted all audio to 44.1 kHz to reduce computational cost and to calculate spectral features based on the mostly audible region. The processed tracks are rendered from the digital audio workstation with all other tracks inactive, but with an unchanged signal path including send effects and bus processing [1].

3. FEATURES

The set of features we consider (Table 1) has been tailored to reflect properties relevant to the production of music in the dynamic, spatial and spectral domain. We consider the mean of the feature over all frames of a track fragment.

We use the perceptually informed measure of loudness relative to the loudness of the mix, as a simple RMS level can be strongly influenced by high energy at frequencies the human ear is not very sensitive to. To accurately measure loudness in the context of multitrack content, we use the highest performing modification in [12] (i.e. using a time constant of 280 ms and a pre filter gain of $+10$ dB) on the most recent ITU standard on measuring audio programme loudness [3].

[1] When disabling the other tracks, non-linear processes on groups of tracks (such as bus dynamic range compression) will result in a different effective effect on the rendered track since the processor may be triggered differently (such as a reduced trigger level). While for the purpose of this experiment, the difference in triggering of bus compression does not affect the considered features significantly, it should be noted that for rigorous extraction of processed tracks, in such a manner that when summed together they result in the final mix, the true, time-varying bus compression gain should be measured and applied on the single tracks.

Category	Feature	Reference
Dynamic	Loudness	[3, 12]
	Crest factor (100 ms and 1 s)	[17]
	Activity	[7]
Spatial	SPS	[16]
	$P_{[band]}$	[16]
	Side/mid ratio	
	Left/right imbalance	
Spectral	Centroid	[5]
	Brightness	
	Spread	
	Skewness	
	Kurtosis	
	Rolloff (.95 and .85)	
	Entropy	
	Flatness	
	Roughness	
	Irregularity	
	Zero-crossing rate	
	Low energy	[5]
	Octave band energies	

Table 1: List of extracted features

To reflect the properties of the signal related to dynamic range on the short term, we calculate the crest factor over a window of 100 ms and over a window of 1 s [17].

To quantify gating, muting, and other effects that make the track (in)audible during processing, we measure the percentage of time the track is active, with the activity state indicated by a Schmitt trigger with thresholds at -25 and -30 dB LUFS [7].

To analyse the spatial processing, we use the Stereo Panning Spectrum (SPS), which shows the spatial position of a certain frequency bin in function of time, and the Panning Root Mean Square ($P_{[band]}$), the RMS of the SPS over a number of frequency bins [16]. In this work, we use the absolute value of SPS, averaged over time, and the standard P_{total} (all bins), P_{low} (0-250 Hz), P_{mid} (250-2500 Hz) and P_{high} (2500-22050 Hz), also averaged over time. Furthermore, we propose a simple stereo width measure, the side/mid ratio, calculated as the power of side channel (sum of left and right channel) over the power of the mid channel (average of left channel and polarity-reversed right channel). We also define the left/right imbalance, as $(R - L)/(R + L)$ where L is the total/average power of the left channel, and R is the total/average power of the right channel. A centred track has low imbalance and low side/mid ratio, while a hard panned track has high imbalance and high side/mid ratio. Note that while these features are related, they do not mean the same thing. A source could have uncorrelated signals with the exact same energy in the left and right channel respectively, which would lead to a low left/right imbalance and a high side/mid ratio.

Finally, we use features included in the MIR Toolbox [5] (with the default 50 ms window length) as well as octave band energies to describe the spectral characteristics of the audio.

4. ANALYSIS AND DISCUSSION

4.1 Analysis of variance

Table 2 shows the mean values of the features, as well as the standard deviation between different mixing engineers and the standard deviation between different songs. Most considered features show greater variance for the same engineer across different songs, than for the same song over different engineers. Exceptions to this are the left/right imbalance and spectral roughness, which on average appear to be more dependent on the engineer than on the source content. The change of features (difference before and after processing, where applicable) varies more for different mixing engineers than for different songs, too, for all features. However, when considering the features instrument by instrument, the source material only rarely causes the means of the feature to differ significantly (the means are only significantly different through the effect of source material for the zero-crossing rate of the snare drum track, and for the spectral entropy of the vocal track). This suggests that engineers would disagree on processing values, whereas the source material has less effect.

For each feature, we perform an analysis of variance to investigate for which feature we can reject the hypothesis that the different 'treatments' (different source material, mixing engineer or instrument) result in the same feature value. For those features for which there is a significant effect ($p < 0.05$), we perform a multiple comparison of population means using the Bonferroni correction to establish what the mean values of the feature are as a function of the determining factor, and which instruments or songs have a significantly lower or higher mean than others. We discuss the outcome of these tests in the following paragraphs.

As some elements were not used by the mixing engineer, some missing values are dropped when calculating the statistics in the following sections.

4.2 Balance and dynamics processing

In general, the relative loudness of tracks, averaged over all instruments, is dependent on the song ($p < 5 \cdot 10^{-11}$). However, when looking at each instrument individually, the relative loudness of the bass guitar ($p < 0.01$), snare drum ($p < 0.05$) and other drum instruments ('rest', i.e. not snare or kick drum, $p < 5 \cdot 10^{-4}$) is dependent on mixing engineer.

In automatic mixing research, a popular assumption is that the loudness of the different tracks or sources should be equal [7]. A possible exception to this is the main element, usually the vocal, which can be set at a higher loudness [1]. From Figure 1, it is apparent that the vocal is significantly louder than the other elements considered here, whereas no significant difference of the mean relative loudness of the other elements can be shown. Furthermore, the relative loudness of the vocal shows a relative narrow range of values (-2.7 ± 1.6 LU), suggesting an agreement on a 'target loudness' of about -3 LU relative to the overall mix loudness.

It should be noted that due to crosstalk between the drum microphones, the effective loudness of the snare drum and kick drum will differ from the loudness measured from the snare drum and kick drum tracks. As a result, disagreement of the relative loudnesses of snare drum and other drum elements such as overhead and room microphones does not necessarily suggest a significantly different desired loudness of the snare drum, as the snare drum is present in both of these tracks. In this work, however, we are interested in the manipulations of the different tracks as they are available to the engineer.

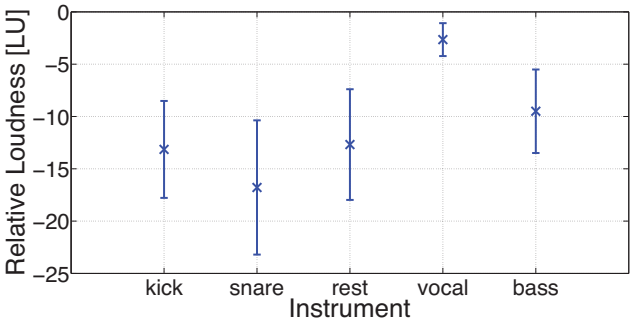


Figure 1: Average and standard deviation of loudness of sources relative to the total loudness of the mix, across songs and mixing engineers.

The crest factor is affected by both the instrument ($p < 5 \cdot 10^{-3}$) and song ($p < 10^{-20}$), and every instrument individually shows significantly different crest factor values for different engineers ($p < 5 \cdot 10^{-3}$). One exception to the latter is the kick drum for a crest factor window size of 1 s, where the hypothesis was not disproved for one group of engineers.

All instruments show an increase in crest factor compared to the raw values (ratio significantly greater than one). This means that the short-term dynamic range is effectively expanded, which can be an effect of dynamic range compression as transients are left unattenuated due to the response time of the compressor, while the rest of the signal is reduced in level.

The percentage of the time the track was active did not meaningfully change under the influence of different source material, individual mixing engineers or instruments. A drop in activity in some instances is due to gating of kick drum, but this is the decision of certain mixing engineers for certain songs, and no consistent trend.

4.3 Stereo panning

Both the average left/right imbalance and average side/mid ratio were significantly higher for the non-pop/rock songs ($p < 10^{-6}$).

The Panning Root Mean Square values $P_{[band]}$ all show a larger value for the total mix and for the 'rest' group. The difference is significant except for the lowest band, where only the bass is significantly more central than the total mix. This can be explained by noting that most of the low frequency sources are panned centre (see further).

In literature on automatic mixing and mixing engineering textbooks, it is stated that low-frequency sources as well as lead vocals and snare drums should be panned central [1, 2, 4, 6, 8–10, 14]. To quantify the spatialisation for different frequencies, we display the panning as a function

Feature	Kick drum	Snare drum	Rest drums	Bass	Lead vocal	Average	Mix
Loudness [LU]	-13.15 $\pm^{4.05}_{3.89}$	-16.78 $\pm^{6.17}_{4.57}$	-12.68 $\pm^{5.46}_{2.80}$	-2.65 $\pm^{1.52}_{1.31}$	-9.50 $\pm^{3.51}_{2.86}$	-10.95 $\pm^{4.14}_{3.09}$	N/A
Crest (100 ms)	3.599 $\pm^{0.603}_{0.330}$	4.968 $\pm^{0.998}_{0.469}$	4.510 $\pm^{1.065}_{0.354}$	2.565 $\pm^{0.443}_{0.166}$	3.315 $\pm^{0.403}_{0.208}$	3.791 $\pm^{0.634}_{0.274}$	3.332 $\pm^{0.294}_{0.116}$
Crest (1 s)	9.824 $\pm^{3.074}_{1.911}$	16.724 $\pm^{6.458}_{3.135}$	12.472 $\pm^{4.710}_{1.823}$	4.339 $\pm^{1.098}_{0.449}$	5.283 $\pm^{1.102}_{0.514}$	9.728 $\pm^{2.907}_{1.398}$	5.315 $\pm^{0.997}_{0.554}$
Activity	0.676 $\pm^{0.250}_{0.122}$	0.861 $\pm^{0.161}_{0.078}$	0.909 $\pm^{0.115}_{0.029}$	0.958 $\pm^{0.076}_{0.009}$	0.844 $\pm^{0.089}_{0.044}$	0.850 $\pm^{0.117}_{0.048}$	0.995 $\pm^{0.009}_{0.004}$
L/R imbalance	**0.075** $\pm^{\mathbf{0.094}}_{\mathbf{0.137}}$	**0.144** $\pm^{\mathbf{0.153}}_{\mathbf{0.227}}$	0.361 $\pm^{0.303}_{0.213}$	**0.107** $\pm^{\mathbf{0.135}}_{\mathbf{0.176}}$	**0.045** $\pm^{\mathbf{0.072}}_{\mathbf{0.085}}$	**0.146** $\pm^{\mathbf{0.139}}_{\mathbf{0.152}}$	0.088 $\pm^{0.075}_{0.074}$
Side/mid ratio	**0.036** $\pm^{\mathbf{0.055}}_{\mathbf{0.076}}$	**0.036** $\pm^{\mathbf{0.040}}_{\mathbf{0.043}}$	0.242 $\pm^{0.183}_{0.154}$	**0.009** $\pm^{\mathbf{0.013}}_{\mathbf{0.015}}$	**0.022** $\pm^{\mathbf{0.018}}_{\mathbf{0.022}}$	0.069 $\pm^{0.060}_{0.059}$	0.101 $\pm^{0.049}_{0.046}$
P_{total}	0.104 $\pm^{0.102}_{0.090}$	0.108 $\pm^{0.082}_{0.059}$	0.307 $\pm^{0.028}_{0.027}$	0.075 $\pm^{0.093}_{0.083}$	**0.134** $\pm^{\mathbf{0.022}}_{\mathbf{0.027}}$	0.145 $\pm^{0.060}_{0.052}$	0.234 $\pm^{0.030}_{0.027}$
P_{low}	**0.066** $\pm^{\mathbf{0.078}}_{\mathbf{0.087}}$	0.122 $\pm^{0.102}_{0.073}$	0.243 $\pm^{0.045}_{0.041}$	0.040 $\pm^{0.063}_{0.059}$	**0.147** $\pm^{\mathbf{0.034}}_{\mathbf{0.042}}$	0.123 $\pm^{0.061}_{0.056}$	0.188 $\pm^{0.042}_{0.034}$
P_{mid}	**0.066** $\pm^{\mathbf{0.074}}_{\mathbf{0.076}}$	0.114 $\pm^{0.090}_{0.064}$	0.290 $\pm^{0.023}_{0.023}$	0.052 $\pm^{0.082}_{0.067}$	**0.177** $\pm^{\mathbf{0.027}}_{\mathbf{0.035}}$	0.140 $\pm^{0.054}_{0.048}$	0.248 $\pm^{0.027}_{0.023}$
P_{high}	0.106 $\pm^{0.104}_{0.091}$	0.105 $\pm^{0.081}_{0.058}$	0.309 $\pm^{0.029}_{0.028}$	0.076 $\pm^{0.094}_{0.085}$	**0.124** $\pm^{\mathbf{0.022}}_{\mathbf{0.028}}$	0.144 $\pm^{0.061}_{0.053}$	0.231 $\pm^{0.033}_{0.029}$
Centroid [Hz]	2253.8 $\pm^{1065.6}_{729.8}$	4395.3 $\pm^{1448.6}_{554.2}$	4130.8 $\pm^{1228.1}_{483.2}$	1046.5 $\pm^{520.1}_{232.4}$	2920.2 $\pm^{452.1}_{264.7}$	2949.3 $\pm^{872.1}_{418.6}$	2478.8 $\pm^{517.9}_{247.1}$
Brightness	0.306 $\pm^{0.105}_{0.103}$	0.598 $\pm^{0.156}_{0.069}$	0.557 $\pm^{0.115}_{0.058}$	0.135 $\pm^{0.082}_{0.031}$	0.455 $\pm^{0.071}_{0.040}$	0.410 $\pm^{0.100}_{0.056}$	0.362 $\pm^{0.070}_{0.034}$
Spread	3250.1 $\pm^{783.2}_{447.5}$	4363.6 $\pm^{701.9}_{335.9}$	4422.1 $\pm^{734.6}_{292.3}$	2426.6 $\pm^{559.2}_{320.4}$	3369.9 $\pm^{324.6}_{191.3}$	3566.5 $\pm^{587.5}_{298.0}$	3453.2 $\pm^{421.7}_{200.6}$
Skewness	3.649 $\pm^{1.068}_{0.886}$	1.492 $\pm^{0.663}_{0.301}$	1.665 $\pm^{0.682}_{0.246}$	6.234 $\pm^{1.885}_{0.630}$	2.470 $\pm^{0.573}_{0.243}$	3.102 $\pm^{0.912}_{0.427}$	2.779 $\pm^{0.600}_{0.257}$
Kurtosis	23.847 $\pm^{11.997}_{9.164}$	5.965 $\pm^{2.905}_{1.474}$	7.053 $\pm^{3.449}_{1.263}$	58.870 $\pm^{31.874}_{11.107}$	11.579 $\pm^{4.267}_{1.784}$	21.463 $\pm^{9.834}_{4.477}$	13.646 $\pm^{4.511}_{2.073}$
Rolloff .95 [Hz]	8880.1 $\pm^{3679.2}_{2151.2}$	13450.9 $\pm^{3100.6}_{1582.2}$	13373.4 $\pm^{2594.1}_{1007.4}$	4389.4 $\pm^{2714.7}_{1244.5}$	9879.0 $\pm^{1335.7}_{725.3}$	9994.5 $\pm^{2498.0}_{1240.8}$	9679.0 $\pm^{1563.8}_{734.3}$
Rolloff .85 [Hz]	4513.7 $\pm^{2736.6}_{1788.8}$	8984.3 $\pm^{3139.7}_{1348.5}$	8755.3 $\pm^{2742.5}_{975.6}$	1625.5 $\pm^{1205.0}_{594.3}$	5595.8 $\pm^{1121.4}_{609.7}$	5894.9 $\pm^{2047.2}_{986.1}$	5026.2 $\pm^{1337.8}_{599.8}$
Entropy	0.655 $\pm^{0.104}_{0.090}$	0.840 $\pm^{0.084}_{0.057}$	0.832 $\pm^{0.051}_{0.025}$	0.552 $\pm^{0.073}_{0.026}$	0.735 $\pm^{0.043}_{0.016}$	0.723 $\pm^{0.066}_{0.038}$	0.744 $\pm^{0.043}_{0.015}$
Flatness	0.148 $\pm^{0.072}_{0.051}$	0.350 $\pm^{0.142}_{0.056}$	0.337 $\pm^{0.118}_{0.045}$	0.073 $\pm^{0.035}_{0.020}$	0.167 $\pm^{0.030}_{0.018}$	0.215 $\pm^{0.074}_{0.035}$	0.174 $\pm^{0.046}_{0.020}$
Roughness	**84.72** $\pm^{\mathbf{84.85}}_{\mathbf{98.32}}$	**36.30** $\pm^{\mathbf{41.16}}_{\mathbf{43.32}}$	67.57 $\pm^{71.76}_{46.28}$	**236.04** $\pm^{\mathbf{160.38}}_{\mathbf{176.05}}$	**247.00** $\pm^{\mathbf{216.15}}_{\mathbf{247.36}}$	**134.33** $\pm^{\mathbf{319.30}}_{\mathbf{338.44}}$	**1843.31** $\pm^{\mathbf{1341.50}}_{\mathbf{1419.35}}$
Irregularity	0.158 $\pm^{0.098}_{0.063}$	0.235 $\pm^{0.151}_{0.079}$	0.297 $\pm^{0.135}_{0.069}$	0.502 $\pm^{0.176}_{0.065}$	0.540 $\pm^{0.165}_{0.094}$	0.346 $\pm^{0.136}_{0.075}$	0.705 $\pm^{0.090}_{0.078}$
Zero-crossing	584.7 $\pm^{509.5}_{409.4}$	2222.0 $\pm^{1183.3}_{604.7}$	1988.9 $\pm^{944.1}_{466.1}$	246.6 $\pm^{217.8}_{89.6}$	1177.5 $\pm^{233.7}_{143.6}$	1243.9 $\pm^{554.3}_{305.4}$	905.2 $\pm^{237.4}_{118.8}$
Low energy	0.752 $\pm^{0.113}_{0.081}$	0.723 $\pm^{0.084}_{0.055}$	0.682 $\pm^{0.047}_{0.034}$	0.507 $\pm^{0.096}_{0.033}$	0.544 $\pm^{0.065}_{0.048}$	0.641 $\pm^{0.073}_{0.048}$	**0.541** $\pm^{\mathbf{0.035}}_{\mathbf{0.038}}$

Table 2: Average values of features per instrument, including average over instrument and value of total mix, with standard deviation between different songs by the same mixing engineer (top), and between different mixes of the same song (bottom). Values for which the variation across different mixes for the same song is greater than the variation across different songs for the same engineer are displayed in bold.

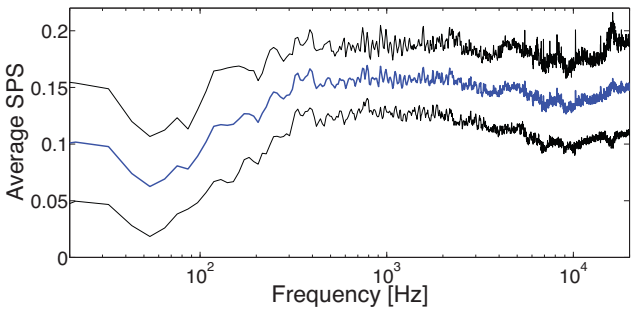


Figure 2: Mean Stereo Panning Spectrum (with standard deviation) over all mixes and songs

of frequency in Figure 2, using the average Stereo Panning Spectrum over all mixes and songs. From this figure a clear increase in SPS with increasing frequency is apparent between 50 Hz and 400 Hz. However, this trend does not extend to the very low frequencies (20-50 Hz) or higher frequencies (>400 Hz).

4.4 Equalisation

To assess the spectral processing of sources, mostly equalisation in this context, we consider both the absolute values of the spectral features (showing the desired features of the processed audio) as well as the change in features (showing common manipulations of the tracks). When only taking the manipulations into account, and not the features of the source audio, similar to blindly applying a software equaliser's presets, the results would be less translatable to situations where the source material's spectral characteristics differs from that featured in this work [2]. However, considering the change in features could reveal common practices that are less dependent on the features of the source material. Therefore, we investigate both.

The spectral centroid of the whole mix varies strongly depending on the mixing engineer ($p < 5 \cdot 10^{-6}$). The centroid of the snare drum track is consistently increased through processing, due to a reduction of the low energy content as well as spill of instruments like kick drum (see further regarding the reduction of low energy) and/or the emphasis of a frequency range above the original centroid.

The brightness of each track except bass guitar and kick drum (the sources with the highest amount of low energy) is increased.

For a large set of spectral features (spectral centroid, brightness, skewness, roll-off, flatness, zero-crossing, and roughness), the engineers disagree on the preferred value for all instruments except kick drum. In other words, the values describing the spectrum of a kick drum across engineers are overlapping, implying a consistent spectral target (a certain range of appropriate values). For other features

(spread, kurtosis and irregularity) the value corresponding with the kick drum track is also significantly different across engineers. The roughness shows no significantly different means for any instrument except the 'rest' bus.

The low energy of each track is reduced for each instrument, with significantly more reduction for snare drum than for kick drum and bass guitar. Its absolute value for bass and vocal is significantly different across engineers, whereas there is a general overlap for all other instruments including the mix. As the variation in the resulting value of low energy is higher than the variation for the unprocessed versions, no target value is apparent for any instrument, nor for the total mix.

Analysis of the octave band energies reveals definite trends across songs and mixing engineers, for a certain instrument as well as the mix. The standard deviation does not consistently decrease or increase over the octave bands for any instrument when compared to the raw audio. The suggested 'mix target spectrum' is in agreement with [11], which derived a 'target spectrum' based on average spectra of number one hits from various genres and over several decades. Figure 4 shows the measured average mix spectrum against the octave band values of the average spectrum of a number one hit after 2000 from that work, which lies within a standard deviation from our result with the exception of the highest band. The average relative change in energies is not significantly different from zero (no bands are consistently boosted or cut for certain instruments), but taking each song individually in consideration, a strong agreement of reasonably drastic boosts or cuts is shown for some songs. This confirms that the equalisation is highly dependent on the source material, and engineers largely agree on the necessary treatment for source tracks showing spectral anomalies.

5. CONCLUSION

We conducted a controlled experiment where eight multitrack recordings mixed by eight mixing engineers were analysed in terms of dynamic, spatial and spectral processing of common key elements.

We measured a greater variance of features across songs than across engineers, for each considered instrument and for the total mix, whereas the mean values corresponding to the different engineers were more often statistically different from each other.

The relative loudness of the lead vocal track was found to be significantly louder than all other tracks, with an average value of -3 LU relative to the total mix loudness.

The amount of panning as a function of frequency was investigated, and found to be increasing with frequency up to about 400 Hz, above which it stays more or less constant.

We measured a consistent decrease of low frequency energy and an increase of crest factor for all instruments, and an increase of the spectral centroid of the snare drum track. Spectral analysis has shown a definite target spectrum that agrees with the average spectrum of recent commercial recordings.

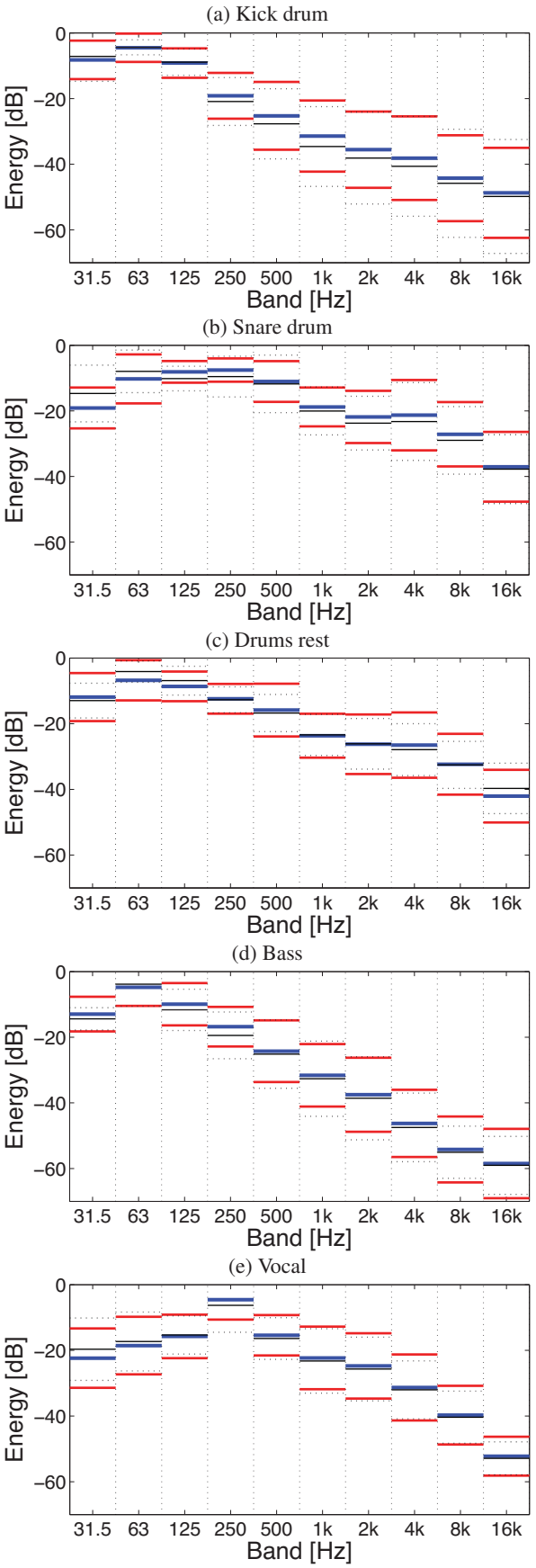


Figure 3: Average octave band energies (blue) with standard deviation (red) for different instruments after processing, compared to the raw signal (black).

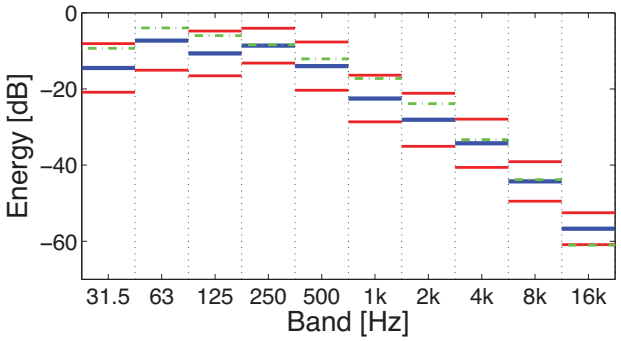


Figure 4: Average octave band energies for total mix, compared to 'After 2000' curve from [11] (green dashed line)

6. FUTURE WORK

Future work will be concerned with perceptual evaluation of mixes and its relation to features, using both qualitative ('which sonic descriptors correspond with which features?') and quantitative analysis ('which manipulation of audio is preferred?').

Further research is needed to establish the desired loudness of sources, as opposed to loudness of tracks, and its variance throughout songs, genres, and mixing engineers.

An extrapolation of the analysis described in this paper to other instruments is needed to validate the generality of the conclusions regarding the processing of drums, bass and lead vocal at the mixing stage, and to further explore laws underpinning the processing of different instruments.

Based on the findings of this work, which showed trends and variances of different relevant features, we can inform knowledge engineered or machine learning based systems that automate certain mixing tasks (balancing, panning, equalising and compression).

This work was based on a still relatively limited set of mixes, for which the engineers came from the same institution. Through initiatives such as the public multitrack testbed presented in this paper, it will be possible to analyse larger corpora of mixes, where more parameters can be investigated with more significance.

7. ACKNOWLEDGEMENTS

The authors would like to thank George Fazekas for assistance with launching the multitrack testbed.

This study was funded in part by the Engineering and Physical Sciences Research Council (EPSRC) Semantic Media grant (EP/J010375/1).

8. REFERENCES

[1] Alex Case. *Mix Smart: Professional Techniques for the Home Studio*. Focal Press. Taylor & Francis, 2011.

[2] Brecht De Man and Joshua D. Reiss. A knowledge-engineered autonomous mixing system. In *135th Convention of the Audio Engineering Society*, 2013.

[3] ITU. Recommendation ITU-R BS.1770-3 Algorithms to measure audio programme loudness and true-peak audio level. Technical report, Radiocommunication Sector of the International Telecommunication Union, 2012.

[4] Roey Izhaki. *Mixing audio: concepts, practices and tools*. Focal Press, 2008.

[5] Olivier Lartillot and Petri Toiviainen. MIR in Matlab (II): A toolbox for musical feature extraction from audio. In *Proceedings of the 8th International Society for Music Information Retrieval Conference*, 2007.

[6] Stuart Mansbridge, Saoirse Finn, and Joshua D. Reiss. An autonomous system for multi-track stereo pan positioning. In *133rd Convention of the Audio Engineering Society*, 2012.

[7] Stuart Mansbridge, Saoirse Finn, and Joshua D. Reiss. Implementation and evaluation of autonomous multi-track fader control. In *132nd Convention of the Audio Engineering Society*, 2012.

[8] Bobby Owsinski. *The Mixing Engineer's Handbook*. Course Technology, 2nd edition, 2006.

[9] Enrique Perez-Gonzalez and Joshua D. Reiss. Automatic mixing: Live downmixing stereo panner. In *10th International Conference on Digital Audio Effects (DAFx-10)*, 2007.

[10] Pedro Pestana. *Automatic mixing systems using adaptive digital audio effects*. PhD thesis, Catholic University of Portugal, 2013.

[11] Pedro Duarte Pestana, Zheng Ma, Joshua D. Reiss, Alvaro Barbosa, and Dawn A. A. Black. Spectral characteristics of popular commercial recordings 1950-2010. In *135th Convention of the Audio Engineering Society*, 2013.

[12] Pedro Duarte Pestana, Joshua D. Reiss, and Alvaro Barbosa. Loudness measurement of multitrack audio content using modifications of ITU-R BS.1770. In *Audio Engineering Society Convention 134*, 2013.

[13] Jeff Scott and Youngmoo E. Kim. Analysis of acoustic features for automated multi-track mixing. In *Proceedings of the 12th International Society for Music Information Retrieval Conference*, 2011.

[14] Jeff Scott and Youngmoo E. Kim. Instrument identification informed multi-track mixing. In *Proceedings of the 14th International Society for Music Information Retrieval Conference*, 2013.

[15] M. Senior. *Mixing Secrets*. Taylor & Francis, 2012.

[16] George Tzanetakis, Randy Jones, and Kirk McNally. Stereo panning features for classifying recording production style. In *Proceedings of the 8th International Society for Music Information Retrieval Conference*, 2007.

[17] Earl Vickers. The loudness war: Background, speculation, and recommendations. *129th Convention of the Audio Engineering Society*, 2010.

DETECTING DROPS IN ELECTRONIC DANCE MUSIC: CONTENT BASED APPROACHES TO A SOCIALLY SIGNIFICANT MUSIC EVENT

Karthik Yadati, Martha Larson, Cynthia C. S. Liem, Alan Hanjalic
Delft University of Technology
{n.k.yadati,m.a.larson,c.c.s.liem,a.hanjalic}@tudelft.nl

ABSTRACT

Electronic dance music (EDM) is a popular genre of music. In this paper, we propose a method to automatically detect the characteristic event in an EDM recording that is referred to as a *drop*. Its importance is reflected in the number of users who leave comments in the general neighborhood of drop events in music on online audio distribution platforms like SoundCloud. The variability that characterizes realizations of drop events in EDM makes automatic drop detection challenging. We propose a two-stage approach to drop detection that first models the sound characteristics during drop events and then incorporates temporal structure by zeroing in on a watershed moment. We also explore the possibility of using the drop-related social comments on the SoundCloud platform as weak reference labels to improve drop detection. The method is evaluated using data from SoundCloud. Performance is measured as the overlap between tolerance windows centered around the hypothesized and the actual drop. Initial experimental results are promising, revealing the potential of the proposed method for combining content analysis and social activity to detect events in music recordings.

1. INTRODUCTION

Electronic dance music (EDM) is a popular genre of dance music which, as the name suggests, is created using electronic equipment and played in dance environments. Outside of clubs and dance festivals, EDM artists and listeners actively share music on online social platforms. Central to the enjoyment of EDM is a phenomenon referred to as "The Drop". Within the EDM community, a *drop* is described as a moment of emotional release, where people start to dance "like crazy" [12]. There is no precise recipe for creating a drop when composing EDM. Rather, a drop occurs after a *build*, a building up of tension, and is followed by the re-introduction of the full bassline [1]. A given EDM track may contain one or more drop moments. The designation "The Drop" is generally reserved for the overall phenomenon rather than specific drop events.

In this paper we address the challenge of automatically detecting a drop in a given EDM track. The social significance of the drop in the EDM context can be inferred, for instance, from the websites that compile a playlist of the best drops [1]. It is also evident from vivid social activity around drop events on online audio distribution platforms such as SoundCloud [2]. We also mention here a documentary, scheduled to be released in 2014, tracking the evolution of EDM as a cultural phenomenon, and titled *The Drop* [3]. Ultimately, the drop detection approach proposed in this paper could serve both EDM artists and listeners. For example, it would enable artists to compare drop creation techniques, and would also support listeners to better locate their favorite drop moments.

The challenge of drop detection arises from the high variability in different EDM tracks, which differ in their musical content and temporal development. Our drop detection approach uses audio content analysis and machine learning techniques to capture this variability. As an additional source of reference labels for classifier training, we explore the utility of drop-related social data in the form of *timed comments*, comments associated with specific time codes. We draw our data from SoundCloud, a music distribution platform that supports timed comments and is representative of online social sharing of EDM. The paper makes three contributions:

- We propose a two-step content-based drop detection approach.
- We verify the ability of the approach to detect *drops* in EDM tracks.
- We demonstrate utility of the social features (timed comments on SoundCloud) to reduce the amount of hand-labeled data needed to train our classifier.

The remainder of this paper is organized as follows. Section 2 discusses related work, and is followed by the presentation and evaluation of our method in sections 3 and 4. Section 5 provides a summary and an outlook towards future work.

[1] http://www.beatport.com/charts/top-10-edm-drops-feb1/252641
[2] http://soundcloud.com
[3] http://www.imdb.com/title/tt2301898/

2. RELATED WORK

Although Electronic Dance Music is a popular music genre attracting large audiences, it has received little attention in the music information retrieval research community. Research on EDM is limited to a small number of contributions. Here, we mention the most notable. Hockman et al. [5] propose a genre-specific beat tracking system that is designed to analyze music from the following EDM subgenres: Hardcore, Jungle, Drum and Bass. Kell et al. in [6] also apply audio content analysis to EDM in order to investigate track ordering and selection, which is usually carried out by human experts, i.e., Disc Jockeys (DJ). The work report findings on which content features influence the process of ordering and selection. A musicological perspective is offered by Collins in [3], who applies audio content analysis and machine learning techniques to empirically study the creative influence of earlier musical genres on the later ones using a date annotated database of EDM tracks, with specific focus on the sub-genres Detroit techno and Chicago house. Our work strives to redress the balance and give more attention to EDM. It draws attention to SoundCloud as an important source of music data and associated social annotations, and also to "The Drop", a music event of key significance for the audience of EDM.

The rise of social media has also seen the rise in availability of user-contributed metadata (e.g., comments and tags). Social tags have recently grown in importance in music information retrieval research. In [11], they were used to predict perceived or induced emotional responses to music. This work reports findings on the correlation between the emotion tags associated with songs on Last.fm—"happy", "sad", "angry" and "relax"—and the user emotion ratings for perceived and induced emotions. Social data is generally noisy, since generating precise labels is not users' primary motivation for tagging or commenting. However, this data can still prove useful as weak reference labels, reducing the burden of producing ground-truth labels for a large set of music tracks, which is an expensive and time consuming task. Social tags available on Last.fm have been used to automatically generating tags for songs [4]. An interesting direction of research is described in [13], where the authors use content-based analysis of the song to improve the tags provided by users. Existing work makes use of social tags that users assign to a song as a whole. In contrast, our work makes use of *timed comments* that users contribute associated with specific time points during a song.

Obtaining time-code level ground-truth labels for a large set of music tracks is an expensive and time consuming task. One way to obtain reference labels is to use crowdsourcing, where users are explicitly offered a task (e.g., label the type of emotion [9]). Our approach of using timed comments spares the expense of crowdsourcing. It has the additional advantage that users have contributed the comments spontaneously, i.e., they have not been asked to explicitly assign them, making them a more natural expression of user reactions during their listening experience.

3. PROPOSED APPROACH

Our proposed two-step approach is based on general properties of the "The Drop". As previously mentioned drops are characterized by a build up towards a climax followed by reintroduction of the bassline. We hypothesize that the switch will coincide with a structural segment that ends at a drop moment. For this reason, the first step in our approach is segmentation. However, not all segment boundaries are drops. For this reason, the second step in our approach is a content-based classification of segments that eliminates segments whose boundaries are not drop points. Figure 1 illustrates the two-stage approach, where we first segment to identify drop candidates and then classify in order to isolate candidates that are actually drop moments.

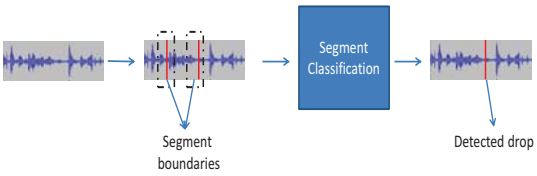


Figure 1. Two-stage approach to drop detection

The classification framework we propose to find drop events is illustrated in Figure 2. At the heart of the framework are the following modules: Segmentation, feature extraction, classification and evaluation.

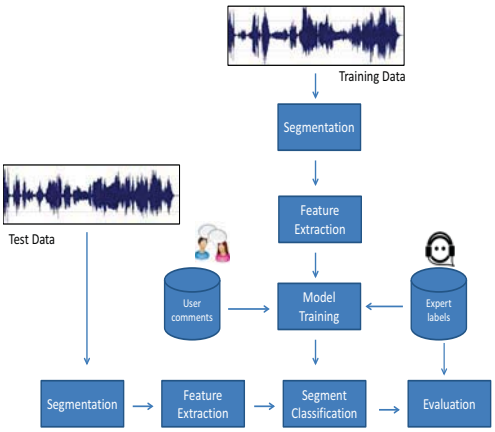


Figure 2. The proposed classification framework

3.1 Segmentation

The segmentation step carries out unsupervised segment boundary detection. Exploratory experiments revealed that the segmentation method proposed in [10] gives a good first approximation of the drops in an EDM track, and we have adopted it for our experiments. The method uses the chroma features computed from the audio track to identify the segment boundaries. We use the same parameters as used in [10]: 12 pitch classes, a window length of 209 ms, and a hop size of 139 ms. We carried out an intermediate evaluation to establish the quality of the drop candidates

generated by the segmentation step alone. The average distance between the actual drop (ground-truth) and a segment boundary generated by our segmentation method is 2.5 seconds, and less than 8% of the drops are missed in our training set (described in Section 4.1).

3.2 Feature Extraction for Classification

An overview of the feature extraction process is illustrated in Figure 3.

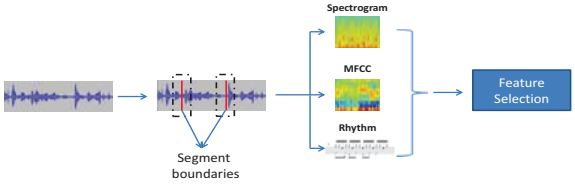


Figure 3. Feature extraction procedure

After segmentation, we extract content-based features from a fixed length window around the segment boundary. We use the following features: Spectrogram, MFCC and features related to rhythm. We adopt Mel-Frequency Cepstral Coefficients (MFCC) and features computed from the spectrogram because of their effectiveness. A unique feature of a drop is that it is preceded by a buildup or *build*. Figure 4 indicates that this buildup can be clearly observed in the spectrogram of an audio segment containing a drop. This provides additional motivation to use features computed from the spectrogram in our approach. We use the statistics computed from the spectrogram in our method (mean and standard deviations of the frequencies). For MFCC and spectrogram calculation, we use a window size of 50 msec with a 50% overlap with the subsequent windows. We use 13 coefficients for the MFCC. Due to a

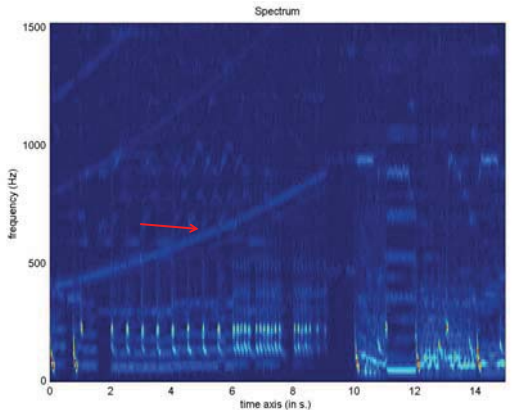


Figure 4. Spectrogram of an audio segment indicating a build (red arrow) towards a drop at 10 seconds.

switch of rhythm at the drop moment, features related to rhythm are another important source of information. We use the rhythm related features: rhythm patterns, rhythm histogram, temporal rhythm histogram [8]. We concatenate the rhythm features, MFCC and statistics computed from the spectrogram into a single feature vector. Feature selection, following the approach of [2], is performed on the training data in order to reduce the dimensionality of the feature vector and also to ensure that we use the most informative features in the classification step.

3.3 Training and Classification

To train the classifier, we assign drop (1) vs. non-drop (0) labels to time-points in the track using two sources of information: high fidelity ground-truth (manual labels provided by an expert) and user comments (weak reference labels).

Prior to training the model, we map the ground-truth labels to the nearest segment boundaries. We note that the segmentation step reduces the search space for the drop, as we no longer search for it in the entire track, but focus on features around the segment boundaries. We use a binary SVM classifier with a linear kernel as our training algorithm.

3.4 Evaluation

Our method predicts time points in a track at which the drop occurs. We consider each detected drop to be a distinct drop. The fact that the drop can only be hypothesized at a segment boundary keeps detections from occurring close together, given that the average length of segments generated by our segmentation algorithm is 16.5 seconds.

In order to report the performance in terms of accuracy and precision, we utilize the F1-score. Although the drop is annotated as a point in the track, it is characterized by the music around the point. This aspect of the drop motivates our choice of using a tolerance window of varying temporal resolutions around the hypothesized drop and use temporal overlap to compute the F1-score. We follow these steps to compute the F1-score:

1. Place a tolerance window of size t seconds centered around the hypothesized (from our approach) and the reference drop (ground-truth).
2. Compute the number of true positives (tp), false positives (fp) and false negatives (fn) as illustrated in Figure 5 (the unit of measurement being seconds). Note that the numbers computed here are related to the number of seconds of overlap between the windows placed over the actual drop and the predicted drop. These are computed for every detected drop in the track.
3. Compute the F1-score using the following equation: $F1 = \frac{2tp}{2tp+fn+fp}$.
4. Repeat the above steps for different sizes of t. We use windows sizes of t = *15 sec, 13 sec, 11 sec, 9 sec, 7 sec, 5 sec, 3 sec* to compute the F1 score.
5. If there is more than one drop in the track, repeat all the above steps and compute an average F1-score for each size of the window t.

4. EXPERIMENTS

We have proposed a classification framework for detecting drops in an EDM track. We use MIRToolbox [7] to extract features related to spectrogram and MFCC, while we

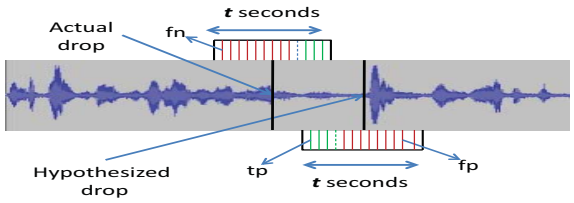


Figure 5. Illustration to compute true positive (tp), false positive (fp) and false negative (fn) using a rectangular window of size *t* seconds.

use the source code provided by the authors of [8] to extract features related to rhythm. We carry out feature selection with a mechanism, adopted from [2], that uses support vector machines to identify the most informative features. For the binary classification of drop vs. non-drop, we use a support vector machine classifier provided in LibSVM. The experiments have been designed to address the two research questions of this paper:

- Can our proposed approach detect drops successfully? (Section 4.3), and
- What is the utility of the timed comments in the limited presence of explicit ground-truth data? (Section 4.4)

4.1 Dataset

In order to evaluate our method, we collect music and social data from SoundCloud, which can be seen as a representative of modern online social audio distribution platforms. It allows users to upload, record and share their self-created music. One of the unique features of SoundCloud is that it allows users to comment at particular time-points in the sound. These comments are referred to as "timed comments". Figure 6 illustrates a screenshot of the audio player on SoundCloud along with the timed comments.

Figure 6. Screenshot of the audio player on SoundCloud.

These comments offer a rich source of information as they are associated with a specific time-point and could indicate useful information about the sound difficult to infer from the signal. Table 1 illustrates a few example timed comments, which provide different kinds of information about the sound. These timed comments can be noisy with respect to their timestamps due to discrepancies between when users hear interesting events, and when they comment on them.

SoundCloud provides a well-documented API that can be used to build applications using SoundCloud data and information on select social features. In order to collect our dataset, we used the Python SDK to search for recent sounds belonging to the following three sub-genres of EDM: *dubstep*, *electro* and *house*. Using the returned list of track identification numbers, we download the track (if its allowed by the user who uploaded the sound) and the corresponding timed comments. We then filter out the comments which do not contain the word "drop". At the end of the data collection process, we have a set of tracks belonging to the above mentioned genres, the associated timed comments containing the word "drop", and the corresponding timestamp of the comment. Table 2 provides some statistics of the dataset.

Timestamp	Comment
01:21	Dunno what it is about this song, inspires me to make more tunes though! love it!
00:28	Love the rhythm!!
00:49	love that drop! nice bassline! nice vocals! epic!

Table 1. Examples of timed comments on SoundCloud.

Genre	# files	Aver. Duration	Aver. # comments	Aver. # drop comments	Aver. # drops
Dubstep	36	4 min.	278	4	3
Electro	36	3.6 min.	220	3	3
House	28	3.9 min.	250	5	2

Table 2. Statistics of the dataset

As we have filtered out the non-drop comments and all the tracks in the dataset have at least one drop comment, we can assume that there is at least one drop in each track. We use a dataset of 100 tracks with a split of 60–20–20 for the training, development and testing respectively.

4.2 Ground-truth annotations

As we are developing a learning framework to detect drops in an EDM track, we need reference labels for the time-points at which drops occur in our dataset, as mentioned previously. We utilize two sources of information: explicit ground-truth (high fidelity labels) and implicit ground-truth (user comments). In order to obtain high fidelity drop labels, one of the authors has listened to the 100 tracks and manually marked the drop points. The labeled points refer to the point where the buildup ends and the bassline is re-introduced. Instead of listening to the entire track, the author skips 30 seconds after he hears a drop as it is highly unlikely that a second drop would occur within 30 seconds. It took approximately 6 hours for the author to label the entire dataset. When computing F1-score in the experiments, we use the manual labels as ground-truth.

Explicit ground-truth labels are expensive as creating them requires experts to spend time and effort to listen to the tracks and mark the drop points. Relying on explicit ground-truth data also hampers the scalability of the dataset, as it would require much more time and effort from the annotators for a larger dataset. Keeping with the social nature of SoundCloud, users contribute comments, some which remark on the pretense or quality of a drop (Table 1). We investigate the possibility of using these timed comments as weak reference labels in predicting the drop. We refer to timed comments as *weak* reference labels owing to their noisy nature. For example, only 20 % of the drop comments in the training set are located at the actual drop in a track. Note that we treat each comment as a distinct

drop. We have a total of 190 drops and 225 drop comments in our dataset. As we can see, there are more comments than the actual drops. Mapping multiple drop comments, which are nearer to each other, to a single time point is a consideration for the future.

4.3 Detecting drop using content-based features

In this experiment, we evaluate the performance of the content based features in detecting a drop using the explicit ground-truth labels. We compute the F1-score for each track separately. The F1-score is averaged if there is more than one drop in the track. In Table 3, we report three results: (1) F1-score, averaged across the entire dataset; (2) Highest F1-score for a single track and (3) Lowest F1-score for a single track. As mentioned before, we use windows of sizes *t = 3, 5, 7, 9, 11, 13, 15 sec*. The size of the window (t) represents the temporal precision to which the F1-score is reported. Observing the results for the average performance (first row of Table 3), we achieve a maximum F1 score of 0.71 for a 15 second tolerance window. However, we already achieve an F1 score greater than 0.6 for a tolerance window as small as 3 seconds. The second row of Table 3 illustrates the F1 scores for one single track which has the best drop detection and we observe that the F1 scores are high and go up to 0.96 for a 15 second tolerance window. The third row of Table 3 illustrates the F1 scores for one single track which has the worst drop detection and we observe that the F1 scores are very low, as it has more false positives. Moreover, the structure segment boundaries do not capture the drops particularly well in this track.

4.4 Utility of timed comments

Timed comments are an important source of information as they could indicate the time point where a drop occurs. Figure 7 illustrates a pipeline for the experiment to assess the utility of timed comments as weak reference labels. It is carried out in three stages labeled as (1), (2) and (3) in the figure. The stages are explained here. We divide the complete training set of N tracks into two mutually exclusive sets of n and $N - n$ tracks. Assuming that the n tracks have ground-truth labels, we train a model (1) and use it to classify the unlabeled segment boundaries from the $N - n$ tracks. We segment boundaries labeled positive by the classifier, which will be of low fidelity, and add them it to the training data. In the second stage (2), we use the expanded training data (n tracks + low fidelity positive segment boundaries) to predict the drop segments in the test set and compute the F1 score for evaluation. Then, the features computed from a window sampled around user drop comments are added to the training data. The data now includes features from the n tracks, and low fidelity predicted positive segment boundaries, and around sampled at user comments. We use this data to train a model (3) and use it to predict the drop segments in the test set and compute the F1 score for evaluation.

In this experiment, we use the following training data sizes which are expressed in terms of the number of tracks:

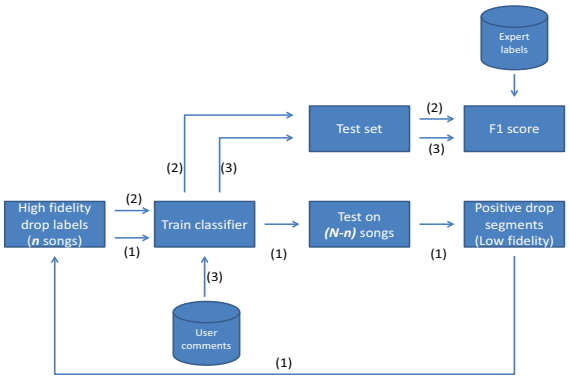


Figure 7. Procedure to assess the utility of timed comments in detecting drop.

$n = 5, 10, 20, 30, 40, 50$. F1 scores over different window sizes are computed to demonstrate the drop detection performance. Figure 8 illustrates the performance of the binary classifier when we have increasing sizes of training data. Due to space constraints, we illustrate the results only for one size of the tolerance window: 11 seconds. Difference in F1 scores when we add user comments is visualized in Figure 8. Inspecting the figure, we can say that

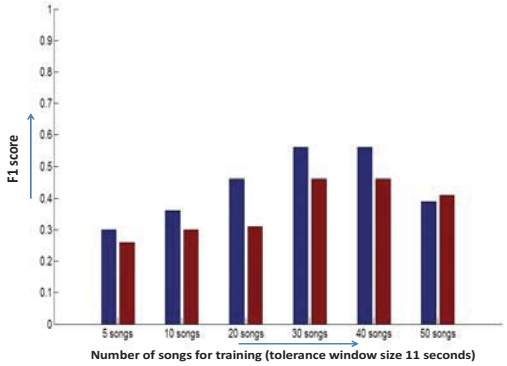


Figure 8. F1 scores for combining high fidelity ground-truth labels and user comments for a tolerance window size of 11 seconds and different training set sizes: 5 tracks, 10 tracks, 20 tracks, 30 tracks, 40 tracks, 50 tracks. First bar in each group indicates the results of stage (3) of the experiment and the second bar indicates the F1 score for the stage (2) of the experiment

reasonable F1 scores are obtained when we use $n = 30$ and $n = 40$ tracks as training set and a tolerance window size of 11 seconds. We observe that the F1 scores are lower than with explicit ground-truth annotations, which we attribute to the noise of user comments.

5. CONCLUSION AND OUTLOOK

We have proposed and evaluated content-based approach that detects an important music event in EDM referred to as a drop. To this end, we have made use of music and user-contributed timed-comments from an online social audio

	3 sec	5 sec	7 sec	9 sec	11 sec	13 sec	15 sec
Average Perfermance	0.61	0.62	0.66	0.66	0.68	0.69	0.71
Track with Best Performanc	0.83	0.9	0.92	0.94	0.95	0.96	0.96
Track with Worst Performance	0.2	0.36	0.43	0.47	0.49	0.51	0.52

Table 3. Experimental results indicating the average, best and worst F1 scores for increasing window sizes

distribution platform: SoundCloud. We reported performance in terms of F1, using a tolerance window of varying time resolutions around the reference drop time-points and the drop time-points hypothesized by our approach. With a tolerance window of 5 seconds, which we estimate to be an acceptable size to listeners, we obtain an F1 score greater than 0.6. "Timed-comments", contributed by users in association with specific time-codes were demonstrated to be useful as weak labels to supplement hand-labeled reference data. We achieved a reasonable accuracy using a standard set of music related features. One of the future steps would be to come up with a set of features which can model the variability and the temporal structure during drop events, which will in turn improve the accuracy. We concentrated on a subset of genres: dubstep, electro and house in this paper as these were the more popular genres on SoundCloud (in terms of number of comments). An immediate direction would be to expand the current dataset by including various sub-genres of EDM, e.g., techno and drum & bass.

Our work demonstrates that musical events in popular electronic music can be successfully analyzed with the help of time-level social comments contributed by users in online social sharing platforms. This approach to music event detection opens up new vistas for future research. Our next step is to carry out a user study with our drop detector aimed at discovering exactly how it can be of use to EDM artists and listeners. Such a study could also reveal the source of "noise" in the timed comments, allowing us to understand why users often comment about drops in neighborhoods far from where an actual drop has occurred. This information could in-turn allow us to identify the most useful drop comments to add to our training data. Further, we wish to widen our exploration of information sources that could possibly support drop detection to also include MIDI files that are posted by users online together with the audio. Currently, the availability of these files is limited, but we anticipate that they might be helpful for bootstrapping. Another source of information is a crowdsourcing, which could be used to identify drops directly, or to filter comments directly related to the drop, from less-closely related or unrelated comments.

6. ACKNOWLEDGEMENT

This research is supported by funding from the European Commission's 7th Framework Program under grant agreement no. 610594 (CrowdRec) and 601166 (PHENICX).

7. REFERENCES

[1] M.J. Butler. *Unlocking the Groove: Rhythm, Meter, and Musical Design in Electronic Dance Music*. Profiles in popular music. Indiana University Press, 2006.

[2] Yi-Wei Chen and Chih-Jen Lin. Combining SVMs with various feature selection strategies. In *Feature Extraction*, volume 207 of *Studies in Fuzziness and Soft Computing*, pages 315–324. Springer Berlin Heidelberg, 2006.

[3] Nick Collins. Influence in early electronic dance music: An audio content analysis investigation. In *ISMIR*, pages 1–6, 2012.

[4] Douglas Eck, Paul Lamere, Thierry Bertin-Mahieux, and Stephen Green. Automatic generation of social tags for music recommendation. In *NIPS*, 2007.

[5] Jason Hockman, Matthew E. P. Davies, and Ichiro Fujinaga. One in the jungle: Downbeat detection in hardcore, jungle, and drum and bass. In *ISMIR*, pages 169–174, 2012.

[6] Thor Kell and George Tzanetakis. Empirical analysis of track selection and ordering in electronic dance music using audio feature extraction. In *ISMIR*, pages 505–510, 2013.

[7] Olivier Lartillot and Petri Toiviainen. Mir in matlab (ii): A toolbox for musical feature extraction from audio. In *ISMIR*, pages 127–130, 2007.

[8] Thomas Lidy and Andreas Rauber. Evaluation of feature extractors and psycho-acoustic transformations for music genre classification. In *ISMIR*, pages 34–41, 2005.

[9] Erik M. Schmidt and Youngmoo E. Kim. Modeling musical emotion dynamics with conditional random fields. In *ISMIR*, pages 777–782, 2011.

[10] Joan Serrà, Meinard Müller, Peter Grosche, and Josep LLuis Arcos. Unsupervised music structure annotation by time series structure features and segment similarity. *IEEE Transactions on Multimedia*, PP(99):1–1, 2014.

[11] Yading Song, Simon Dixon, Marcus Pearce, and Andrea R. Halpern. Do online social tags predict perceived or induced emotional responses to music? In *ISMIR*, pages 89–94, 2013.

[12] John Steventon. *DJing for Dummies*. –For dummies. Wiley, 2007.

[13] Yi-Hsuan Yang, Dmitry Bogdanov, Perfecto Herrera, and Mohamed Sordo. Music retagging using label propagation and robust principal component analysis. In *WWW*, pages 869–876, 2012.

TOWARDS AUTOMATIC CONTENT-BASED SEPARATION OF DJ MIXES INTO SINGLE TRACKS

Nikolay Glazyrin
Ural Federal University
nglazyrin@gmail.com

ABSTRACT

DJ mixes and radio show recordings constitute an important and underexploited music and data source. In this paper we try to approach the problem of separation of a continuous DJ mix into single tracks or timestamping a mix. Sharing some aspects with the task of structural segmentation, this problem has a number of distinctive features that make difficulties for structural segmentation algorithms designed to work with a single track. We use the information derived from spectrum data to separate tracks from each other. We show that the metadata that usually comes with DJ mixes can be exploited to improve the separation. An iterative algorithm that can consider both content-based data and user provided metadata is proposed and evaluated on a collection of freely available timestamped DJ mix recordings of various styles.

1. INTRODUCTION

DJ mixes provide a great source of music data, which does not gain much attention from the MIR community yet. The work by Kell and Tzanetakis [6], which gives an analysis of track selection and ordering in DJ mixes is one of the few exceptions.

Besides playing in clubs many DJs nowadays produce weekly radio shows with latest and greatest and sometimes exclusive tracks. These shows are often freely available through the internet and are very popular among electronic music lovers. Tracklists for the shows are often provided by DJs themselves or by their fans.

For many people it is important to know which track is playing now. The cue sheet file format [2] suits perfectly to carry this kind of information. It was designed to describe how the tracks on CD are laid out, but later it was supported by many audio players and CD burning software. There are communities, such as `http://cuenation.com` or `http://themixingbowl.org`, which bring together the people who create cue sheets for DJ mixes and radio shows. But the wiki page [1] on the first site says nothing about any tools for automatical or semi-automatical generation of cue sheets.

The most time consuming part of this process is finding the moments when one track gives place to another. This may be a big problem for an untrained listener, because making smooth transitions between tracks is one of the skills every DJ should have. For a trained person it is not so hard, but to find a precise position of a transition one has to listen carefully through dozens of seconds of the audio. A tool that can propose most probable transition positions can facilitate this task. Such a tool can also be used by DJs who upload their mixes to special sharing services or online radio stations. These services will be able to timestamp the mix automatically instead of forcing the uploader to do this. The timestamps may be then used to provide fast access to particular tracks within the mix and to easily share previews of unreleased tracks played in radio shows. Timestamped recordings of DJ mixes can be used by recommendation systems to calculate content-based features and relate them to sequential tracks.

The task of DJ mix separation is essentially the task of audio segmentation, so the concepts and approaches can be shared between these tasks. But some conditions and requirements make them different. These differences will be discussed in section 2. In section 3 we describe the proposed method to separate tracks in DJ mix recordings. In section 4 we describe the experiments and the evaluation methodology. Finally, in section 5 we conclude and formulate open problems and directions for future work.

2. PROBLEM FORMULATION AND RELATED WORK

Music structural segmentation is a very popular and elaborated task. Paulus et al. in [9] distinguish three different classes of music segmentation methods. Repetition-based methods try to identify recurring patterns. Novelty-based methods try to find transitions between contrasting parts. Homogenity-based methods, contrary to novelty-based ones, try to determine fragments that are consistent with respect to some characteristic. Combined methods have also been proposed. Some recent ones try to combine novelty-based and homogenity-based approaches [4] or combine novelty-based approach with harmonical information in a joint probabilistic model [10].

A DJ mix can be viewed as a very long composition of individual tracks. These tracks constitute the segments in our task. It is important that no track can occur more than once within a typical mix. So repetition-based methods are

not suitable at the level of tracks.

Novelty-based approach seems to be the most suitable for track boundaries detection. Algorithms that implement this approach generally have 2 main steps: segmentation and grouping.

Segmentation is usually done using an intermediate representation in the form of self-similarity matrix (or self-distance matrix). Since the original audio is not very informative, it needs to be transformed into a sequence of feature vectors, for which this matrix is calculated. The list of features often used for this includes MFCCs, constant-Q spectrum, various low-level spectrum features, such as spectral centroid, spectral spread and others.

The most popular method of obtaining initial segmentation from a self-similarity matrix was proposed by Foote [3]. It is based on so called checkerboard novelty kernels, which are essentially an $M \times M$ matrix with checkerboard-like structure. Novelty estimations can be obtained by convolving this kernel along the main diagonal of the self-similarity matrix. Peaks of the resulting novelty function provide the initial segment borders.

Homogenity-based methods come up as a direct continuation of this novelty-based segmentation. They group similar segments together. A good review of the whole variety of methods can be found in [9]. Many of them perform clustering of segments, e.g. [7], [5]. Any information about the desired result can be helpful at this stage to build the most effective grouping procedure.

In case of DJ mix separation the grouping procedure becomes especially important. It is quite common for dance compositions to have a so called "break" in the middle, where the sound can change dramatically. Such breaks should be overcome to properly detect track boundaries. At the same time, two adjacent segments that belong to different tracks should not be joined.

A typical DJ mix lasts considerably longer than a typical musical composition. So the method must be able to work with recordings that span hours of audio. On the other hand, this loosens the requirements to border detection: an error of seconds or sometimes even tens of seconds can be acceptable. Even humans can have different opinions about one exact moment when a track has transitioned to the next one. An interesting task of detecting transition periods (where two or more tracks are playing simultaneously) comes up here, but we don't consider it in this paper. Marolt in [8] works with similar time scale and boundaries requirements, but with a limited set of possible segment types that sound quite differently.

Transitions can vary significantly for different music styles. It is more likely to find sharp cuts in drum'n'bass mixes, than in deep house mixes, which tend to have long gradual transitions. Average track length is also dependent on music style. But these are generally not the strict rules.

Radio shows often have an intro, which is played in the beginning and often becomes a part of the tracklist. Jingles, interludes or talks where the music gets faded can occur at random places within a recording. But it is not required to discriminate them, as they usually don't get included into tracklists.

The existence of tracklists also makes a great difference from structural segmentation task. It can be seen from the potential applications described in section 1, that the separation of a DJ mix is not much valuable per se. But it becomes really useful when it can be connected with metadata: artist name and track title. Because this metadata is often available, it can also be used in the algorithm. For example, the information about the number of segments in the separation gives a barrier to segmentation and/or grouping process. And if a large music base is available to the algorithm, parts of a mix can be matched to corresponding music recordings to provide even better estimation of track borders.

There may be the cases where matching is not possible though. Sometimes DJs play tracks that are not yet released officially, and therefore cannot appear in any catalogue or database. Some tracks never get released officially. Some tracks have been released years ago, and it's almost impossible to obtain rights on them or find them in any database. That is why the development of the informed automatic DJ mix separation system cannot be reduced to a number of calls to track identification software.

Therefore, further we will suppose that there is a tracklist available for a mix recording, but not the timestamps, and no identification software is available. And the task will be to determine those timestamps based on the audio data and the information from the tracklist, or to align the mix tracklist to the audio. The authors are not informed about any works on this task existing at the moment.

3. SYSTEM DESCRIPTION

We adopt the approach based on novelty-based segmentation followed by grouping of similar segments.

3.1 Features

Constant-Q log-spectrograms are calculated at first for audio recordings, which sampling frequency has been left at the default value of 44100 Hz. We used Constant Q plugin from Queen Mary vamp plugin set [1] with the following parameters: step size and block size are both equal to 16384 samples (0.37 s), 12 components per octave, spanning MIDI pitches from 36 to 84 (65 to 936 Hz) with tuning frequency of 440 Hz. A relatively large block size and zero overlap have been chosen because of the large time scale and to speed up computations. Low frequencies are captured, because electronic dance music often has very accented bass that changes from track to track. The upper frequency limit has been chosen rather arbitrarily, and we do not investigate its influence in this study.

A sliding 2D median filter is then applied to spectrogram with window size (31, 1) (which corresponds to 11.5 seconds and 1 spectral component) to smooth it.

[1] http://www.vamp-plugins.org/plugin-doc/qm-vamp-plugins.html

3.2 Segmentation

To accelerate calculations, the self-distance matrix is calculated for a spectrogram with 10 times less resolution by time axis (3.7 s per column), where each 10 sequential columns of original spectrogram are replaced with their average. We also restricted it to only include cosine distances between segments which are no more than 10 minutes apart from each other, because it is very unlikely to meet a track that lasts longer than that in a DJ mix.

Novelty score is then calculated from the self-distance matrix using the checkerboard kernels with gaussian taper proposed in [3]. We used relatively small kernels of size 16 (composed of 4 squares of size 8×8). All the peaks of the resulting novelty function form the initial set of borders.

3.3 Clustering

Here we find a use for the information from the mix tracklist. The total number of tracks provides the desired number of clusters. This is an imporant advantage over the traditional segmentation task, where the number of segments is unknown. On the other hand, there is a very strong requirement to the borders between segments. If one true border is not detected or one false border is detected in the beginning of the mix, all the subsequent tracks become misaligned with the real audio, even if all the other borders are detected perfectly.

Another piece of information from the tracklist that can be used here is the presence of intro and outro. Many radioshows and regular podcasts have such an intro, fewer ones have also an outro. These segments are relatively short (shorter than 1 minute), but are often included in tracklists. A reasonable assumption is that if the name of the first track contains the string *intro* and/or the name of the last track contains the string *outro*, then an intro and/or an outro should be expected. A good clustering algorithm could be able to detect them automatically, but we add a special handling for these cases. If an intro is expected, among the novelty function peaks during the first 60 seconds of audio the highest one is selected and declared as the intro right border. The same is done at the end of the recording if an outro is expected there.

For the remainder of the recording an iterative clustering procedure is applied. Within each segment the average of all its feature vectors is calculated and normalized by dividing all its components by the maximal one. All the pairwise distances between segments whose beginnings are not more than 600 seconds away from each other are calculated as Euclidean distances between their average feature vectors. This gives a Segment Distance Matrix similar to the one introduced in [4].

All the segment pairs $((l_i, r_i), (l_j, r_j)),\ i < j$ (where l_i and r_i are correspondingly segment's left and righ borders) for which the distance was calculated are sorted according to the following condition: $D_{ij} \cdot (r_j - l_i)$, where D_{ij} is the distance between i-th and j-th segments. Only the pair that produces the smallest value is then merged. If the segments from this pair are not contiguous, all the intermediate ones are also included. To avoid too big segments, a pair gets a penalty when $r_j - l_i > 1.25 \cdot average\_track\_length$: its condition becomes $100000 \cdot (r_j - l_i)$.

4. RESULTS

The proposed method was evaluated on a collection of 103 DJ mix recordings [2] downloaded from free online sources. The corresponding timestamped tracklists in the form of `.cue` files were downloaded from `http://cuenation.com` and used without any correctons. Timestamps have only been used to validate the correctness of track separation. All recordings were taken from different radio shows and live sessions of different disk jokeys. Most of recordings are dated 2014, but there were also recordings from 2007-2013. The dominant music style within the selected recordings is trance (uplifting, progressive, big room, psychedelic), probably due to overall popularity of DJs playing this music. But house, drum'n'bass, breakbeat, techno, hardstyle, downtempo mixes are also included.

For the reasons described in section 3.3 we pay less attention to the conventional precision and recall metrics. Instead, two values have been calculated for each mix: the average and the maximum absolute distances in seconds from true track beginnings to detected ones. This way we can evaluate the usefulness of the method in real life applications: if the average absolute distance approaches the average track length within a mix, the method becomes nearly useless for this mix. The maximum absolute distance gives an estimation of the worst case. These values are then averaged across the whole collection to give an integral measure of method performance.

Frame-based pairwise precision, recall and F-measure have also been calculated to provide more traditional estimation of segmentation quality. They are defined as follows. Each recording is separated into 1 second frames. All frame pairs where both frames belong to the same track form the sets P_E (for the system result) and P_A (for the ground truth). The *pairwise precision rate* can be calculated by $P = \frac{|P_E \cap P_A|}{|P_E|}$, *pairwise recall rate* by $R = \frac{|P_E \cap P_A|}{|P_A|}$, and *pairwise F-measure* by $F = \frac{2PR}{P+R}$. These values are then also averaged across the collection.

As a baseline we will use the same values calculated for the naive separation, where all track borders are evenly spaced within the mix and all tracks have the same duration. In case of explicit intro/outro information the naive separation will allocate them 30 seconds in the beginning or in the end of the mix.

In the first experiment [3] the system was not informed about the presence of intro and outro sections in the mixes. The results are shown in Table 1. The "Good" column shows the number of mixes where the average absolute distance is less than 90 seconds (rather arbitrary limit). From the numbers in this table it seems that the proposed method performs not much better than the naive separation, which

[2] The list of file names is available from `https://github.com/nglazyrin/MixSplitter/blob/master/mix_list.txt`

[3] Full log is available from `https://github.com/nglazyrin/MixSplitter/blob/master/logs/paper_test.log`

Separation	CAvg. abs. dist.	CAvg. max dist.	Good
Proposed	143.73 s	328.99 s	42
Baseline	152.83 s	318.35 s	30

Table 1. Results with no information about intro and outro.

Separation	CAvg. abs. dist.	CAvg. max dist.	Good
Proposed	111.82 s	286.61 s	62
Baseline	126.87 s	284.41 s	49

Table 2. Results with information about intro and outro.

is confirmed by p-value of 0.096 returned by Wilcoxon test. But looking closer at the performance on particular mixes, we can see that in some cases the proposed method has real advantage. E.g. for the mix *M.PRAVDA - Best of 2013 (Part 2) (promodj.com).mp3* it gives average absolute distance of 8.59 s (which is great) versus 60.22 s obtained by the naive separation. On the other hand, for some mixes (e.g. *Trancecoda Podcast 008 - GMix Eddie Bitar.mp3*) the average absolute distance exceeds 6 minutes, which is absolutely inacceptable.

In the second experiment [4] the system was informed about the presence of intro and outro secions and could react appropriately. From the Table 2 we can see that this information can be really helpful. In this experiment the $p < 0.01$ was returned by Wilcoxon test. The result has moved nearer to the "Good" limit of 90 seconds average difference, and the difference between the proposed and the baseline methods became bigger. And if the limit of "goodness" has decreased to 60 seconds, the difference gets more explicit: 54 good separations by the proposed method versus 24 good naive separations. For 30 seconds limit on average absolute difference only 25 versus 6 good separations are left.

This result shows that the proposed method can give good result for a reasonable amount of mixes (62 out of 103 here). But for some mixes the results are still too bad. We provide two case-studies that describe common errors of the method.

Table 3 shows the comparison of true and detected borders for one of the mixes – *4H_Community_Guest_Mix_The_2nd_Anniversary_of_Room51_Show_by_Breeze_Quadrat_PureFM.mp3* – with average absolute difference of 177.11 seconds. First 3 tracks are aligned good, but then the system detects wrong border in the middle of 4th track. In spite of more or less properly detected other borders (the detected value in row $i+1$ is near the true value in row i), they all mark beginnings of track $i+1$ instead of i-th track.

The same information is represented graphically on Figure 1. Vertical yellow lines on the constant Q spectrogram mark the true borders, vertical black lines correspond to detected borders.

The errors of this kind can be overcome with a better

[4] Full log is available from `https://github.com/nglazyrin/MixSplitter/blob/master/logs/paper_test_explicit_intro_outro.log`

No.	Detected	True	Difference
1	0.00 s	0.00 s	0.00 s
2	308.38 s	312.08 s	3.70 s
3	628.57 s	613.00 s	-15.56 s
4	872.56 s	1029.11 s	156.55 s
5	1025.19 s	1363.48 s	338.29 s
6	1360.54 s	1757.29 s	396.75 s
7	1757.15 s	1961.62 s	204.47 s
8	1970.53 s	2292.27 s	321.74 s
9	2321.36 s	2552.34 s	230.98 s
10	2748.30 s	2979.58 s	231.28 s
11	3247.66 s	3198.78 s	-48.88 s

Table 3. Detailed result for the mix by 4H Community.

Separation	Precision	Recall	F-measure
Proposed (1)	0.8145	0.7761	0.7941
Baseline (1)	0.7024	0.6397	0.6688
Proposed (2)	0.8077	0.7892	0.7977
Baseline (2)	0.7069	0.6637	0.6839

Table 4. Framewise precision, recall and F-measure.

sorting function for segment pairs or with a different segment grouping strategy. As can be seen from Table 4 (the number in parentheses in the first column corresponds to the experiment number), the proposed method really locates borders much better than the baseline. But since some borders are misplaced, the final pairwise precision and recall rates are not so close to 1 as they could be.

Another source of errors are mixes that contain tracks of various durations, e.g. a pile of 1 minute long tracks followed by 4 minute long tracks, or several interludes throughout the recording. An example of such mix is *01-friction_-_bbc_radio1_(chase_and_status_special)-sat-10-13-2013-talion.mp3*, which contains 35 tracks per 2 hours, and 6 of them are grouped between 55 and 65 minutes. The separation is shown on the Figure 2. The described method tends to join short segments and to return more or less evenly spaced track borders because of the sorting condition and the penalty for long tracks. So it does not fit to these highly-variable mixes, which are characteristic for music genres such as drum'n'bass. But the separations obtained without using the penalty were worse than the ones obtained by the baseline method.

Table 5 groups the resutls by music genres, which were manually annotated for each mix. The mixes labeled as having *various* genre contain tracks from two or more very different genres, such as house and drum'n'bass. The Cnt column gives the total count of mixes of a given genre within our test collection.

Because the test set is very unbalanced by music genre (which is dictated by the available cue sheet files), it's hard to make conclusions for music genres other than house and trance (which can be themselves separated into various subgenres). The proposed system outperforms the baseline method on these genres, but both methods are failing on

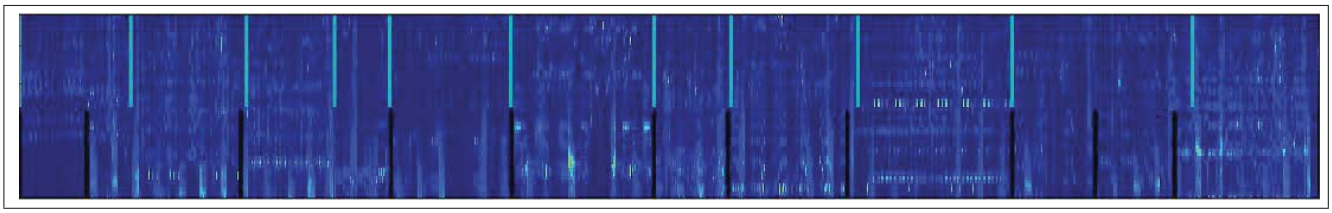


Figure 1. The separation for the mix by 4H Community.

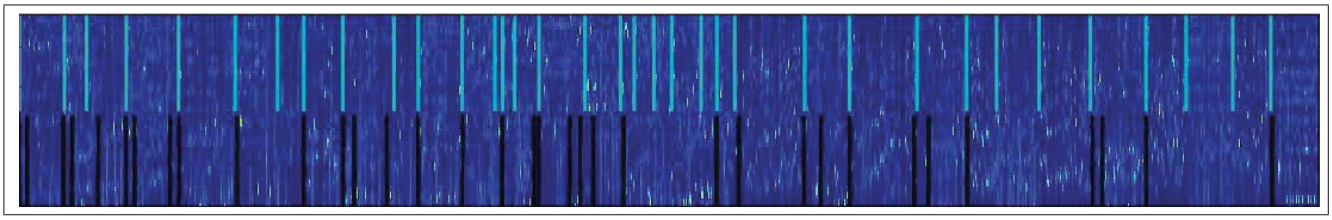


Figure 2. The separation for the mix by Chase & Status.

Style	Cnt	Separation	Abs. dist.	Max dist.
trance	59	Proposed	91.43 s	244.26 s
		Baseline	114.85 s	255.38 s
house	29	Proposed	106.55 s	280.78 s
		Baseline	117.88 s	270.36 s
techno	4	Proposed	122.11 s	284.51 s
		Baseline	104.61 s	219.00 s
downtempo	3	Proposed	304.59 s	702.77 s
		Baseline	308.58 s	609.34 s
hardstyle	2	Proposed	81.91 s	232.66 s
		Baseline	93.22 s	221.95 s
drum'n'bass	2	Proposed	330.67 s	798.21 s
		Baseline	343.03 s	865.92 s
various	2	Proposed	211.28 s	429.65 s
		Baseline	203.85 s	407.48 s
breakbeat	2	Proposed	191.88 s	399.71 s
		Baseline	124.22 s	346.01 s

Table 5. Results by music genre.

downtempo and drum'n'bass music.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a method for informed content-based separation of DJ mixes into single tracks that outperforms a naive baseline evenly separating method. We showed that this method provides good results for a reasonable amount of mixes. The resulting separations are good enough to use them for further applications. We also showed how a simple information about the presence of intro and outro sections in the mix can improve the separation quality.

This paper establishes a basis for further work on DJ mixes separation. Another clustering methods need to be developed to prevent false border detection errors and border miss errors. It makes sense also to include higher frequencies into the initial spectrum, as they may carry some meaningful details. On the other hand, the novelty detection method does not seem to have a major impact, because the initial border candidate set if sufficiently large to select values nearby the true borders.

More feature types need to be exploited. It also makes sense to consider the tempo information to avoid false border detections, because the tempo does not change often during transitions, but changes within a track when a break starts or ends. A deeper modification or a new method is needed to handle mixes that contain tracks with highly-varying durations. A separate method to detect interludes and talks can be helpful here.

Finally, a significant improvement may be expected from the usage of a track identification system, as it may help to align at least some of the tracks properly. But this poses a separate technical and legal task.

6. REFERENCES

[1] Online: `http://wiki.themixingbowl.org/Cue_sheet`, acessed on May 5, 2014.

[2] Online: `http://wiki.hydrogenaudio.org/index.php?title=Cue_sheet`, accessed on May 5, 2014.

[3] J. Foote: "Automatic audio segmentation using a measure of audio novelty" *Proceedings of IEEE International Conference on Multimedia and Expo*, Vol. 1, pp. 452–455, 2000.

[4] F. Kaiser, and G. Peeters: "A simple fusion method of state and sequence segmentation for music structure discovery" *Proceedings of the 14th International Society for Music Information Retrieval Conference*, pp. 257-262, 2013.

[5] F. Kaiser, and T. Sikora: "Music Structure Discovery in Popular Music using Non-negative Matrix Factorization" *Proceedings of the 11th International Society for Music Information Retrieval Conference*, pp. 429–434, 2010.

[6] T. Kell and G. Tzanetakis: "Empirical analysis of track selection and ordering in electronic dance music using audio feature extraction" *Proceedings of the 14th International Society for Music Information Retrieval Conference*, pp. 505-510, 2013.

[7] M. Levy, M. Sandler, and M. Casey: "Extraction of High-Level Musical Structure From Audio Data and Its Application to Thumbnail Generation" *Proceedings of the International Conference on Acoustics, Speech and Signal Processing 2006*, Vol. 5, 2006.

[8] M. Marolt: "Probabilistic Segmentation and Labeling of Ethnomusicological Field Recordings" *Proceedings of the 10th International Society for Music Information Retrieval Conference*, pp. 75–80, 2009.

[9] J. Paulus, M. Müller, and A. Klapuri: "Audio-based Music Structure Analysis" *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, pp. 625–636, 2010.

[10] J. Pauwels, F. Kaiser, and G. Peeters: "Combining harmony-based and novelty-based approaches for structural segmentation" *Proceedings of the 14th International Society for Music Information Retrieval Conference*, pp. 601-606, 2013.

MedleyDB: A MULTITRACK DATASET FOR ANNOTATION-INTENSIVE MIR RESEARCH

Rachel Bittner[1], Justin Salamon[1,2], Mike Tierney[1], Matthias Mauch[3], Chris Cannam[3], Juan Bello[1]
[1]Music and Audio Research Lab, New York University
[2]Center for Urban Science and Progress, New York University
[3]Centre for Digital Music, Queen Mary University of London
{rachel.bittner,justin.salamon,mt2568,jpbello}@nyu.edu {m.mauch,chris.cannam}@eecs.qmul.ac.uk

ABSTRACT

We introduce *MedleyDB*: a dataset of annotated, royalty-free multitrack recordings. The dataset was primarily developed to support research on melody extraction, addressing important shortcomings of existing collections. For each song we provide melody f_0 annotations as well as instrument activations for evaluating automatic instrument recognition. The dataset is also useful for research on tasks that require access to the individual tracks of a song such as source separation and automatic mixing. In this paper we provide a detailed description of *MedleyDB*, including curation, annotation, and musical content. To gain insight into the new challenges presented by the dataset, we run a set of experiments using a state-of-the-art melody extraction algorithm and discuss the results. The dataset is shown to be considerably more challenging than the current test sets used in the MIREX evaluation campaign, thus opening new research avenues in melody extraction research.

1. INTRODUCTION

Music Information Retrieval (MIR) relies heavily on the availability of annotated datasets for training and evaluating algorithms. Despite efforts to crowd-source annotations [9], most annotated datasets available for MIR research are still the result of a manual annotation effort by a specific researcher or group. Consequently, the size of the datasets available for a particular MIR task is often directly related to the amount of effort involved in producing the annotations.

Some tasks, such as cover song identification or music recommendation, can leverage weak annotations such as basic song metadata, known relationships or listening patterns oftentimes compiled by large music services such as *last.fm* [1] . However, there is a subset of MIR tasks dealing with detailed information from the music signal for which time-aligned annotations are not readily available, such as the fundamental frequency (f_0) of the melody (needed for melody extraction [13]) or the activation times of the different instruments in the mix (needed for instrument recognition [1]). Annotating this kind of highly specific information from real world recordings is a time consuming process that requires qualified individuals, and is usually done in the context of large annotation efforts such as the Billboard [3], SALAMI [15], and Beatles [8] datasets. These sets include manual annotations of structure, chords, or notes, typically consisting of categorical labels at time intervals on the order of seconds. The annotation process is even more time-consuming for f_0 values or instrument activations for example, which are numeric instead of categorical, and at a time-scale on the order of milliseconds. Unsurprisingly, the datasets available for evaluating these taks are often limited in size (on the order of a couple dozen files) and comprised solely of short excerpts.

When multitrack audio is available, annotation tasks that would be difficult with mixed audio can often be expedited. For example, annotating the f_0 curve for a particular instrument from a full audio mix is difficult and tedious, whereas with multitrack stems the process can be partly automated using monophonic pitch tracking techniques. Since no algorithm provides 100% estimation accuracy in real-world conditions, a common solution is to have experts manually correct these machine annotations, a process significantly simpler than annotating from scratch. Unfortunately, collections of royalty-free multitrack recordings that can be shared for research purposes are relatively scarce, and those that exist are homogeneous in genre. This is a problem not only for evaluating annotation-intensive tasks but also for tasks that by definition require access to the individual tracks of a song such as source separation and automatic mixing.

In this paper we introduce *MedleyDB*: a multipurpose audio dataset of annotated, royalty-free multitrack recordings. The dataset includes melody f_0 annotations and was primarily developed to support research on melody extraction and to address important shortcomings of the existing collections for this task. Its applicability extends to research on other annotation-intensive MIR tasks, such as instrument recognition, for which we provide instrument activations. The dataset can also be directly used for re-

[1] http://www.last.fm

search on source separation and automatic mixing. Further track-level annotations (e.g. multiple f_0 or chords) can be easily added in the future to enable evaluation of additional MIR tasks.

The remainder of the paper is structured as follows: in Section 2 we provide a brief overview of existing datasets for melody extraction evaluation, including basic statistics and content. In Section 3 we provide a detailed description of the *MedleyDB* dataset, including compilation, annotation, and content statistics. In Section 4 we outline the types of annotations provided and the process by which they were generated. In Section 5 we provide some insight into the challenges presented by this new dataset by examining the results obtained by a state-of-the-art melody extraction algorithm. The conclusions of the paper are provided in Section 6.

2. PRIOR WORK

2.1 Datasets for melody extraction

Table 1 provides a summary of the datasets commonly used for the benchmarking of melody extraction algorithms. It can be observed that datasets that are stylistically varied and contain "real" music (e.g. ADC2004 and MIREX05) are very small in size, numbering no more than two dozen files and a few hundred seconds of audio. On the other hand, large datasets such as MIREX09, MIR1K and the RWC pop dataset tend to be stylistically homogeneous and/or include music that is less realistic. Furthermore, all datasets, with the exception of RWC, are limited to relatively short excerpts. Note that the main community evaluation for melody extraction, the MIREX AME task, [2] has been limited to the top 4 datasets.

In [14], the authors examined how the aforementioned constraints affect the evaluation of melody extraction algorithms. Three aspects were studied – inaccuracies in the annotations, the use of short excerpts instead of full-length songs, and the limited number of excerpts used. They found that the evaluation is highly sensitive to systematic annotation errors, that performance on excerpts is not necessarily a good predictor for performance on full songs, and that the collections used for the MIREX evaluation [5] are too small for the results to be statistically stable. Furthermore, they noted that the only MIREX dataset that is sufficiently large (MIREX 2009) is highly homogeneous (Chinese pop music) and thus does not represent the variety of commercial music that algorithms are expected to generalize to. This finding extrapolates to the MIR1K and RWC sets.

To facilitate meaningful future research on melody extraction, we sought to compile a new dataset addressing the following criteria:

1. **Size**: the dataset should be at least one order of magnitude greater than previous heterogeneous datasets such as ADC2004 and MIREX05.

[2] http://www.music-ir.org/mirex/wiki/Audio_Melody_Extraction

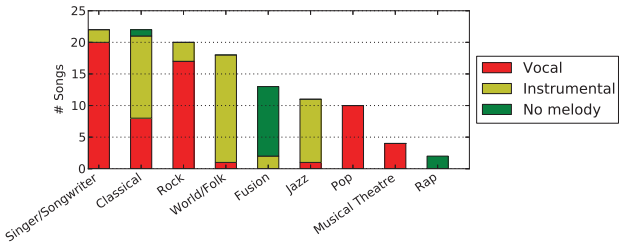


Figure 1. Number of songs per genre with breakdown by melody source type.

2. **Duration**: the dataset should primarily consist of full length songs.
3. **Quality**: the audio should be of professional or near-professional quality.
4. **Content**: the dataset should consist of songs from a variety of genres.
5. **Annotation**: the annotations must be accurate and well-documented.
6. **Audio**: each song and corresponding multitrack session must be available and distributable for research purposes.

2.2 Multitrack datasets

Since we opted to use multitracks to facilitate the annotation process, it is relevant to survey what multitrack datasets are currently available to the community. The TRIOS [6] dataset provides 5 score-aligned multitrack recordings of musical trios for source separation, the MASS [3] dataset contains a small collection of raw and effects-processed multitrack stems of musical excerpts for work in source separation, and the Mixploration dataset [4] for automatic mixing contains 24 versions of four songs. These sets are too small and homogeneous to fit our criteria; the closest candidate is the Structural Segmentation Multitrack Dataset [7] which contains 103 rock and pop songs with structural segmentation annotations. While the overall size of this dataset is satisfactory, there is little variety in genre and the dataset is not uniformly formatted, making batched processing difficult or impossible.

Since no sufficient multitrack dataset currently exists, we curated *MedleyDB* which fits our needs and can be used for other MIR tasks as well, and is described in detail in the following section.

3. DATASET

3.1 Overview

The dataset consists of 122 songs, 108 of which include melody annotations. The remaining 14 songs do not have a discernible melody and thus were not appropriate for melody extraction. We include these 14 songs in the dataset because of their use for other applications including instrument ID, source separation and automatic mixing.

[3] http://mtg.upf.edu/download/datasets/mass

Name	# Songs	Song duration	Total duration	% Vocal Songs	Genres	Content
ADC2004	20	∼20 s	369 s	60%	Pop, jazz, opera	Real recordings, synthesized voice and MIDI
MIREX05	25	∼10–40 s	686 s	64%	Rock, R&B, pop, jazz, solo classical piano	Real recordings, synthesized MIDI
INDIAN08	8	∼60 s	501 s	100%	North Indian classical music	Real recordings
MIREX09	374	∼20–40 s	10020 s	100%	Chinese pop	Recorded singing with karaoke accompaniment
MIR1K	1000	∼10 s	7980 s	100%	Chinese Pop	Recorded singing with karaoke accompaniment
RWC	100	∼240 s	24403 s	100%	Japanese Pop, American Pop	Real recordings
MedleyDB	108	∼20–600 s	26831 s	57%	Rock, pop, classical, jazz, rock, pop, fusion, world, musical theater, singer-songwriter	Real recordings

Table 1. Existing collections for melody extraction evaluation (ADC2004 through RWC) and the new *MedleyDB* dataset.

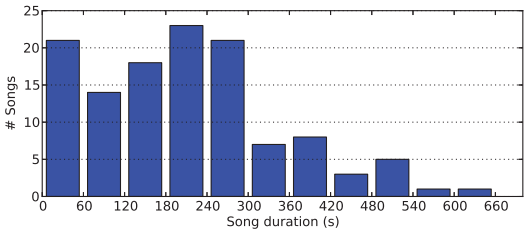


Figure 2. Distribution of song durations.

Each song in the dataset is freely available online [4] under a Creative Commons Attribution - NonCommercial - ShareAlike 3.0 Unported license [5], which allows the release of the audio and annotations for non-commercial purposes.

We provide a stereo mix and both dry and processed multitrack stems for each song. The content was obtained from multiple sources: 30 songs were provided by various independent artists, 32 were recorded at NYU's Dolan Recording Studio, 25 were recorded by Weathervane Music [6], and 35 were created by Music Delta [7]. The majority of the songs were recorded in professional studios and mixed by experienced engineers.

In Figure 1 we give the distribution of genres present within the dataset, as well as the number of vocal and instrumental songs within each genre. The genres are based on nine generic genre labels. Note that some genres such as Singer/Songwriter, Rock and Pop are strongly dominated by vocal songs, while others such as Jazz and World/Folk are mostly instrumental. Note that the Rap and most of the Fusion songs do not have melody annotations. Figure 2 depicts the distribution of song durations. A total of 105 out of the 122 songs in the dataset are full length songs, and the majority of these are between 3 and 5 minutes long. Most recordings that are under 1 minute long were created by Music Delta. Finally, the most represented instruments in the dataset are shown in Figure 3. Unsurprisingly, drums, bass, piano, vocals and guitars dominate the distribution.

[4] http://marl.smusic.nyu.edu/medleydb

[5] http://creativecommons.org/licenses/by-nc-sa/3.0/deed.en_US

[6] http://weathervanemusic.org/

[7] http://www.musicdelta.com/

3.2 Multitrack Audio Structure

The structure of the audio content in *MedleyDB* is largely determined by the recording process, and is exemplified in Figure 4, which gives a toy example of how the data could be organized for a recording of a jazz quartet.

At the lowest level of the process, a set of microphones is used to record the audio sources, such that there may be more than one microphone recording a single source – as is the case for the piano and drum set in Figure 4. The resulting files are *raw* unprocessed mono audio tracks. Note that while they are "unprocessed", they are edited such that there is no content present in the raw audio that is not used in the mix. The raw files are then grouped into stems, each corresponding to a specific sound source: double bass, piano, trumpet and drum set in the example. These *stems* are stereo audio components of the final mix and include all effects processing, gain control, and panning. Finally, we refer to the *mix* as the complete polyphonic audio created by mixing the stems and optionally mastering the mix.

Therefore, a song consists of the *mix*, *stems*, and *raw audio*. This hierarchy does not perfectly model every style of recording and mixing, but it works well for the majority of songs. Thus, the audio provided for this dataset is organized with this hierarchy in mind.

3.3 Metadata

Both song and stem-level metadata is provided for each song. The song-level metadata includes basic information about the song such as the artist, title, composer, and website. Additionally, we provide genre labels corresponding to the labels in Figure 1. Some sessions correspond to

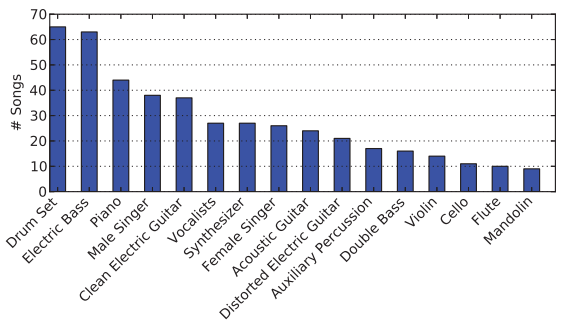


Figure 3. Occurrence count of the most frequent instruments in the dataset.

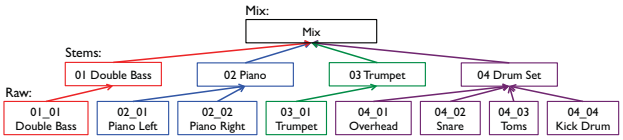


Figure 4. The hierarchy of audio files for a jazz quartet.

recordings of ensembles, where the microphones may pick up sound from sources other than the one intended, a phenomenon known as *bleeding*. Because bleed can affect automated annotation methods and other types of processing, songs that contain any stems with bleed are tagged.

Stem-level metadata includes instrument labels based on a predefined taxonomy given to annotators, and a field indicating whether the stem contains melody.

The metadata is provided as a YAML [8] file, which is both human-readable as a text file, and a structured format that can be easily loaded into various programming environments.

4. ANNOTATIONS

4.1 Annotation Task Definitions

When creating annotations for *MedleyDB*, we were faced with the question of what definition of melody to use. The definition of melody used in MIREX 2014 defines melody as the predominant pitch where, "pitch is expressed as the fundamental frequency of the main melodic voice, and is reported in a frame-based manner on an evenly-spaced time-grid." Many of the songs in the dataset do not reasonably fit the definition of melody used by MIREX because of the constraint that the melody is played by a single voice, but we felt that the annotations should have consistency with the existing melody annotations.

Our resolution was to provide melody annotations based on three different definitions of melody that are in discussion within the MIR community. [9] In the definitions we consider, melody is defined as:

1. The f_0 curve of the predominant melodic line drawn from a single source.
2. The f_0 curve of the predominant melodic line drawn from multiple sources.
3. The f_0 curves of all melodic lines drawn from multiple sources.

Definition 1 coincides with the definition for the melody annotations used in MIREX. This definition requires the choice of a lead instrument and gives the f_0 curve for this instrument. Definition 2 expands on definition 1 by allowing multiple instruments to contribute to the melody. While a single lead instrument need not be chosen, an indication of which instrument is predominant at each point in time is required to resolve the f_0 curve to a single point at each time frame. Definition 3 is the most complex, but also the most general. The key difference in this definition is that at a given time frame, multiple f_0 values may be "correct".

For instrument activations, we simply assume that signal energy in a given stem, above a predefined limit, is indicative of the presence of the corresponding instrument in the mix. Based on this notion, we provide two types of annotations: a list of time segments where each instrument is active; and a matrix containing the activation confidence per instrument per unit of time.

4.2 Automatic Annotation Process

The melody annotation process was semi-automated by using monophonic pitch tracking on selected stems to return a good initial estimate of the f_0 curve, and by using a voicing detection algorithm to compute instrument activations. The monophonic pitch tracking algorithm used was pYIN [11] which is an improved, probabilistic version of the well-known YIN algorithm.

As discussed in the previous section, for each song we provide melody annotations based upon the 3 different definitions. The melody annotations based on Definition 1 were generated by choosing the single most dominant melodic stem. The Definition 2 annotations were created by sectioning the mix into regions and indicating the predominant melodic stem within each region. The melody curve was generated by choosing the f_0 curve from the indicated instrument at each point in time. The Definition 3 annotations contain the f_0 curves from each of the annotated stems.

The annotations of instrument activations were generated using a standard envelope following technique on each stem, consisting of half-wave rectification, compression, smoothing and down-sampling. The resulting envelopes are normalized to account for overall signal energy and total number of sources, resulting in the $t \times m$ matrix H, where t is the number of analysis frames, and m is the number of instruments in the mix. For the i^{th} instrument, the confidence of its activations as a function of time can be approximated via a logistic function:

$$C(i,t) = 1 - \frac{1}{1 + e^{(H_{it} - \theta)\lambda}}. \quad (1)$$

where λ controls the slope of the function, and θ the threshold of activation. Frames where instrument i is considered active are those for which $C(i,t) \geq 0.5$. No manual correction was performed on these activations.

Note that monophonic pitch tracking, and the automatic detection of voicing and instrument activations, fail when the stems contain bleed from other instruments, which is the case for 25 songs within the collection. Source separation, using a simple approach based on Wiener filters [2], was used on stems with bleed to clean up the audio before applying the algorithms. The parameters of the separation were manually and independently optimized for each track containing bleed.

[8] http://www.yaml.org/
[9] http://ameannotationinitiative.wikispaces.com

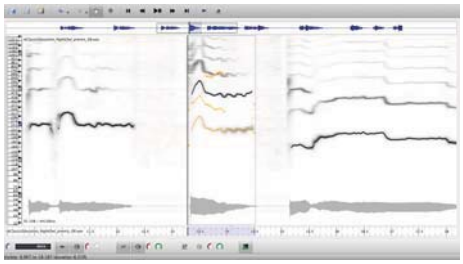


Figure 5. Screenshot of *Tony*. An estimated pitch curve is selected and alternative candidates are shown in yellow.

4.3 Manual Annotation Process

The manual annotation process was facilitated by the use of a recently developed tool called *Tony* [10], which enables efficient manual corrections (see Figure 5). Tony provides 3 types of semi-manual correction methods: (1) deletion (2) octave shifting and (3) alternative candidates.

When annotating the f_0 curves, unvoiced vocal sounds, percussive attacks, and reverb tail were removed. Sections of a stem which were active but did not contain melody were also removed. For example, a piano stem in a jazz combo may play the melody during a solo section and play background chords throughout the rest of the piece. In this case, only the solo section would be annotated, and all other frames would be marked as unvoiced.

The annotations were created by five annotators, all of which were musicians and had at least a bachelor's degree in music. Each annotation was evaluated by one annotator and validated by another. The annotator/validator pairs were randomized to make the final annotations as unbiased as possible.

4.4 Annotation Formats

We provide melody annotations based on the three definitions for 108 out of the 122 songs. Note that while definition 1 is not appropriate for all of the annotated songs (i.e. there are songs where the melody is played by several sources and there is no single clear predominant source throughout the piece), we provide type 1 melody annotations for all 108 melodic tracks so that an algorithm's performance on type 1 versus type 2 melody annotations can be compared over the full dataset. Of the 108 songs with melody annotations, 62 contain predominantly vocal melodies and the remaining 47 contain instrumental melodies.

Every melody annotation begins at time 0 and has a hop size of 5.8 ms (256 samples at $f_s = 44.1$ kHz). Each time stamp in the annotation corresponds to the center of the analysis frame (i.e. the first frame is centered on time 0). In accordance with previous annotations, frequency values are given in Hz, where unvoiced frames (i.e. frames where there is no melody) are indicated by a value of 0 Hz.

We provide instrument activation annotations for the entire dataset. Confidence values are given as matrices where the first column corresponds to time in seconds, starting at 0 with a hop size of 46.4 ms (2048 samples at $f_s = 44.1$ kHz), and each subsequent column corresponds to an instrument identifier. Confidence values are continuous in the range $[0, 1]$. We also provide a list of activations, each a triplet of start time, end time and instrument label.

Dataset	ν	VxR	VxF	RPA	RCA	OA
MDB – All	.2	.78 (.13)	.38 (.14)	.55 (.26)	.68 (.19)	**.54** (.17)
MDB – All	-1	.57 (.20)	.20 (.12)	.52 (.26)	.68 (.19)	**.57** (.18)
MDB – VOC	-1	.69 (.15)	.23 (.13)	.63 (.23)	.76 (.15)	**.66** (.14)
MDB – INS	-1	.41 (.15)	.16 (.09)	.38 (.23)	.57 (.18)	**.47** (.17)
MIREX11	.2	.86	.24	.80	.82	**.75**

Table 2. Performance of Melodia [12] on different subsets of *MedleyDB* (*MDB*) for type 1 melody annotations, and comparison to performance on the MIREX datasets. For each measure we provide the mean with the standard deviation in parentheses.

5. NEW CHALLENGES

To gain insight into the challenges presented by this new dataset and its potential for supporting progress in melody extraction research, we evaluate the performance of the Melodia melody extraction algorithm [12] on the subset of *MedleyDB* containing melody annotations. In the following experiments we use the melody annotations based on Definition 1, which can be evaluated using the standard five measures used in melody extraction evaluation: voicing recall (VxR), voicing false alarm (VxF), raw pitch accuracy (RPA), raw chroma accuracy (RCA), and overall accuracy (OA). For further details about the measures see [13].

In the first row of Table 2 we give the results obtained by Melodia using the same parameters (voicing threshold $\nu = .2$) employed in MIREX 2011 [12]. The first thing we note is that for all measures, the performance is considerably lower on *MedleyDB* than on MIREX11. The overall accuracy is 21 percentage points lower, a first indication that the new dataset is more challenging. We also note that the VxF rate is considerably higher compared to the MIREX results. In the second row of Table 2 we provide the results obtained when setting ν to maximize the overall accuracy ($\nu = -1$). The increase in overall accuracy is relatively small (3 points), indicating that the dataset remains challenging despite using the best possible voicing parameter. In the next two rows of Table 2, we provide a breakdown of the results by vocal vs. instrumental songs. We see that the algorithm does significantly better on vocal melodies compared to instrumental ones, consistent with the observations made in [12]. For instrumental melodies we observe a 19-point drop between raw chroma and pitch accuracy, indicating an increased number of octave errors. The bias in performance towards vocal melodies is likely the result of all previous datasets being primarily vocal.

In Table 3 we provide a breakdown of the results by genre. In accordance with the the previous table, we see that genres with primarily instrumental melodies are considerably more challenging. Finally, we repeat the experiment carried out in [14], where the authors compared performance on recordings to shorter sub-clips taken from the same recordings to see whether the results on a dataset of

Genre	VxR	VxF	RPA	RCA	OA
MUS	.73 (.16)	.14 (.04)	.74 (.18)	.87 (.08)	**.73** (.14)
POP	.74 (.12)	.22 (.09)	.65 (.20)	.73 (.15)	**.69** (.12)
S/S	.66 (.13)	.23 (.12)	.64 (.19)	.74 (.16)	**.66** (.11)
ROC	.71 (.18)	.29 (.15)	.53 (.29)	.73 (.18)	**.59** (.16)
JAZ	.44 (.14)	.12 (.06)	.55 (.17)	.68 (.15)	**.57** (.14)
CLA	.46 (.20)	.15 (.07)	.35 (.30)	.56 (.22)	**.51** (.23)
WOR	.40 (.12)	.18 (.09)	.44 (.19)	.63 (.14)	**.44** (.13)
FUS	.41 (.04)	.17 (.02)	.32 (.07)	.51 (.01)	**.43** (.04)

Table 3. Performance of Melodia [12] ($\nu = -1$) on different genres in *MedleyDB* for type 1 melody annotations. For each measure we provide the mean with the standard deviation in parentheses.

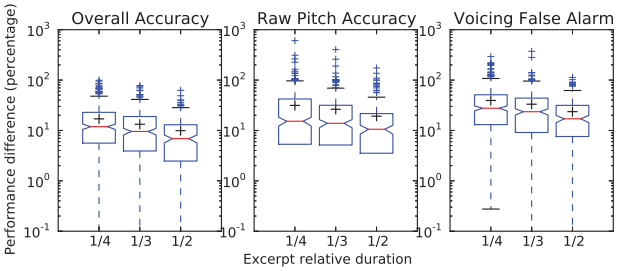


Figure 6. Relative performance differences between full songs and excerpts. The large black crosses mark the means of the distributions.

excerpts would generalize to a dataset of full songs. The novelty in our experiment is that we use full length songs, as opposed to clips sliced into even shorter sub-clips. The results are presented in Figure 6, and are consistent with those reported in [14]. We see that as the relative duration of the excerpts (1/4, 1/3 or 1/2 of the full song) gets closer to 1, the relative difference in performance goes down (significant by a Mann-Whitney U test, $\alpha = 0.01$). This highlights another benefit of *MedleyDB*: since the dataset primarily contains full length songs, one can expect better generalization to real-world music collections. While further error analysis is required to understand the specific challenges presented by *MedleyDB*, we identify (by inspection) some of the musical characteristics across the dataset that make *MedleyDB* more challenging – rapidly changing notes, a large melodic frequency range (43-3662 Hz), concurrent melodic lines, and complex polyphony.

6. CONCLUSION

Due to the scarcity of multitrack audio data for MIR research, we presented *MedleyDB* – a dataset of over 100 multitrack recordings of songs with melody f_0 annotations and instrument activations. We provided a description of the dataset, including how it was curated, annotated, and its musical content. Finally, we ran a set of experiments to identify some of the new challenges presented by the dataset. We noted how the increased proportion of instrumental tracks makes the dataset significantly more challenging compared to the MIREX datasets, and confirmed that performance on excerpts will not necessarily generalize well to full-length songs, highlighting the greater generalizability of *MedleyDB* compared with most existing datasets. Since 2011 there has been no significant improvement in performance on the MIREX AME task. If we previously attributed this to some glass ceiling, we now see that there is still much room for improvement. *MedleyDB* represents a shift towards more realistic datasets for MIR research, and we believe it will help identify future research avenues and enable further progress in melody extraction research and other annotation-intensive MIR endeavors.

7. REFERENCES

[1] J.G.A. Barbedo. Instrument recognition. In T. Li, M. Ogihara, and G. Tzanetakis, editors, *Music Data Mining*. CRC Press, 2012.

[2] L. Benaroya, F. Bimbot, and R. Gribonval. Audio source separation with a single sensor. *IEEE TASLP*, 14(1):191–199, 2006.

[3] J. A. Burgoyne, J. Wild, and I. Fujinaga. An expert ground truth set for audio chord recognition and music analysis. In *ISMIR'11*, pages 633–638, 2011.

[4] M. Cartwright, B. Pardo, and J. Reiss. Mixploration: Rethinking the audio mixer interface. In *19th Int. Conf. on Intelligent User Interfaces*, pages 365–370, 2014.

[5] J. Stephen Downie. The music information retrieval evaluation exchange (2005–2007): A window into music information retrieval research. *Acoustical Science and Technology*, 29(4):247–255, 2008.

[6] J. Fritsch and M. D. Plumbley. Score informed audio source separation using constrained nonnegative matrix factorization and score synthesis. In *IEEE ICASSP'13*, pages 888–891, 2013.

[7] S. Hargreaves, A. Klapuri, and M. Sandler. Structural segmentation of multitrack audio. *IEEE TASLP*, 20(10):2637–2647, 2012.

[8] C. Harte, M. B. Sandler, S. A Abdallah, and E. Gómez. Symbolic representation of musical chords: A proposed syntax for text annotations. In *ISMIR'05*, pages 66–71, 2005.

[9] M. I. Mandel and D. P. W. Ellis. A web-based game for collecting music metadata. *J. of New Music Research*, 37(2):151–165, 2008.

[10] M. Mauch and G. Cannam, C. Fazekas. Efficient computer-aided pitchtrack and note estimation for scientific applications, 2014. SEMPRE'14, extended abstract.

[11] M. Mauch and S. Dixon. pYIN: A fundamental frequency estimator using probabilistic threshold distributions. In *IEEE ICASSP'14*, 2014. In press.

[12] J. Salamon and E. Gómez. Melody extraction from polyphonic music signals using pitch contour characteristics. *IEEE TASLP*, 20(6):1759–1770, 2012.

[13] J. Salamon, E. Gómez, D. P. W. Ellis, and G. Richard. Melody extraction from polyphonic music signals: Approaches, applications and challenges. *IEEE Signal Processing Magazine*, 31(2):118–134, 2014.

[14] J. Salamon and J. Urbano. Current challenges in the evaluation of predominant melody extraction algorithms. In *ISMIR'12*, pages 289–294, 2012.

[15] J. B. L. Smith, J. A. Burgoyne, I. Fujinaga, D. De Roure, and J .S. Downie. Design and creation of a large-scale database of structural annotations. In *ISMIR'11*, pages 555–560, 2011.

MELODY EXTRACTION FROM POLYPHONIC AUDIO OF WESTERN OPERA: A METHOD BASED ON DETECTION OF THE SINGER'S FORMANT

Zheng Tang
University of Washington, Department of Electrical Engineering
zhtang@uw.edu

Dawn A. A. Black
Queen Mary University of London, Electronic Engineering and Computer Science
dawn.black@qmul.ac.uk

ABSTRACT

Current melody extraction approaches perform poorly on the genre of opera [1, 2]. The singer's formant is defined as a prominent spectral-envelope peak around 3 kHz found in the singing of professional Western opera singers [3]. In this paper we introduce a novel melody extraction algorithm based on this feature for opera signals. At the front end, it automatically detects the singer's formant according to the Long-Term Average Spectrum (LTAS). This detection function is also applied to the short-term spectrum in each frame to determine the melody. The Fan Chirp Transform (FChT) [4] is used to compute pitch salience as its high time-frequency resolution overcomes the difficulties introduced by vibrato. Subharmonic attenuation is adopted to handle octave errors which are common in opera vocals. We improve the FChT algorithm so that it is capable of correcting outliers in pitch detection. The performance of our method is compared to 5 state-of-the-art melody extraction algorithms on a newly created dataset and parts of the ADC2004 dataset. Our algorithm achieves an accuracy of 87.5% in singer's formant detection. In the evaluation of melody extraction, it has the best performance in voicing detection (91.6%), voicing false alarm (5.3%) and overall accuracy (82.3%).

1. INTRODUCTION

Singing voice can be considered to carry the main melody in Western opera. Melody extraction from a polyphonic signal including singing voice requires both of the following: estimation of the correct pitch of singing voice in each time frame and voicing detection to determine when the singing voice is present or not.

The singer's (or singing) formant was first introduced by Johan Sundberg [3] and described as a clustering of the third, fourth, and fifth formants to form a prominent spectral-envelope peak around 3 kHz. It is purportedly generated by widening the pharynx and lowering the larynx. The existence of a singer's formant has been confirmed in the singing voices of classically trained male Western opera singers and some female singers, but it has not yet been found in soprano singers [5] or Chinese opera singers [6]. It has been proposed that singers develop the singer's formant in order to be heard above the orchestra. In Western opera, orchestral instruments typically occupy the same frequency range as the singers. Therefore singers train their vocal equipment in order to raise the amplitude of frequencies at this range.

The LTAS is the average of all short-term spectra in a signal, has been shown to be an excellent tool to observe the singer's formant [7] as can be seen in Figure 1. Characteristics of the singer's formant in the spectral domain include a peak greater than 20 dB below the overall sound pressure level, a peak-location at 2.5-3.2 kHz, and a bandwidth of around 500-800 Hz [5, 7]. However, to date, there has been no method developed to automatically detect the presence of a singer's formant or to quantify its characteristics.

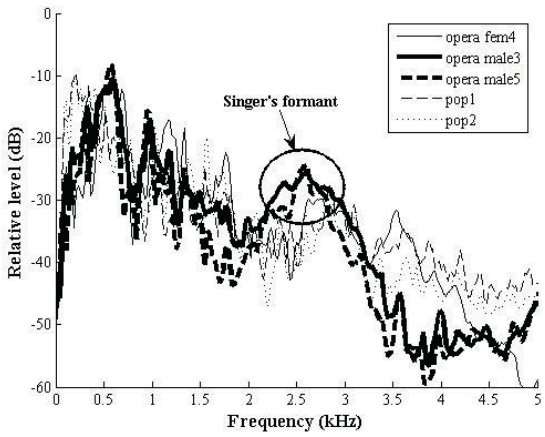


Figure 1. Normalized LTAS for 5 audio excerpts from the ADC2004 test collection [1].

1.1 Related Work

In 2004, the Music Technology Group of the Universitat Pompeu Fabra organized a melody extraction contest presented at the International Society for Music Information Retrieval Conference. The Music Information Retrieval Evaluation eXchange (MIREX) was set up in 2005 and audio melody extraction has been a highly competitive field ever since. Currently, over 60 algorithms have been

submitted and evaluated. So far, none of the approaches consider the presence of the singer's formant.

The majority of algorithms presented at MIREX are salience based [2]. These assume that the fundamental frequency of the melody is equivalent to the most salient pitch value in each frame. The Short-Time Fourier Transform (STFT) is often chosen to compute pitch salience [7, 8]. In 2008, Pablo Cancela proposed the Fan Chirp Transform (FChT) method, combined with Constant Q Transform (CQT) in music processing. The FChT is a time-warped version of the Fourier Transform that provides better time-frequency resolution [4, 9]. Although the STFT provides adequate resolution in the majority of cases, it fails to generate a satisfying outcome when dealing with Western opera signals. This is because opera typically exhibits complex spectral characteristics due to vocal ornamentations such as vibrato [1]. Vibrato is a regular fluctuation of singing pitch produced by singers. This increases the difficulty in tracking the melody. With better resolution, the fast change of pitch salience can be better observed and tracked by using FChT.

It has been proposed that the singer's formant may cause octave errors [2]. The presence of a spectral peak (the singer's formant) at a higher frequency may cause the fundamental frequency to be confused with the frequency at the centre of the singer's formant. To address this, Cancela developed a method called 'subharmonic attenuation' that can minimize the negative effects of ghost pitch values at the multiple and submultiple peaks of a certain fundamental frequency [2, 9].

Voicing detection typically receives much less attention than pitch detection, to the extent that some previous melody extraction algorithms did not contain this procedure [10]. The most common approach is to set an energy threshold, which might be fixed or dynamic [9]. However, this technique is too simplistic since the loudness of musical accompaniment in Western opera may fluctuate considerably. It is therefore impossible to define an appropriate threshold. An alternative technique is to use a trained classifier based on a Hidden Markov Model (HMM) [11] but it is time-consuming to create a large dataset for training and there are always exceptions beyond the scope of the training set. In 2009, Regnier and Peeters proposed a voicing detection algorithm based on extraction of vocal vibrato [12], but has not been applied to melody extraction. In general, the high rate of false positives when detecting voiced frames limits the overall accuracy of melody extraction algorithms and a reduction of this is beneficial [2, 13].

This paper is organized as follows. In Section 2, we describe the design and implementation of our proposed algorithm for melody extraction. Starting with a general workflow of the system, the function and novelty of each component is explained in detail. Section 3 explains the evaluation process and presents a comparison of existing algorithms. The creation of the new dataset is also presented in this section. Finally, we draw conclusions from the results and give suggestions for future work.

2. DESCRIPTION OF THE ALGORITHM

2.1 General Workflow

Figure 2 shows an overview of our system. In order to extract the pitch of singing voice from polyphonic audio, we must first determine whether the audio contains singing voice. The presence of a singer's formant would indicate the presence of a classically trained singer. The LTAS is used to determine whether a singer's formant exists in the audio, and hence determines whether our method can be applied.

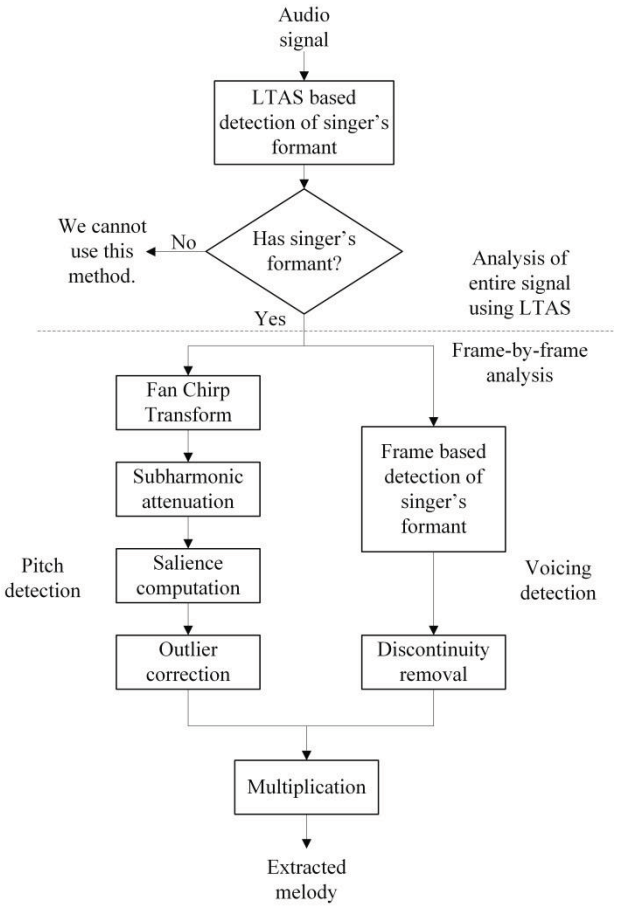


Figure 2. System overview.

Once the presence of a singer is confirmed the spectrum is analysed on a frame-by-frame basis. Two decisions are made for each frame: firstly, does the frame contain singing and hence a salient pitch? Secondly, what is the salient pitch of that frame?

We examine the spectral content of each frame to establish the presence of a singer's formant in that frame. If present, that frame is designated 'voiced' and assumed to contain melody carried by the singer's pitch.

Each frame is also transformed to the frequency domain using the FChT and further processed by subharmonic attenuation to obtain the pitch.

2.2 Singer's Formant Detection and Voicing Detection

Based on the characteristics of the singer's formant (see Section 1) we introduce a novel algorithm to automatically detect the presence of a singer's formant (and hence the presence of a classically trained singer). Using Monson's method to compute the LTAS of the input audio signal [14] the presence of a singer's formant would be confirmed if the LTAS exhibited the following properties:

1. There exists a spectral peak which has an amplitude greater than 20 dB less than the overall sound pressure level.
2. The peak is located between 2.5 and 3.2 kHz.
3. The peak has a bandwidth of around 500-800 Hz.

However, these properties were observed through analysis of singing voice in the absence of musical accompaniment [7]. When analysing singing with accompaniment, these criteria had to be modified in the following ways: the amplitude threshold of the spectral peak was found to be lower than the theoretical value and thus the first criteria becomes:

1. The spectral peak has an amplitude greater than 30 dB less than the overall sound pressure level.

The LTAS exhibited irregular fluctuations that made accurate identification of the singer's formant peak problematic. We therefore smoothed the LTAS (20 point average) and used polynomial fitting of degree 30. This smoothing and polynomial fitting will shift the location of the spectral peak and hence the range of the peak must be expanded. The second criteria is therefore modified to:

2. The peak is located between 2.2 and 3.4 kHz.

Similarly, we observe that the polynomial bandwidth may be slightly different from the LTAS curve. Therefore the bandwidth of the singer's formant is set to be larger than the original value:

3. The peak has a bandwidth larger than 600 Hz.

We must then add another criteria to ensure the significance of the peak. In order to measure the significance, we employed the first-order and second-order derivatives of the LTAS to measure the LTAS curvature and, from empirical evidence, designate significance to be a peak with a curvature greater than 0.01:

4. The curvature exceeds 0.01 at the location of the spectral peak.

In order to illustrate the criteria, we present the following figures. Figure 3 shows the fitting polynomials of smoothed LTAS for 5 samples from the MIREX ADC2004 test collection [1]. The singer's formant can be clearly observed for the male opera samples. Presented in Figure 4 is the second-order derivative of LTAS. This is negative when the curve is convex and hence can be used to determine the formant bandwidth. Our constraint that the bandwidth be at least 600 Hz is illustrated. In Figure 5, we show that the constraint on curvature can ensure the degree of convexity of the curve. It is clear from all plots that the opera signals sung by male singers contain the singer's formant but others do not.

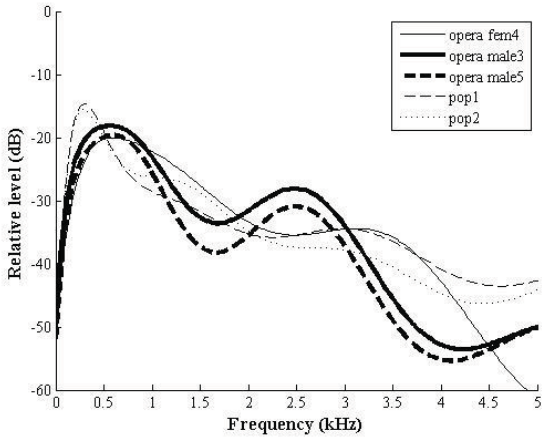


Figure 3. The fitting polynomials of smoothed LTAS for 5 audio excerpts from ADC2004 [1].

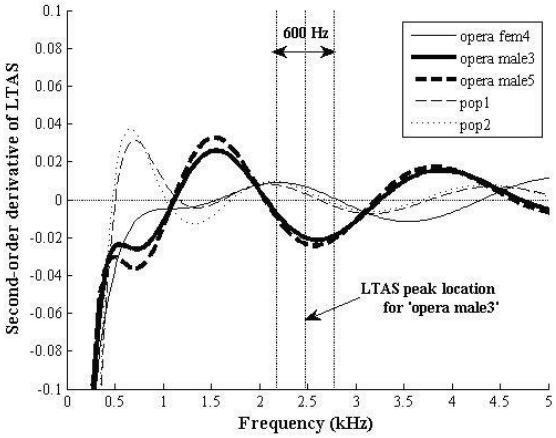


Figure 4. The second-order derivatives of LTAS for 5 audio excerpts from ADC2004 [1].

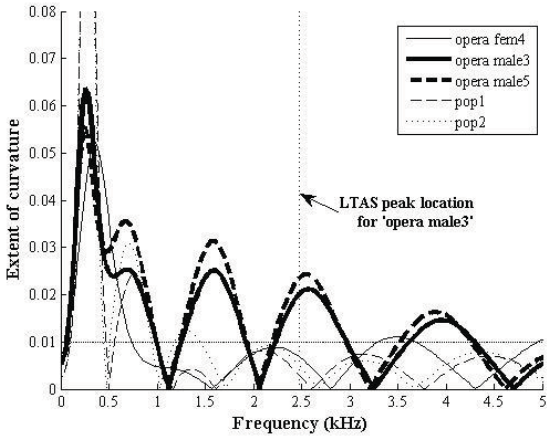


Figure 5. The curvatures of LTAS for 5 audio excerpts from ADC2004 [1].

If the LTAS satisfies all four criteria the audio is presumed to contain a trained singer. Use of the same criteria to analyse the spectrum of a single audio frame can indicate whether the frame is voiced (contains singing) or not. For a single frame, only the second and third criteria are applied, as the other two criteria are more influenced by observed amplitude variations in individual short-term spectra. The output of this stage is a two-value sequence whose length is the number of frames, with '1' indicating a voiced frame and '-1' unvoiced. Subsequently, when considering points of discontinuity causing false detections, single values within a sequence are removed.

2.3 Pitch Detection

If a frame is classified as voiced it can be expected to contain a clearly defined pitch. Vibrato in singing can cause pitch ambiguity. We therefore adopt Cancela's method to perform FChT since it exhibits optimal time-frequency resolution. This chirp-based transform is based on an FFT performed in a warped time domain. It is combined with CQT in order to guarantee high resolution even when the fan chirp rate is not ideal. More details can be found in [4] and [9].

In Western opera the singer's formant will cause peaks at frequencies higher than the fundamental [2]. The algorithm from Cancela provides subharmonic attenuation - an effective solution to this problem. It will suppress multiple and submultiple pitch peaks of the fundamental frequency. Then, we can perform salience computation to detect the pitch in each frame.

In the outlier correction stage, to improve Cancela's method, we compute two additional peaks per frame as candidate substitutes for the wrong pitch. Firstly, the most salient pitch peaks are compared with those from adjacent frames. If a difference of more than 2 semitones occurs on both sides, the estimated pitch in this frame is considered as a wrong detection. In this case we substitute the pitch for this frame with the pitch among the three candidates which is closest to the average of the two adjacent estimations. Due to subharmonic attenuation, the influence of the subharmonics of the top peak is reduced when calculating the other pitch candidates.

Our method is novel to Cancela's in the following ways: (1) The algorithm designed by Cancela extracts multiple salient peaks simultaneously and these are viewed as separate melodies. We introduce the correction block so that the less salient peaks are taken as substitutes of wrong pitch detection in a single estimation of melody. (2) We improve the voicing detection by considering the singer's formant. (3) Cancela's method is not specifically designed for opera items and its potential for dealing with vibrato and other spectrum characteristics has not been explored.

Finally, the estimated pitch sequence is multiplied by the two-value voicing detection sequence. The output of our algorithm follows the standard format of MIREX and records the time-stamp and estimated frequency of each frame.

3. EVALUATION

3.1 The Dataset

The dataset we used for evaluation is a combination of the ADC2004 test collection and our own dataset[1]. Details of the dataset can be found in Table 1.

Among the existing test collections in MIREX, only ADC2004 contains 2 excerpts in the genre of Western opera. In order to evaluate the performance of melody extraction algorithms upon sufficient amount of opera samples for meaningful comparison, we created a new dataset. Nine students from the Central Academy of Drama in Beijing were recorded. All had received more than 5 years of classical voice training except for an amateur Western opera male singer. Their singing voices were recorded in a practice room, about 10×5×5 m with moderate reverberation. The equipment included a Sony PCM-D50 recorder and an AKG C5 microphone. The accompaniments played by orchestra were recorded separately. All the signals were digitized at a sample rate of 44.1 kHz with bit depth 16. We normalized the maximum amplitude of the singing voices to be -1.25 dB. The signal-to-accompaniment ratio is set to 0 dB. The ground truth for melody extraction was generated by a monophonic pitch tracker in SMSTools with manual adjustment [2] using the vocal track only. The frame size was 2048 samples with a step size of 256 samples.

We conducted two evaluations based on this combined dataset. The test set for melody extraction consists of 18 excerpts of 15s-25s duration sung by classically trained Western opera tenors. For the evaluation of singer's formant detection, we will compare them with 14 excerpts sung by trained Western opera sopranos, trained Peking opera singers, pop singers, and a single unprofessional Western opera male singer.

Test set	Singing type	No. of songs	Expectation/ detection of singer's formant
ADC2004	Tenor, Western	2	Yes/ Yes
	Soprano, Western	2	No/ No
	Popular music	4	No/ No
The dataset recorded at the Central Academy of Drama	Tenor, Western	16	Yes/ Yes
	Soprano, Western	2	No/ Yes
	Amateur, Western	2	No/ Yes
	Laosheng, Peking	2	No/ No
	Qingyi, Peking	2	No/ No

Table 1. Test dataset for the evaluations of melody extraction and singer's formant detection.

[1]This database is available for download under a creative commons license at http://c4dm.eecs.qmul.ac.uk/rdr/ all usage should cite this paper.

3.2 Melody Extraction Comparison

Of the many melody extraction algorithms submitted to MIREX, few are freely available. We present five algorithms for comparison. We were limited in our choice by availability, but the methods are representative of the majority of algorithms submitted to MIREX in that they cover common approaches and best performance. Each method is briefly introduced next.

Cancela's algorithm was submitted in 2008 [9]. He used FChT combined with CQT to estimate the pitch in each time frame. Voicing detection is conducted through the calculation of an adaptive threshold, but this procedure is not included in the open-source code provided online. For the purposes of comparison, we added a common voicing detection function utilizing an adaptive energy threshold as described in [9].

Salamon's algorithm was introduced in 2011 [8]. It has been developed into a melody extraction vamp plug-in: MELODIA. This algorithm achieved the best score in MIREX 2011. It applies contour tracking to the salience function calculated by STFT to remove all the contours except for the melody. The voicing detection step is carried out by removing the contours that are not salient.

The algorithm developed by Sutton in 2006 [11] innovatively combines two pitch detectors based on the features of singing voice including pitch instability and high-frequency dominance. A modified HMM processes the estimated melodies and determines the voicing.

The final two algorithms were both proposed by Vincent in 2005 [10]. One makes use of a Bayesian harmonic model to estimate the melody, and the other is achieved via loudness-weighted YIN method. Vincent assumed that the melody was continuous throughout the audio, and voicing detection was not included in his algorithm.

3.3 Results

The evaluation results of singer's formant detection can be found in Table 1. Among the 32 audio files in the dataset, the assumption is that only the 18 excerpts sung by Western opera tenors possess the singer's formant, while the others do not. The results show that 28 of the files (87.5%) meet our expectation. The singer's formant is also detected in the excerpts of the Western opera amateur and sopranos in our dataset. The amateur singer is from the Acting Department at the Central Academy of Drama (Beijing) and declares that he has not received any formal training in opera. However he used to take courses in vocal music due to a requirement of the school. Thus, there is a possibility that the presence of singer's formant only requires a short period of training. Although sources state that there is no singer's formant present in soprano singing [5, 7], the mean pitch of the two excerpts in our dataset is at the low end of the range for sopranos (550.43 Hz). The presence of a singer's formant is pitch related. The higher the pitch, the less likely a singer's formant is present. A precise study of this relationship is a topic for future work.

Table 2 shows the melody extraction results of the 6 algorithms. Voicing detection measures the probability of correct detection of voiced frames, while voicing false alarm is the probability of incorrect detection of unvoiced frames. Raw pitch accuracy and raw chroma accuracy both measure the accuracy of pitch detection, with the latter ignoring octave errors. The overall accuracy is the proportion of frames labeled with correct pitch and voicing. Since Vincent's algorithms did not perform voicing detection, their voicing metrics and overall accuracy are inapplicable.

First author/ completion year	Voicing detection	Voicing false alarm	Raw pitch accuracy	Raw chroma accuracy	Overall accuracy
Vincent (Bayes)/ 2005	N/A	N/A	64.8%	68.6%	N/A
Vincent (YIN)/ 2005	N/A	N/A	69.5%	72.2%	N/A
Sutton/ 2006	89.3%	51.9%	87.0%	87.6%	76.9%
Cancela/ 2008[1]	72.6%	39.3%	83.9%	84.8%	62.4%
Salamon/ 2011	62.3%	21.8%	25.4%	30.1%	31.3%
Our method	91.6%	5.3%	84.3%	85.1%	82.3%

Table 2. Results of the audio melody extraction evaluation.

Our algorithm ranks highest in overall accuracy. We also achieve the highest voicing detection rate as 91.6% and the lowest voicing false alarm rate as 5.3%, which proves that voicing detection based on the singer's formant is extremely effective for male Western opera. The improvement in raw pitch accuracy by outlier correction when compared to Cancela's method is not large. This allows us to hypothesise that the melody in Western opera may be so prominent that the influence of any accompaniment can be disregarded.

Sutton's method also has excellent performance on our dataset. That success might be attributed to his similar focus on the characteristics of singing voice. He also makes use of the vibrato feature to estimate the pitch of melody. Due to the application of a high-frequency correlogram, Sutton's algorithm may indirectly benefit from the presence of a singer's formant. However, the method we propose for voicing detection is much more convenient than the use of an HMM. Moreover, Sutton's algorithm exhibits a much higher voicing false alarm rate.

The poor performance of Salamon's algorithm on our dataset can be explained by the fact that it fails to estimate the pitch in detected unvoiced frames accurately.

We also evaluated the 4 audio files that contradicted our expectation in singer's formant detection (two West-

[1]The voicing detection part of this algorithm is implemented by us and cannot represent the original design of the author.

ern soprano singers and one amateur male Western opera singer). The performance of our algorithm declines significantly with a voicing detection rate of 53.1% and an overall accuracy of 53.7%. This may be due to the fact that the singer's formant, although present, is not as pronounced or stable as the Western opera tenor's.

4. CONCLUSION AND FURTHER WORK

In this paper, we have presented a novel melody extraction algorithm based on the detection of singer's formant. This detection relies on 4 criteria modified from previously proposed characteristics of the singer's formant. The pitch detection step of our algorithm is achieved using FChT and subharmonic attenuation to overcome the known difficulties when detecting the melody in opera. We also improved the algorithm so it is capable of removing outliers in pitch detection.

From the evaluation results, it can be seen that our algorithm can detect the singer's formant accurately. Melody extraction evaluation on our dataset confirms that our algorithm provides a clear improvement in voicing detection. Furthermore, its overall accuracy is comparable to state-of-the-art methods when dealing with Western opera signals.

In the future, we plan to study the performance of this algorithm on signals in other genres and expand its scope of application. Additionally, the possible effects of performing environments and accompaniment music to the usage of singer's formant will also be explored.

5. ACKNOWLEDGMENTS

This paper is based upon a research collaboration with the Department of Peking Opera at the Central Academy of Drama. Thanks to Prof. Ma Li and his students for their recording samples and professional advice on traditional opera. Besides, we would like to express our thanks to Pablo Cancela, Justin Salamon, Emilia Gómez, Christopher Sutton and Emmanuel Vincent for contributing their algorithm codes.

6. REFERENCES

[1] E. Gómez, S. Streich, B. Ong, R. P. Paiva, S. Tappert, J. M. Batke, G. Poliner, D. Ellis, and J. P. Bello: "A quantitative comparison of different approaches for melody extraction from polyphonic audio recordings," Univ. Pompeu Fabra, Barcelona, Spain, 2006, Tech. Rep. MTG-TR-2006-01.

[2] J. Salamon, E. Gómez, D. Ellis, and G. Richard: "Melody extraction from polyphonic music signals: approaches, applications and challenges," *IEEE Signal Processing Magazine*, Vol. 31, No. 2, pp. 118-134, 2013.

[3] J. Sundberg: "Articulatory interpretation of the 'singing formant'," *The Journal of the Acoustical Society of America*, Vol. 55, No. 4, pp. 838-844, 1974.

[4] L. Weruaga, and M. Képesi: "The fan-chirp transform for non-stationary harmonic signals," *Signal Processing*, Vol. 87, No. 6, pp. 1504-1522, 2007.

[5] R. Weiss, Jr, W. S. Brown, and J. Moris: "Singer's formant in sopranos: fact or fiction?" *Journal of Voice*, Vol. 15, No. 4, pp. 457-468, 2001.

[6] J. Sundberg, L. Gu, Q. Huang, and P. Huang: "Acoustical study of classical Peking Opera singing," *Journal of Voice*, Vol. 26, No. 2, pp. 137-143, 2012.

[7] J. Sundberg: "Level and center frequency of the singer's formant," *Journal of voice*, Vol. 15, No. 2, pp. 176-186, 2001.

[8] J. Salamon, and E. Gómez: "Melody extraction from polyphonic music signals using pitch contour characteristics," *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 20, No. 6, pp. 1759-1770, 2012.

[9] P. Cancela, E. López, and M. Rocamora: "Fan chirp transform for music representation," *Proceedings of the 13th Int Conference on Digital Audio Effects DAFx10 Graz Austria*, 2010.

[10] E. Vincent, and M. D. Plumbley: "Predominant-F0 estimation using Bayesian harmonic waveform models," *2005 Music Information Retrieval Evaluation eXchange (MIREX)*, 2005.

[11] C. Sutton: "Transcription of vocal melodies in popular music," Report for the degree of MSc in Digital Music Processing, Queen Mary University of London, 2006.

[12] L. Regnier, and G. Peeters: "Singing voice detection in music tracks using direct voice vibrato detection," *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 1685-1688, 2009.

[13] G. E. Poliner, D. P. Ellis, A. F. Ehmann, E. Gómez, S. Streich, and B. Ong: "Melody transcription from music audio: Approaches and evaluation," *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 15, No. 4, pp. 1247-1256, 2007.

[14] B. B. Monson: "High-frequency energy in singing and speech," Doctoral dissertation, University of Arizona, 2011.

CODEBOOK-BASED SCALABLE MUSIC TAGGING WITH POISSON MATRIX FACTORIZATION

Dawen Liang, John Paisley, Daniel P. W. Ellis
Department of Electrical Engineering
Columbia University
{dliang, dpwe}@ee.columbia.edu, jpaisley@columbia.edu

ABSTRACT

Automatic music tagging is an important but challenging problem within MIR. In this paper, we treat music tagging as a matrix completion problem. We apply the Poisson matrix factorization model jointly on the vector-quantized audio features and a "bag-of-tags" representation. This approach exploits the shared latent structure between semantic tags and acoustic codewords. Leveraging the recently-developed technique of stochastic variational inference, the model can tractably analyze massive music collections. We present experimental results on the CAL500 dataset and the Million Song Dataset for both annotation and retrieval tasks, illustrating the steady improvement in performance as more data is used.

1. INTRODUCTION

Automatic music tagging is the task of analyzing the audio content (waveform) of a music recording and assigning to it human-relevant semantic tags [16] – which may relate to style, genre, instrumentation, or more subtle aspects of the music, such as those contributed by users on social media sites. Such "autotagging" [5] relies on labelled training examples for each tag, and performance typically improves with the number of training examples consumed, although training schemes also take longer to complete. In the era of "Big Data", it is necessary to develop models which can rapidly handle massive amount of data; a starting point for music data is the Million Song Dataset [2], which includes user tags from Last.fm [1].

In this paper, we treat the automatic music tagging as a matrix completion problem, and use the techniques of stochastic variational inference to be able to learn from large amounts of data presented in an online fashion [9]. The "matrix completion" problem treats each track as a row in a matrix, where the elements describe both the acoustic properties (represented, for instance, as a histogram of occurrences of vector-quantized acoustic features) and the relevance of a large vocabulary of tags: We can regard the tag information as incomplete or missing for some of the rows, and seek to "complete" these rows based on information inferred from the complete, present rows.

1.1 Related work

There have been a large number of papers on automatic tagging of music audio in recent years. In addition to the papers mentioned above, work particularly relevant to this paper includes the Codeword Bernoulli Average (CBA) approach of Hoffman *et al.* [7], which uses a similar VQ histogram representation of the audio to build a simple but effective probabilistic model for each tag in a discriminative fashion. Xie *et al.* [17] directly fits a regularized logistic regression model to the normalized acoustic codeword histograms to predict each tag and achieves state-of-the-art results, and Ellis *et al.* [6] further improves tagging accuracy by employing multiple generative models that capture different characteristics of a music piece, which are combined in an optimized "bag-of-systems".

Much of the previous work has been performed on the CAL500 dataset [16] of 502 Western popular music tracks that were carefully labelled by at least three human annotators with their relevance to 149 distinct labels spanning instrumentation, genre, emotions, vocal characteristics, and use cases. This small dataset tends to reward approaches that can maximize the information extracted from the sparse data regardless of the computational cost. A relatively larger dataset in this domain is CAL10k [15] with over 10,000 tracks described by over 500 tags, mined from Pandora's website [1]. However, neither of these datasets can be considered industrial scale, which implies handling millions of tracks and tens of thousands of tags.

Matrix factorization techniques, in particular, nonnegative matrix factorization (NMF), have been widely used to analyze music signals [8, 11] in the context of source separation. Paisley *et al.* [12] derived scalable Bayesian NMF for topic modeling, which we develop here. To our knowledge, this is the first application of matrix factorization to VQ acoustic features for automatic music tagging.

2. DATA REPRESENTATION

For our automatic tagging system, the data comes from two sources: vector-quantized audio features and a "bag-

[1] http://www.pandora.com/

of-tags" representation.

- **Vector-quantized audio features** Instead of directly working with audio features, we vector quantize all the features following the standard procedure: We run the *K*-means algorithm on a subset of randomly selected training data to learn J cluster centroids (codewords). Then for each song, we assign each frame to the cluster with the smallest Euclidean distance to the centroid. We form the VQ feature $y_{\text{VQ}} \in \mathbb{N}^J$ by counting the number of assignments to each cluster across the entire song.

- **Bag-of-tags** Similar to the bag-of-words representation, which is commonly used to represent documents, we represent the tagging information (whether or not the tag applies to a song) with a binary bag-of-tags vector $y_{\text{BoT}} \in \{0,1\}^{|V|}$, where V is the set of all tags.

For each song, we will simply concatenate the VQ feature y_{VQ} and the bag-of-tags vector y_{BoT}, thus the dimension of the data is $D = J + |V|$. When we apply the matrix factorization model to the data, the latent factors we learn will exploit the shared latent structure between semantic tags and acoustic codewords. Therefore, we can utilize the shared latent structure to predict tags when only given the audio features.

3. POISSON MATRIX FACTORIZATION

We adopt the notational convention that bold letters (e.g. $\boldsymbol{y}, \boldsymbol{\theta}, \boldsymbol{\beta}$) denote matrices. $i \in \{1, \cdots, I\}$ is used to index songs. $d \in \{1, \cdots, D\}$ is used to index feature dimensions. $k \in \{1, \cdots, K\}$ is used to index latent factors from the matrix factorization model. Given the data $\boldsymbol{y} \in \mathbb{N}^{I \times D}$ as described in Section 2, the Poisson matrix factorization model is formulated as follows:

$$\begin{aligned} \theta_{ik} &\sim \text{Gamma}(a, ac), \\ \beta_{kd} &\sim \text{Gamma}(b, b), \\ y_{id} &\sim \text{Poisson}(\textstyle\sum_{k=1}^{K} \theta_{ik}\beta_{kd}), \end{aligned} \tag{1}$$

where $\beta_k \in \mathbb{R}_+^D$ denote the kth latent factors and $\theta_i \in \mathbb{R}_+^K$ denote the weights for song i. a and b are model hyperparameters. c is a scalar on the weights that we tune to maximize the likelihood.

There are a couple of reasons to choose a Poisson model over a more traditional Gaussian model [14]. First, the Poisson distribution is a more natural choice to model count data. Secondly, real-world tagging data is extremely noisy and sparse. If a tag is not associated with a song in the data, it could be either because that tag does not apply to the song, or simply because no one has labelled the song with the tag yet. The Poisson matrix factorization model has the desirable property that it does not penalize values of 0 as strongly as the Gaussian distribution [12]. Therefore, even weakly labelled data can be used to learn the Poisson model.

4. VARIATIONAL INFERENCE

To learn the latent factors $\boldsymbol{\beta}$ and the corresponding decomposition weights $\boldsymbol{\theta}$ from the training data $\boldsymbol{y}$, we need to compute the posterior distribution $p(\boldsymbol{\theta}, \boldsymbol{\beta}|\boldsymbol{y})$. However, no closed-form expression exists for this hierarchical model. We therefore employ mean-field variational inference to approximate this posterior [10].

The basic idea behind mean-field variational inference is to choose a factorized family of variational distributions,

$$q(\boldsymbol{\theta}, \boldsymbol{\beta}) = \prod_{k=1}^{K} \Big(\prod_{i=1}^{I} q(\theta_{ik}) \Big) \Big(\prod_{d=1}^{D} q(\beta_{kd}) \Big), \tag{2}$$

to approximate the posterior $p(\boldsymbol{\theta}, \boldsymbol{\beta}|\boldsymbol{y})$, so that the Kullback-Leibler (KL) divergence between the variational distribution and the true posterior is minimized. Following a further approximation discussed in the next section, the factorized distribution allows for a closed-form expression of this variational objective, and thus tractable inference. Here we choose variational distributions from the same family as the prior:

$$\begin{aligned} q(\theta_{ik}) &= \text{Gamma}(\theta_{ik}; \gamma_{ik}, \chi_{ik}), \\ q(\beta_{kd}) &= \text{Gamma}(\beta_{kd}; \nu_{kd}, \lambda_{kd}). \end{aligned} \tag{3}$$

Minimizing the KL divergence is equivalent to maximizing the following variational objective:

$$\mathcal{L} = \mathbb{E}_q[\ln p(\boldsymbol{y}, \boldsymbol{\theta}, \boldsymbol{\beta})] + H(q), \tag{4}$$

where $H(q)$ is the entropy of the variational distribution q. We can optimize the variational objective using coordinate ascent via two approaches: batch inference, which requires processing of the entire dataset for every iteration; or stochastic inference, which only needs a small batch of data for each iteration and can be potentially scale to much larger datasets where batch inference is no longer computationally feasible.

4.1 Batch inference

Although the model in Equation (1) is not conditionally conjugate by itself, as demonstrated in [4] we can introduce latent random variables $z_{idk} \sim \text{Poisson}(\theta_{ik}\beta_{kd})$ with the variational distribution being $q(z_{idk}) = \text{Multi}(z_{id}; \phi_{id})$, where $z_{id} \in \mathbb{N}^K$, $\phi_{idk} \geq 0$ and $\sum_k \phi_{idk} = 1$. This makes the model conditionally conjugate, which means that closed-form coordinate ascent updates are available.

Following the standard results of variational inference for conditionally conjugate model (e.g. [9]), we can obtain the updates for θ_{ik}:

$$\begin{aligned} \gamma_{ik} &= a + \sum_{d=1}^{D} y_{id}\phi_{idk}, \\ \chi_{ik} &= ac + \sum_{d=1}^{D} \mathbb{E}_q[\beta_{kd}]. \end{aligned} \tag{5}$$

The scale c is updated as $c^{-1} = \frac{1}{IK} \sum_{i,k} \mathbb{E}_q[\theta_{ik}]$.

Similarly, we can obtain the updates for β_{kd}:

$$\nu_{kd} = b + \sum_{i=1}^{I} y_{id}\phi_{idk}, \quad \lambda_{kd} = b + \sum_{i=1}^{I} \mathbb{E}_q[\theta_{ik}]. \tag{6}$$

Finally, for the latent variables z_{idk}, the following update is applied:

$$\phi_{idk} \propto \exp\{\mathbb{E}_q[\ln \theta_{ik}\beta_{kd}]\}. \tag{7}$$

The necessary expectations for θ_{ik} are:

$$\mathbb{E}_q[\theta_{ik}] = \gamma_{ik}/\chi_{ik}, \quad \mathbb{E}_q[\ln \theta_{ik}] = \psi(\gamma_{ik}) - \ln \chi_{ik}, \tag{8}$$

where $\psi(\cdot)$ is the digamma function. The expectations for β_{kd} have the same form, but use ν_{kd} and λ_{kd}.

4.2 Stochastic inference

Batch inference will alternate between updating $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$ using the entire data at each iteration until convergence to a local optimum, which could be computationally intensive for large datasets. We can instead adopt stochastic optimization by selecting a subset (mini-batch) of the data at iteration t, indexed by $B_t \subset \{1, \cdots, I\}$, and optimizing over a noisy version of the variational objective $\mathcal{L}$:

$$\mathcal{L}_t = \frac{I}{|B_t|} \sum_{i \in B_t} \mathbb{E}_q[\ln p(y_i, \theta_i|\boldsymbol{\beta})] + \mathbb{E}_q[\ln p(\boldsymbol{\beta})] + H(q). \tag{9}$$

By optimizing $\mathcal{L}_t$, we are optimizing $\mathcal{L}$ in expectation.

The updates for weights θ_{ik} and latent variables z_{idk} are essentially the same as batch inference, except that now we are only inferring weights for the mini-batch of data for $i \in B_t$. The optimal scale c is updated accordingly:

$$c^{-1} = \frac{1}{|B_t|K} \sum_{i \in B_t, k} \mathbb{E}_q[\theta_{ik}]. \tag{10}$$

After alternating between updating weights θ_{ik} and latent variables z_{idk} until convergence, we can take a gradient step, preconditioned by the inverse Fisher information matrix of variational distribution $q(\beta_{kd})$, to optimize β_{kd} (see [9] for more technical details),

$$\nu_{kd}^{(t)} = (1-\rho_t)\nu_{kd}^{(t-1)} + \rho_t\left(b + \frac{I}{|B_t|}\sum_{i \in B_t} y_{id}\phi_{idk}\right),$$
$$\lambda_{kd}^{(t)} = (1-\rho_t)\lambda_{kd}^{(t-1)} + \rho_t\left(b + \frac{I}{|B_t|}\sum_{i \in B_t} \mathbb{E}_q[\theta_{ik}]\right), \tag{11}$$

where $\rho_t > 0$ is a step size at iteration t. To ensure convergence [3], the following conditions must be satisfied:

$$\sum_{t=1}^{\infty} \rho_t = \infty, \quad \sum_{t=1}^{\infty} \rho_t^2 < \infty. \tag{12}$$

One possible choice of ρ_t is $\rho_t = (t_0 + t)^{-\kappa}$ for $t_0 > 0$ and $\kappa \in (0.5, 1]$. It has been shown [9] that this update corresponds to stochastic optimization with a natural gradient step, which better fits the geometry of the parameter space for probability distributions.

4.3 Generalizing to new songs

Once the latent factor $\boldsymbol{\beta} \in \mathbb{R}_+^{K \times D}$ is inferred, we can naturally divide it into two blocks: the VQ part $\boldsymbol{\beta}_{\text{VQ}} \in \mathbb{R}_+^{K \times J}$, and the bag-of-tags part $\boldsymbol{\beta}_{\text{BoT}} \in \mathbb{R}_+^{K \times |V|}$.

Given a new song, we can first obtain the VQ feature y_{VQ} and fit it with $\boldsymbol{\beta}_{\text{VQ}}$ to compute posterior of the weights $p(\theta|y_{\text{VQ}}, \boldsymbol{\beta}_{\text{VQ}})$. We can approximate this posterior with the variational inference algorithm in Section 4.1 with $\boldsymbol{\beta}$ fixed. Then to predict tags, we can compute the expectation of the dot product between the weights θ and $\boldsymbol{\beta}_{\text{BoT}}$ under the variational distribution:

$$\hat{y}_{\text{BoT}} = \mathbb{E}_q[\theta^T \boldsymbol{\beta}_{\text{BoT}}]. \tag{13}$$

Since for different songs the weights θ may be scaled differently, before computing the dot product we normalize $\mathbb{E}_q[\theta]$ so that it lives on the probability simplex. To do automatic tagging, we could annotate the song with top M tags according to $\hat{y}_{\text{BoT}}$. To compensate for a lack of diversity in the annotations, we adopt the same heuristic used in [7] by introducing a "diversity factor" d: For each predicted score, we subtract d times the mean score for that tag. In our system, we set $d = 3$.

5. EVALUATION

We evaluate the model's performance on an annotation task and a retrieval task using CAL500 [16] and Million Song Dataset (MSD) [2]. Unlike the CAL500 dataset where tracks are carefully-annotated, the Last.fm dataset [1] associated with MSD comes from real-world user tagging, and thus contains only weakly labelled data with a tagging vocabulary that is much larger and more diverse. We compare our results on these tasks with two other sets of codebook-based methods: Codeword Bernoulli Average (CBA) [7] and ℓ_2 regularized logistic regression [17]. Like the Poisson matrix factorization model, both methods are easy to train and can scale to relatively large dataset on a single machine. However, since both methods perform optimization in a batch fashion, we will later refer to them as "batch algorithms", along with the Poisson model with batch inference described in Section 4.1.

For the hyperparameters of the Poisson matrix factorization model, we set $a = b = 0.1$, and the number of latent factors $K = 100$. To learn the latent factors $\boldsymbol{\beta}$, we followed the procedure in Section 4.1 for batch inference until the relative increase on the variational objective is less than 0.05%. For stochastic inference, we used a mini-batch size $|B_t| = 1000$ unless otherwise specified and took a full pass of the randomly permuted data. As for the learning rate, we set $t_0 = 1$ and $\kappa = 0.6$. All the source code in Python is available online [2].

5.1 Annotation task

The purpose of annotation task is to automatically tag unlabelled songs. To evaluate the model's ability for annotation, we computed the average per-tag precision, recall,

[2] http://github.com/dawenl/stochastic_PMF

and F-score on a test set. Per-tag precision is defined as the average fraction of songs that the model annotates with tag v that are actually labelled v. Per-tag recall is defined as the average fraction of songs that are actually labelled v that the model also annotates with tag v. F-score is the harmonic mean of precision and recall, and is one overall metric for annotation performance.

5.2 Retrieval task

The purpose of the retrieval task is, when given a query tag v, to provide a list of songs which are related to tag v. To evaluate the models' retrieval performance, for each tag in the vocabulary we ranked each song in the test set by the predicted score from Equation (13). We evaluated the area under the receiver-operator curve (AROC) and mean average precision (MAP) for each ranking. AROC is defined as the area under the curve, which plots the true positive rate against the false positive rate, and MAP is defined as the mean of the average precision (AP) for each tag, which is the average of the precisions at each possible level of recall.

5.3 Results on CAL500

Following the procedure similar to that described in [7, 17], we performed a 5-fold cross-validation to evaluate the annotation and retrieval performance on CAL500. We selected the top 78 tags, which are annotated more than 50 times in the dataset, and learned a codebook of size $J = 2000$. For the annotation task, we labelled each song with the top 10 tags based on the predicted score. Since CAL500 is a relatively small dataset, we only performed batch inference for Poisson matrix factorization model.

The results are reported in Table 1, which shows that the Poisson model has comparable performance on the annotation task, and does slightly worse on the retrieval task. As mentioned in Section 3, the Poisson matrix factorization model is particularly suitable for noisy and sparse data where 0's are not necessarily interpreted as explicit observations. However, this may not be the case for CAL500, as the vocabulary was well-chosen and the data was collected from a survey where the tagging quality is understandably higher than the actual tagging data in the real world, like the one from Last.fm. Therefore, this task cannot fully exploit the advantage brought by the Poisson model. Meanwhile, the amount of data in CAL500 is fairly small – the data $\boldsymbol{y}$ fit to the model is simply a 502-by-2078 matrix. This prevents us from adopting stochastic inference, which will be shown being much more effective than batch inference even on a 10,000-song dataset in Section 5.4.

5.4 Results on MSD

To demonstrate the scalability of the Poisson matrix factorization model, we conducted experiments using MSD and the associated Last.fm dataset. To our knowledge, there has not been any previous work where music tagging results are reported on the MSD.

Model	Prec	Recall	F-score	AROC	MAP
CBA	0.41	0.24	0.29	0.69	0.47
ℓ_2 LogRegr	0.48	0.26	0.34	0.72	0.50
PMF-Batch	0.42	0.23	0.30	0.67	0.45

Table 1. Results for the top 78 popular tags on CAL500, for Codeword Bernoulli Average (CBA), ℓ_2 regularized logistic regression (ℓ_2 LogRegr), and Poisson matrix factorization with batch inference (PMF-Batch). The results for CBA and ℓ_2 LogRegr are directly copied from [17].

Since the Last.fm dataset contains 522,366 unique tags, it is not realistic to build the model with all of them. We first selected the tags with more than 1,000 appearances and removed those which do not carry discriminative information (e.g. "my favorite", "awesome", "seen live", etc.). Then we ran the stemming algorithm implemented in *NLTK* [3] to further reduce the potential duplications and correct for alternate spellings (e.g. "pop-rock" v.s. "pop rock", "love song" v.s. "love songs"), which gave us a vocabulary of 561 tags. Using the default train/test artist split from MSD, we filtered out the songs which have been labelled with tags from the selected vocabulary. This gave us 371,209 songs for training. For test set, we further selected those which have at least 20 tags (otherwise, it is likely that this song is very weakly labelled). This gave us a test set of 2,757 songs. The feature we used is the Echo Nest's timbre feature, which is very similar to MFCC.

We randomly selected 10,000 songs as the data which can fit into the memory nicely for all the batch algorithms, and trained the following models with different codebook sizes $J \in \{256, 512, 1024, 2048\}$: Codeword Bernoulli Average (CBA), ℓ_2 regularized logistic regression (ℓ_2 LogRegr), Poisson matrix factorization with batch inference (PMF-Batch) and stochastic inference by a single pass of the data (PMF-Stoc-$10K$). Here we used batch size $|B_t| = 500$ for PMF-Stoc-$10K$, as otherwise there will only be 10 mini-batches from the subset. However, given enough data, in general larger batch size will lead to relatively superior performance, since the variance of the noisy variational objective in Equation (9) is smaller. To demonstrate the effectiveness of the Poisson model on massive amount of data (exploiting the stochastic algorithm's ability to run without loading the entire dataset into memory), we also trained the model with the full training set with stochastic inference (PMF-Stoc-full). For the annotation task, we labelled each song with the top 20 tags based on the predicted score.

The results are reported in Table 2. In general, the performance of Poisson matrix factorization is comparably better for smaller codebook size J. Specifically, for stochastic inference, even if the amount of training data is relatively small, it is not only significantly faster than batch inference, but can also help improve the performance by quite a large margin. Finally, not surprisingly, PMF-Stoc-full dominates all the metrics, regardless of the size of the codebook, because it is able to learn from more data.

[3] `http://www.nltk.org/`

Codebook size	Model	Precision	Recall	F-score	AROC	MAP
$J = 256$	CBA	0.112 (0.007)	0.121 (0.008)	0.116	0.695 (0.005)	0.112 (0.006)
	ℓ_2 LogRegr	0.091 (0.008)	0.093 (0.006)	0.092	0.692 (0.005)	0.110 (0.006)
	PMF-Batch	0.113 (0.007)	0.105 (0.006)	0.109	0.647 (0.005)	0.094 (0.005)
	PMF-Stoc-$10K$	0.116 (0.007)	0.127 (0.007)	0.121	0.682 (0.005)	0.105 (0.006)
	PMF-Stoc-full	**0.127 (0.008)**	**0.143 (0.008)**	**0.134**	**0.704 (0.005)**	**0.115 (0.006)**
$J = 512$	CBA	0.120 (0.007)	0.127 (0.008)	0.124	0.689 (0.005)	0.117 (0.006)
	ℓ_2 LogRegr	0.096 (0.008)	0.108 (0.007)	0.101	0.693 (0.005)	0.113 (0.006)
	PMF-Batch	0.111 (0.007)	0.108 (0.006)	0.109	0.645 (0.005)	0.098 (0.005)
	PMF-Stoc-$10K$	0.112 (0.007)	0.128 (0.007)	0.120	0.687 (0.005)	0.110 (0.006)
	PMF-Stoc-full	**0.130 (0.008)**	**0.154 (0.008)**	**0.141**	**0.715 (0.005)**	**0.122 (0.006)**
$J = 1024$	CBA	0.118 (0.007)	0.126 (0.007)	0.122	0.692 (0.005)	0.117 (0.006)
	ℓ_2 LogRegr	0.113 (0.008)	0.129 (0.008)	0.120	0.698 (0.005)	0.115 (0.006
	PMF-Batch	0.112 (0.007)	0.109 (0.006)	0.111	0.635 (0.005)	0.098 (0.006)
	PMF-Stoc-$10K$	0.111 (0.007)	0.127 (0.007)	0.118	0.687 (0.005)	0.111 (0.006)
	PMF-Stoc-full	**0.127 (0.008)**	**0.146 (0.008)**	**0.136**	**0.712 (0.005)**	**0.120 (0.006)**
$J = 2048$	CBA	**0.124 (0.007)**	0.129 (0.007)	0.127	0.689 (0.005)	0.117 (0.006)
	ℓ_2 LogRegr	0.115 (0.008)	0.137 (0.008)	0.125	0.698 (0.005)	**0.118 (0.006)**
	PMF-Batch	0.109 (0.007)	0.110 (0.006)	0.110	0.637 (0.005)	0.098 (0.006)
	PMF-Stoc-$10K$	0.107 (0.007)	0.124 (0.007)	0.115	0.682 (0.005)	0.106 (0.006)
	PMF-Stoc-full	0.120 (0.007)	**0.147 (0.008)**	**0.132**	**0.712 (0.005)**	**0.118 (0.006)**

Table 2. Annotation (evaluated using precision, recall, and F-score) and retrieval (evaluated using area under the receiver-operator curve (AROC) and mean average precision (MAP)) performance on the Million Song Dataset with various codebook sizes, from Codeword Bernoulli Average (CBA), ℓ_2 regularized logistic regression (ℓ_2 LogRegr), Poisson matrix factorization with batch inference (PMF-Batch) and stochastic inference by a single pass of the subset (PMF-Stoc-$10K$) and full data (PMF-Stoc-full). One standard error is reported in the parenthesis.

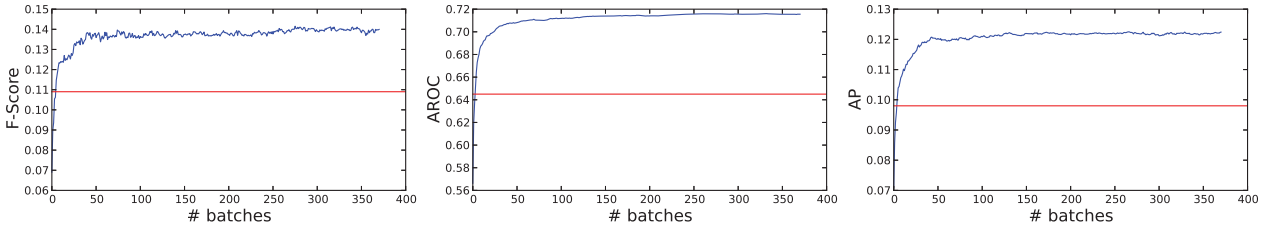


Figure 1. Improvement in performance with the number of mini-batches consumed for the PMF-Stoc-full system with $J = 512$. Red lines indicate the performance of PMF-Batch which is trained on 10k examples; that system's performance is exceeded after fewer than 5 mini-batches.

Figure 1 illustrates how the metrics improve as more data becomes available for the Poisson matrix factorization model, showing how the F-score, AROC, and MAP improve with the number of (1000-element) mini-batches consumed up to the entire 371k training set. We see that initial growth is rapid, thanks to the natural gradient, with much of the benefit obtained after just 50 batches. However, we see continued improvement beyond this; it is reasonable to believe that if more data becomes available, the performance can be further improved.

Table 3 contains information on the qualitative performance of our model. The tagging model works by capturing correlations between semantic tags and acoustic codewords in each latent factor β_k. As discussed, when a new song arrives with missing tag information, only the portion of β_k corresponding to acoustic codewords is used, and the semantic tag portion of β_k is used to make predictions of the missing tags. Similar to related topic models [9], we can therefore look at the highly probable tags for each β_k to understand what portion of the acoustic codeword space is being captured by that factor, and whether it is musically coherent. We show an example of this in Table 3, where we list the top 7 tags from 9 latent factors β_k learned by our model with $J = 512$. We sort the tags according to expected relevance under the variational distribution $\mathbb{E}_q[\beta_{kd}]$. This shows which tags are considered to have high probability for a song that has the given factor expressed. As is evident, each factor corresponds to a particular aspect of a music genre. We note that other factors contained similarly coherent tag information.

6. DISCUSSION AND FUTURE WORK

We present a codebook-based scalable music tagging model with Poisson matrix factorization. The system learns the joint behavior of acoustic features and semantic tags, which

"Pop"	*"Indie"*	*"Jazz"*	*"Classical"*	*"Metal"*	*"Reggae"*	*"Electronic"*	*"Experimental"*	*"Country"*
pop	indie	chillout	piano	metal	reggae	house	instrumental	country
female vocal	rock	lounge	instrumental	death metal	funk	electro	ambient	classic country
dance	alternative	chill	ambient	thrash metal	funky	electronic	experimental	male vocal
electronic	indie rock	downtempo	classic	brutal death metal	dance	dance	electronic	blues
sexy	post punk	smooth jazz	beautiful	grindcore	hip-hop	electric house	psychedelic	folk
love	psychedelic	relax	chillout	heavy metal	party	techno	progressive	love songs
synth pop	new wave	ambient	relax	black metal	sexy	minimal	rock	americana

Table 3. Top 7 tags from 9 latent factors for PMF-Stoc-full with $J = 512$. For each factor, we assign the closest music genre on top. As is evident, each factor corresponds to a particular aspect of a music genre.

can be used to infer the most appropriate tags given the audio alone. The Poisson model is naturally less sensitive to zero values than some alternatives, making it a good match to "noisy" training examples derived from real users' taggings, where the fact that no user has applied a tag does not necessarily imply that the term is irrelevant. By learning this model using stochastic variational inference, we are able to efficiently exploit much larger training sets than are tractable using batch approaches, making it feasible to learn from an entire set of over 370k tagged examples. Although much of the improvement comes in the earlier iterations, we see continued improvement implying this approach can benefit from much larger, effectively unlimited sources of tagged examples, as might be available on a commercial music service with millions of users.

There are a few areas where our model can be easily developed. For example, stochastic variational inference requires we set the learning rate parameters t_0 and κ, which is application-dependent. By using adaptive learning rates for stochastic variational inference [13], model inference can converge faster and to a better local optimal solution. From a modeling perspective, currently the hyperparameters for weights $\boldsymbol{\theta}$ are fixed, indicating that the sparsity level of the weight for each song is assumed to be the same *a priori*. Alternatively we could put *song-dependent* hyper-priors on the hyperparameters of $\boldsymbol{\theta}$ to encode the intuition that some of the songs might have denser weights because more tagging information is available. This would offer more flexibility to the current model.

7. ACKNOWLEDGEMENTS

The authors would like to thank Matthew Hoffman for helpful discussion. This work was supported in part by NSF grant IIS-1117015.

8. REFERENCES

[1] Last.fm dataset, the official song tags and song similarity collection for the Million Song Dataset. `http://labrosa.ee.columbia.edu/millionsong/lastfm`.

[2] T. Bertin-Mahieux, D. Ellis, B. Whitman, and P. Lamere. The Million Song Dataset. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 591–596, 2011.

[3] L. Bottou. Online learning and stochastic approximations. *On-line learning in neural networks*, 17(9), 1998.

[4] A. T. Cemgil. Bayesian inference for nonnegative matrix factorisation models. *Computational Intelligence and Neuroscience*, 2009.

[5] D. Eck, P. Lamere, T. Bertin-Mahieux, and S. Green. Automatic generation of social tags for music recommendation. In *Advances in Neural Information Processing Systems*, pages 385–392, 2007.

[6] K. Ellis, E. Coviello, A. Chan, and G. Lanckriet. A bag of systems representation for music auto-tagging. *Audio, Speech, and Language Processing, IEEE Transactions on*, 21(12):2554–2569, 2013.

[7] M. Hoffman, D. Blei, and P. Cook. Easy as CBA: A simple probabilistic model for tagging music. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 369–374, 2009.

[8] M. Hoffman, D. Blei, and P. Cook. Bayesian nonparametric matrix factorization for recorded music. In *Proceedings of the 27th Annual International Conference on Machine Learning*, pages 439–446, 2010.

[9] M. Hoffman, D. Blei, C. Wang, and J. Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

[10] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

[11] D. Liang, M. Hoffman, and D. Ellis. Beta process sparse nonnegative matrix factorization for music. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 375–380, 2013.

[12] J. Paisley, D. Blei, and M.I. Jordan. Bayesian nonnegative matrix factorization with stochastic variational inference. In E.M. Airoldi, D. Blei, E.A. Erosheva, and S.E. Fienberg, editors, *Handbook of Mixed Membership Models and Their Applications*. Chapman and Hall/CRC, 2015.

[13] R. Ranganath, C. Wang, D. Blei, and E. Xing. An adaptive learning rate for stochastic variational inference. In *Proceedings of The 30th International Conference on Machine Learning*, pages 298–306, 2013.

[14] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems*, pages 1257–1264, 2008.

[15] D. Tingle, Y.E. Kim, and D. Turnbull. Exploring automatic music annotation with acoustically-objective tags. In *Proceedings of the international conference on Multimedia information retrieval*, pages 55–62. ACM, 2010.

[16] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet. Semantic annotation and retrieval of music and sound effects. *IEEE Transactions on Audio, Speech and Language Processing*, 16(2):467–476, 2008.

[17] B. Xie, W. Bian, D. Tao, and P. Chordia. Music tagging with regularized logistic regression. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 711–716, 2011.

Oral Session 2

Transcription

This Page Intentionally Left Blank

TEMPLATE ADAPTATION FOR IMPROVING AUTOMATIC MUSIC TRANSCRIPTION

Emmanouil Benetos†, Roland Badeau‡, Tillman Weyde† and Gaël Richard‡
† Department of Computer Science, City University London, UK
{emmanouil.benetos.1, t.e.weyde}@city.ac.uk
‡ Institut Mines-Télécom, Télécom ParisTech, CNRS LTCI, France
{roland.badeau, gael.richard}@telecom-paristech.fr

ABSTRACT

In this work, we propose a system for automatic music transcription which adapts dictionary templates so that they closely match the spectral shape of the instrument sources present in each recording. Current dictionary-based automatic transcription systems keep the input dictionary fixed, thus the spectral shape of the dictionary components might not match the shape of the test instrument sources. By performing a *conservative* transcription pre-processing step, the spectral shape of detected notes can be extracted and utilized in order to adapt the template dictionary. We propose two variants for adaptive transcription, namely for single-instrument transcription and for multiple-instrument transcription. Experiments are carried out using the MAPS and Bach10 databases. Results in terms of multi-pitch detection and instrument assignment show that there is a clear and consistent improvement when adapting the dictionary in contrast with keeping the dictionary fixed.

1. INTRODUCTION

Automatic music transcription (AMT) is defined as the process of converting an acoustic music signal into some form of music notation [3]. Subtasks of AMT include multi-pitch detection, onset/offset detection, and instrument identification. Recently, the vast majority of transcription approaches use *spectrogram factorization* methods such as non-negative matrix factorization (NMF) and probabilistic latent component analysis (PLCA), which attempt to decompose an input non-negative spectrogram into spectral templates and note activations (e.g. [2, 10, 17]). The spectral templates can either be pre-extracted and stored in a template dictionary [2, 17] or can be estimated using parametric spectral models [10]. An open problem with dictionary-based methods is that the templates might not match the spectral shape of the input instrument sources. Also, unconstrained methods such as NMF and standard PLCA that jointly update the spectral templates and pitch activations can lead to the creation of non-informative bases, and thus, to poor transcription results. It has been shown (e.g. [3]) that the use of templates from the same instrument model or recording conditions can dramatically improve transcription performance.

Related work on automatically estimating or adapting templates for transcription includes [12], where the authors proposed a system for user-assisted (i.e. semi-automatic) music transcription in an NMF setting. The user can label a few notes in the recording; knowledge of the labelled notes can be used in order to create a dictionary that matches the input source. In addition, in [18], the authors propose a dictionary adaptation step within a sparse model that is suitable for single-instrument multi-pitch detection.

In this paper, we propose a method for template adaptation suitable for multiple-instrument polyphonic music transcription (supporting both multi-pitch detection and instrument assignment). The proposed method is based on a multiple-instrument transcription system using PLCA, and supporting tuning changes and frequency modulations. By performing a conservative transcription in a pre-processing step, notes are detected with a high degree of confidence and are used in order to expand the current template dictionary. An additional PLCA-based dictionary adaptation step can further refine the dictionary, so that it matches closely the input source(s). Two system variants are proposed, for single- and multiple-instrument transcription. Experiments using the MAPS [8] and Bach10 [7] databases show a consistent improvement in multi-pitch detection and instrument assignment performance when the proposed template adaptation method is used.

The outline of the paper is as follows. In Section 2, the proposed single-instrument transcription system is presented, with the multiple-instrument version presented in Section 3. The employed datasets, evaluation metrics, and results are detailed in Section 4. Finally, conclusions are drawn and future directions are indicated in Section 5.

2. SINGLE-INSTRUMENT SYSTEM

In the following, we describe a method for single-instrument polyphonic music transcription based on a dictionary of pre-extracted note templates, which is adapted in order to

EB is supported by a City University London Research Fellowship. This work was supported by a Télécom ParisTech sabbatical grant.

match the input instrument source. The proposed system contains a "conservative" transcription pre-processing step in order to detect notes with a high degree of confidence, a dictionary adaptation step, and a final transcription step. The diagram of the proposed system can be seen in Fig. 1.

2.1 Pre-processing

As a pre-processing step, we perform an initial transcription which uses a fixed template dictionary (in which the templates might not be extracted from the same instrument source, model, or recording conditions). The main goal is to only detect notes for which we have a high degree of confidence; in order to achieve this, we perform a "conservative" transcription, as in [16], where the employed transcription system detects notes with high precision and low recall. In other words, the system returns few false alarms but might miss several notes present in the recording.

In order to perform the conservative transcription pre-processing step, we use the spectrogram factorization-based model of [2], which is based on probabilistic latent component analysis (PLCA) [14] and supports the use of a fixed template dictionary. It should be noted that the system in [2] ranked first in the MIREX transcription tasks [1]. The model of [2] takes as input a normalized log-frequency spectrogram $V_{\omega,t} \in \mathbb{R}^{\Omega \times T}$ (ω denotes frequency and t time) and approximates it as a bivariate probability distribution $P(\omega, t)$. $P(\omega, t)$ is in turn decomposed as:

$$P(\omega, t) = P(t) \sum_{p,f,s} P(\omega|s, p, f) P_t(f|p) P_t(s|p) P_t(p) \quad (1)$$

where p, f, s denote pitch, log-frequency shifting, and instrument source (in the single-instrument case, s refers to instrument model), respectively. $P(t)$ is the spectrogram energy (known quantity) and $P(\omega|s, p, f)$ are pre-extracted spectral templates for pitch p, source/model s, which are also pre-shifted across log-frequency according to parameter f. $P_t(f|p)$ is the time-varying log-frequency shifting for pitch p, $P_t(s|p)$ is the source contribution, and $P_t(p)$ is the pitch activation. As a log-frequency representation we use the constant-Q transform [13] with 60 bins/octave, resulting in $f \in [1, \ldots, 5]$, where $f = 3$ is the ideal tuning position for the template (using equal temperament).

Using a fixed template dictionary, the parameters that need to be estimated are $P_t(f|p)$, $P_t(s|p)$, and $P_t(p)$. This can be achieved using the expectation-maximization (EM) algorithm [5], with 15-20 iterations being typically sufficient. The resulting multi-pitch output is given by $P(p, t) = P(t)P_t(p)$.

In order to extract note events in spectrogram factorization-based AMT algorithms, typically thresholding is performed on the pitch activations ($P(p, t)$ in this case). The value of the threshold θ controls the levels of precision/recall. A low threshold has a high recall and low precision; the opposite occurs with a high threshold. By selecting a high value of θ, in essence we perform a conservative transcription. The final output of the pre-processing step is a collection of pitches and time frames $\{p_1, t_1\}, \{p_2, t_2\}, ..., \{p_N, t_N\}$ which can be used in order to adapt the template dictionary.

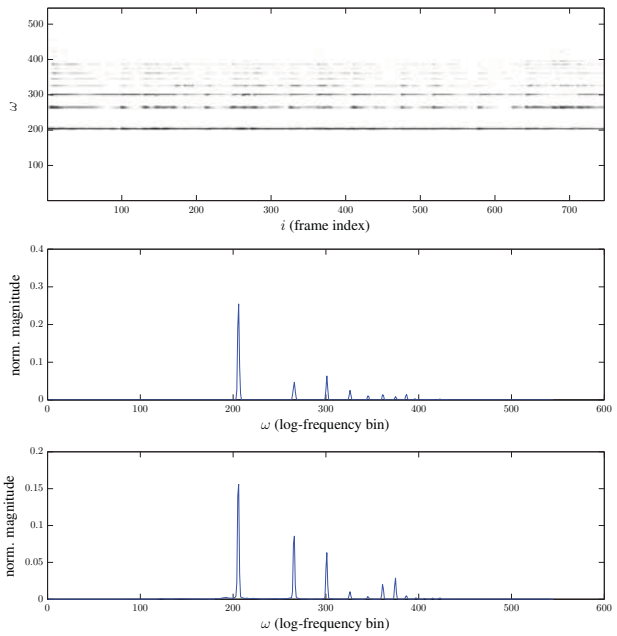


Figure 2. Top: a collection of spectra $\hat{V}^{(42)}$ (note D4) from piano recording 'alb_se2' taken from the MAPS database (piano model: ENSTDkCl). Middle: extracted normalised template $P(\omega|p = 42)$. Bottom: a D4 template from piano model AkPnBcht from the MAPS database.

2.2 Template Adaptation

Given a collection of detected pitches, the first step regarding template adaptation is to collect the spectra that correspond to the aforementioned pitches in the recording. Thus, for each pitch p all time frames t_{ip} that contain that pitch are collected (where $i = 1, ..., N_p$ and N_p is the number of frames containing p).

Subsequently, for each pitch p we create a collection of spectra where that pitch is observed:

$$\hat{V}^{(p)} = V_{\omega, t \in t_{ip}} \otimes \mathbf{h}_p \quad (2)$$

where $\mathbf{h}_p$ is a harmonic comb that serves as an indicator function (setting to zero all frequency bins not belonging to pitch p), and $\otimes$ denotes elementwise multiplication. In other words, $\hat{V}^{(p)} \in \mathbb{R}^{\Omega \times N_p}$ is a collection of the spectra corresponding to detected pitch p in the input recording.

Using information from $\hat{V}^{(p)}$, new spectral templates are created for each p that was detected in the conservative transcription step. In order to create the new templates, the standard PLCA algorithm is used with one component [14], with the input in each case being $\hat{V}^{(p)}$. The output for each p is a spectral template $\mathbf{w}^{(p)}$ which can be used in order to expand the present dictionary.

Given that the conservative transcription step might not have detected all possible pitches present in the recording, information from the extracted templates can be used in order to estimate missing templates. As in the user-assisted case of [12], we can derive templates at missing pitches by simply shifting existing templates across log-frequency. Given a missing pitch template, we consider a neighbor-

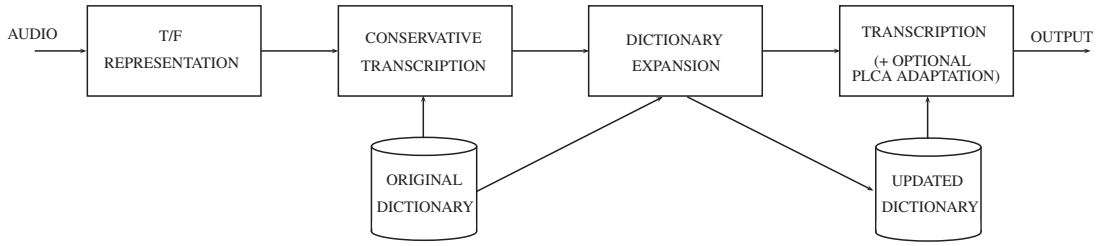


Figure 1. Proposed system diagram.

hood of up to 4 semitones; if a template exists in the neighborhood, it is shifted accordingly in order to estimate the missing template. Finally, the resulting template dictionary is pre-shifted across log-frequency over a semitone range in order to account for tuning deviations and frequency modulations. The output of the template adaptation step is normalized and denoted as $P(\omega|s = s_{new}, p, f)$, where s_{new} refers to the new instrument source that is added to the existing dictionary.

The template adaptation step is illustrated in Fig. 2, where a collection of extracted spectra for note D4 of a piano recording can be seen, along with the computed template, as well as with a template for the same note taken from a different piano model. By comparing the two piano spectra, the importance of adapting templates to the specific instrument source can be seen.

2.3 Transcription

Having created an expanded dictionary with a set of note templates taken from the instrument source present in the recording, the recording is re-transcribed using the new dictionary and the model of (1). In order to further adapt the extracted templates to the input source, an optional step is also applied on updating the new template set during the PLCA iterations. The modified iterative update rule is based on the work of [15] (which incorporated prior information on PLCA update rules) and is applied only for the new set of templates. It is formulated as:

$$\hat{P}(\omega|s_{new}, p, f) = \frac{\sum_t \alpha P_t(p, f, s_{new}|\omega)V_{\omega,t} + (1-\alpha)P(\omega|s_{new}, p, f)}{\sum_{\omega,t} \alpha P_t(p, f, s_{new}|\omega)V_{\omega,t} + (1-\alpha)P(\omega|s_{new}, p, f)} \quad (3)$$

where $P_t(p, f, s|\omega)$ is the posterior of the model (defined in [2]), and α is a parameter which controls the weight of the PLCA adaptation, with $(1-\alpha)$ giving weight to the set of extracted templates from the procedure of Section 2.2. In this work, α is set to 0.05, thus the PLCA template adaptation is only slightly changing the shape of the templates (given that the model is unconstrained, giving a large weight to the PLCA adaptation step would result in non-meaningful templates).

Finally, the output of the transcription step is given by $P(p, t) = P(t)P_t(p)$. For converting the non-binary pitch activation into a binary piano-roll representation, as in [6] we perform thresholding on $P(p, t)$ followed by a process removing note events with a duration less than 80ms.

3. MULTIPLE-INSTRUMENT SYSTEM

In dictionary-based multiple-instrument transcription, the dictionary typically consists of one or more sets of templates per instrument. Thus, in order to update dictionary templates for multiple instruments, modifications need to be made from the system presented in Section 2.

Regarding the pre-processing step, we still use the model of (1), which supports multiple-instrument transcription. In this case, s denotes instrument source. An advantage of the model of (1) is that it can produce an instrument assignment output (i.e. each detected note is assigned to a specific instrument). Thus, having estimated the unknown model parameters, the instrument assignment output for instrument s_{ins} is given by the following time-pitch representation:

$$P(s = s_{ins}, p, t) = P_t(s = s_{ins}|p)P_t(p)P(t) \quad (4)$$

The representation $P(s, p, t)$ can be thresholded in the same way as the pitch activation in order to derive a binary piano-roll representation of the notes produced by a specific instrument. Here, we perform "conservative" thresholding (i.e. use a high θ value) for every instrument in $P(s, p, t)$ in order to create a collection of detected pitches and time frames per instrument:

$$\{s_1, p_1, t_1\}, \{s_2, p_2, t_2\}, ..., \{s_N, p_N, t_N\} \quad (5)$$

where $s \in 1, \ldots, S$, $p \in 1, \ldots, 88$, and $t \in 1, \ldots, T$.

For performing multi-instrument template adaptation, we collect all time frames t_{ips} that contain pitch p and instrument s. We create a collection of spectra $\hat{V}^{(p,s)}$ where a pitch is observed for a specific instrument, in the same way as in (2). Using information from $\hat{V}^{(p,s)}$, new spectral templates are created for specific cases of s and p using the single-component PLCA algorithm. As in Section 2.2, templates at missing pitches are derived by translating existing templates across log-frequency. The output of the template adaptation step is denoted as $P(\omega|s = \{s_{new1}, s_{new2}, ...\}, p, f)$ where $s_{new1}, s_{new2}, ...$ denote the new sets of templates for the existing instruments.

Finally, the input recording is re-transcribed using the model of (1), by utilizing the expanded dictionary. We also apply the same optional PLCA-based dictionary adaptation step shown in Section 2.3. The multiple-instrument transcription system has two sets of outputs: the pitch activation $P(p, t)$ (which is used for multi-pitch detection evaluation) and the instrument contribution $P(s, p, t)$ (which is used for instrument assignment evaluation).

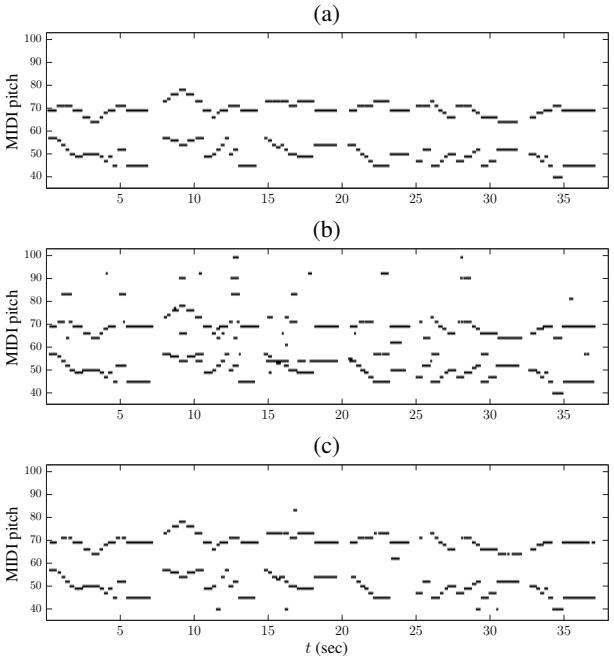


Figure 3. (a) The pitch ground truth for the bassoon-violin duet 'Nun bitten' from the Bach10 database. (b) The transcription piano-roll without template adaptation. (c) The transcription piano-roll with template adaptation.

An example of how template adaptation can improve transcription performance for a multiple-instrument recording (bassoon and violin) is given in Fig. 3, where the transcription output with template adaptation has significantly fewer false alarms compared with transcription without template adaptation (in which many extra detected notes can be seen in higher pitches).

4. EVALUATION

4.1 Datasets

For training the single-instrument system of Section 2, we used isolated note recordings from the 'AkPnBcht' and 'SptkBGCl' piano models of the MAPS database [8]. We used the standard PLCA algorithm with one component [14] in order to extract a single template per note, covering the complete piano note range. For testing the single-instrument system, we used thirty piano segments of 30s duration from the MAPS database from the 'ENSTDkCl' piano model; the test dataset has in the past been used for multi-pitch evaluation (e.g. [2,4,19]). For comparative purposes, we also extracted training templates from the same test source ('ENSTDkCl').

For training the multiple-instrument system of Section 3, we used isolated note samples of bassoon and violin from the RWC database [11], covering the complete note range of the instruments. For testing the multiple-instrument system, we created ten duets of bassoon-violin, mixed from single instrument tracks from the multi-track Bach10 dataset [7]. The duration of the recordings varies from 25-41sec. For comparative purposes, we also extracted dictionary templates for bassoon and violin from the single instrument tracks of the Bach10 database, in order to demonstrate the upper performance limit of the transcription system.

System	Pre_n	Rec_n	F_n
C1	66.41%	48.41%	55.33%
C2	68.07%	48.80%	56.26%
C3	67.84%	49.38%	56.56%
C4 (oracle)	70.43%	50.35%	58.17%

Table 1. Multi-pitch detection results for the single-instrument system using the MAPS database.

4.2 Metrics

For evaluating the performance of the proposed systems for multi-pitch detection, we employ onset-only note-based transcription metrics, which are used in the MIREX note tracking task [1]. A detected note is considered correct if its pitch matches a ground truth pitch and its onset is within a 50ms tolerance of a ground-truth onset. The resulting note-based precision, recall, and F-measure are defined as:

$$Pre_n = \frac{N_{tp}}{N_{sys}} \quad Rec_n = \frac{N_{tp}}{N_{ref}} \quad F_n = \frac{2Rec_n Pre_n}{Rec_n + Pre_n} \quad (6)$$

where N_{tp} is the number of correctly detected pitches, N_{sys} is the number of pitches detected by the system, and N_{ref} is the number of reference pitches.

For the instrument assignment evaluations we use the pitch ground-truth of each instrument separately (compared with the instrument-specific piano-roll output of the system), and compute the F-measure metrics for bassoon (F_b) and violin (F_v).

4.3 Results - Single Instrument Evaluation

For single-instrument transcription evaluation using the 30 MAPS recordings, results are shown in Table 1 using four different system configurations. Configuration C1 corresponds to the system without template adaptation; C2 to the system with template adaptation; C3 to the system with template adaptation using both the creation of the new dictionary plus the PLCA update of the dictionary, as shown in Section 2.2. Finally, C4 refers to comparative experiments without template adaptation, but using templates from the same instrument source ('ENSTDkCl' model in the single-instrument case), which is meant to demonstrate the upper performance limit of the transcription system.

From the single-instrument multi-pitch detection results, it can be seen that an improvement of +0.9% in terms of F_n is reported when using the template adaptation procedure; the improvement rises to +1.2% when also using the PLCA dictionary adaptation updates. The performance difference between the original C1 system (without knowledge of the source templates) and the 'optimal' system (C4) which contains templates from the same test source is 2.8%; thus, the proposed template adaptation steps can help bridge the gap, without requiring any knowledge of

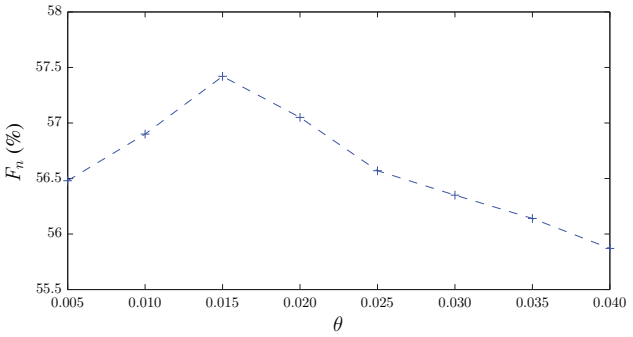


Figure 4. Multi-pitch detection results on the MAPS-ENSTDkCl set using different values of θ.

the test instrument source. Regarding precision and recall, in all cases it can be seen that the transcription system has fewer false alarms than missed note detections. The proposed template adaptation steps help in equally improving precision and recall.

In order to determine the value of the conservative transcription threshold θ, we used a training subset of 10 recordings from the MAPS 'SptkBGCl' models; the value of $\theta = 0.028$ was selected by maximising Pre_n. In Figure 4, transcription performance on the MAPS-ENSTDkCl set is reported by selecting various values for θ. It can be seen that the transcription performance can reach up to F_n=57.4% with $\theta = 0.015$, which enforces the argument that the proposed template adaptation method can successfully adapt dictionary templates so that they match the input instrument source.

Another comparison for the single-instrument system is made, where the dictionary derived from Section 2.2 replaces the dictionary of instrument 'SptkBGCl' (instead of expanding the original dictionary). The resulting F_n is 55.88%, indicating that expanding the dictionary leads to better results compared with replacing the dictionary. It should also be noted that the achieved transcription performance outperforms the system in [19] which reports a frame-based F-measure of 52.4%, whereas the template adaptation system reports a frame-based F of 59.73%. Finally, no rigorous figures for statistical significance of the results can be given since all signal frames cannot be considered as independent samples. However, the reported tests are run on several thousands of frames which leads, if the samples were independent, to a statistically significant difference of the order of 0.6% (with 95% confidence).

4.4 Results - Multiple Instrument Evaluation

For multiple-instrument evaluation, we also use the four different system configurations that were used for single-instrument transcription. For system configuration C3, we perform the PLCA dictionary update using 3 variants: by updating the bassoon only, by updating the violin only, or by updating both dictionaries. Transcription results for the multiple-instrument case are shown in Table 2.

It can be seen that without any template adaptation (C1), $F_n = 67.72\%$; by performing the proposed template adaptation step (C2), the multi-pitch detection F-measure improves by +4.8%.

System	Pre_n	Rec_n	F_n	F_b	F_v
C1	64.79%	71.20%	67.72%	70.19%	42.10%
C2	69.71%	75.72%	72.51%	70.81%	45.98%
C3 (violin)	70.02%	75.41%	72.50%	70.54%	44.51%
C3 (bassoon)	72.49%	77.67%	74.90%	68.77%	45.87%
C3 (both)	71.30%	77.37%	74.11%	67.57%	44.08%
C4 (oracle)	74.90%	82.94%	78.64%	81.25%	62.05%

Table 2. Multi-pitch detection and instrument assignment results for the multiple-instrument system using the Bach10 dataset.

By performing template adaptation with C3 which also includes the PLCA update rule of (3), although no performance gain is obtained over the C2 configuration for the violin updates, there is a +2.4% improvement over C2 when updating the bassoon dictionary only. Finally, when updating both dictionaries, there is a performance drop for F_b and F_v over the C2 configuration (but the system still outperforms the original C1 system). The performance of the PLCA-based dictionary updates can be explained by the fact that the update rule of (3) might combine the observed spectra from both instruments and produce dictionaries that might represent a combination of the two instruments. Finally, the C4 system represents the upper performance limit, which is +11.7% higher than when using a dictionary from a different instrument models or recording conditions. It can be seen that the proposed template adaptation methods help in bridging that performance gap, resulting in a dictionary that closely matches the test instrument sources.

Regarding instrument assignment performance, in all cases the bassoon note identification reports better results compared to violin note identification. It can be seen that with the proposed template adaptation, the bassoon identification remains relatively constant (a small improvement of +0.6% is reported when comparing C1 with C2). On the other hand, violin identification improves by +3.9%; this indicates that the RWC bassoon templates closely matched the Bach10 bassoon models, whereas the violin RWC templates could greatly benefit from template adaptation.

By comparing the MAPS and Bach10 evaluations, an observation can be made that the performance improvement using the proposed template adaptation method depends on the mismatch between the original dictionary and the spectral shape of the instruments present in the recordings. Thus, the 11.7% performance gap for the Bach10 dataset led to a greater improvement for the template adaptation method compared to the 2.8% performance gap reported for the MAPS dataset (which led to a smaller, yet consistent improvement when using the proposed template adaptation method).

5. CONCLUSIONS

In this paper, we proposed a novel method for template adaptation for automatic music transcription, that can be used in dictionary-based systems. We utilized a multiple-

instrument transcription system based on probabilistic latent component analysis, and performed a conservative transcription pre-processing step in order to detect notes with a high confidence. Based on the initial transcription, the spectra of the detected notes are collected, processed, and are used in order to create a new dictionary that closely matches the spectral characteristics of the input instrument source(s). Both single-instrument and multi-instrument variants of the proposed method are presented and evaluated, in terms of multi-pitch detection and instrument assignment. Experimental results using the MAPS and Bach10 datasets show that there is a clear and consistent performance improvement when using the proposed template adaptation method, especially when there is a large discrepancy between the original dictionary and the spectral characteristics of the test instrument sources.

In the future, we will evaluate the proposed system using multiple-instrument recordings with more than two instruments. Parametric models (such as source-filter models) will also be investigated for updating the note templates, along with adaptive methods for deriving the conservative transcription threshold. We also plan to evaluate the proposed system in the next MIREX evaluations [1]. Finally, the proposed template adaptation steps will also be evaluated in the context of score-informed source separation using spectrogram factorization models [9].

6. REFERENCES

[1] Music Information Retrieval Evaluation eXchange (MIREX). `http://music-ir.org/mirexwiki/`.

[2] E. Benetos, S. Cherla, and T. Weyde. An efficient shift-invariant model for polyphonic music transcription. In *6th Int. Workshop on Machine Learning and Music*, Prague, Czech Republic, September 2013.

[3] E. Benetos, S. Dixon, D. Giannoulis, H. Kirchhoff, and A. Klapuri. Automatic music transcription: challenges and future directions. *J. Intelligent Information Systems*, 41(3):407–434, December 2013.

[4] J. J. Carabias-Orti, T. Virtanen, P. Vera-Candeas, N. Ruiz-Reyes, and F. J. Cañadas-Quesada. Musical instrument sound multi-excitation model for non-negative spectrogram factorization. *IEEE J. Selected Topics in Signal Processing*, 5(6):1144–1158, 2011.

[5] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Statistical Society*, 39(1):1–38, 1977.

[6] A. Dessein, A. Cont, and G. Lemaitre. Real-time polyphonic music transcription with non-negative matrix factorization and beta-divergence. In *11th Int. Society for Music Information Retrieval Conf.*, pages 489–494, 2010.

[7] Z. Duan, B. Pardo, and C. Zhang. Multiple fundamental frequency estimation by modeling spectral peaks and non-peak regions. *IEEE Trans. Audio, Speech, and Language Processing*, 18(8):2121–2133, 2010.

[8] V. Emiya, R. Badeau, and B. David. Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle. *IEEE Trans. Audio, Speech, and Language Processing*, 18(6):1643–1654, 2010.

[9] S. Ewert, B.Pardo, M. Müller, and M. D. Plumbley. Score-informed source separation for musical audio recordings. *IEEE Signal Processing Magazine*, 31(3):116–124, May 2014.

[10] B. Fuentes, R. Badeau, and G. Richard. Harmonic adaptive latent component analysis of audio and application to music transcription. *IEEE Trans. Audio, Speech, and Language Processing*, 21(9):1854–1866, 2013.

[11] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka. RWC music database: music genre database and musical instrument sound database. In *International Conference on Music Information Retrieval*, October 2003.

[12] H. Kirchhoff, S. Dixon, and Anssi Klapuri. Missing template estimation for user-assisted music transcription. In *IEEE Int. Conf. Audio, Speech and Signal Processing*, pages 26–30, 2013.

[13] C. Schörkhuber and A. Klapuri. Constant-Q transform toolbox for music processing. In *7th Sound and Music Computing Conf.*, Barcelona, Spain, July 2010.

[14] M. Shashanka, B. Raj, and P. Smaragdis. Probabilistic latent variable models as nonnegative factorizations. *Computational Intelligence and Neuroscience*, 2008. Article ID 947438.

[15] P. Smaragdis and G. Mysore. Separation by "humming": user-guided sound extraction from monophonic mixtures. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 69–72, New Paltz, USA, October 2009.

[16] D. Tidhar, M. Mauch, and S. Dixon. High precision frequency estimation for harpsichord tuning classification. In *IEEE Int. Conf. Audio, Speech and Signal Processing*, pages 61–64, Dallas, USA, March 2010.

[17] F. Weninger, C. Kirst, B. Schuller, and H.-J. Bungartz. A discriminative approach to polyphonic piano note transcription using supervised non-negative matrix factorization. In *IEEE Int. Conf. Audio, Speech and Signal Processing*, pages 6–10, May 2013.

[18] T. B. Yakar, P. Sprechmann, R. Litman, A. M. Bronstein, and G. Sapiro. Bilevel sparse models for polyphonic music transcription. In *14th Int. Society for Music Information Retrieval Conf.*, pages 65–70, 2013.

[19] K. Yoshii and M. Goto. Infinite composite autoregressive models for music signal analysis. In *13th Int. Society for Music Information Retrieval Conf.*, pages 79–84, October 2012.

NOTE-LEVEL MUSIC TRANSCRIPTION BY MAXIMUM LIKELIHOOD SAMPLING

Zhiyao Duan
University of Rochester
Dept. Electrical and Computer Engineering
zhiyao.duan@rochester.edu

David Temperley
University of Rochester
Eastman School of Music
dtemperley@esm.rochester.edu

ABSTRACT

Note-level music transcription, which aims to transcribe note events (often represented by pitch, onset and offset times) from music audio, is an important intermediate step towards complete music transcription. In this paper, we present a note-level music transcription system, which is built on a state-of-the-art frame-level multi-pitch estimation (MPE) system. Preliminary note-level transcription achieved by connecting pitch estimates into notes often lead to many spurious notes due to MPE errors. In this paper, we propose to address this problem by randomly sampling notes in the preliminary note-level transcription. Each sample is a subset of all notes and is viewed as a note-level transcription candidate. We evaluate the likelihood of each candidate using the MPE model, and select the one with the highest likelihood as the final transcription. The likelihood treats notes in a transcription as a whole and favors transcriptions with less spurious notes. Experiments conducted on 110 pieces of J.S. Bach chorales with polyphony from 2 to 4 show that the proposed sampling scheme significantly improves the transcription performance from the preliminary approach. The proposed system also significantly outperforms two other state-of-the-art systems in both frame-level and note-level transcriptions.

1. INTRODUCTION

Automatic Music Transcription (AMT) is one of the fundamental problems in music information retrieval. Generally speaking, AMT is the task of converting a piece of music audio into a musical score. A complete AMT system needs to transcribe both the pitch and rhythmic content [5]. On transcribing the pitch content, AMT can be performed at three levels from low to high: frame-level, note-level, and stream-level [7]. Frame-level transcription (also called *multi-pitch estimation*) aims to estimate concurrent pitches and instantaneous polyphony in each time frame. Note-level transcription (also called *note tracking*) transcribes notes, which are characterized not only by pitch, but also by onset and offset. Stream-level transcription (also called *multi-pitch streaming*) organizes pitches (or notes) into streams according to their instruments. From the frame-level to the stream-level, more parameters and structures need to be estimated, and the system is closer to a complete transcription system.

While there are many systems dealing with frame-level music transcription, only a few transcribe music at the note level [5]. Among these systems, most are built based on frame-level pitch estimates. The simplest way to convert frame-level pitch estimates to notes is to connect consecutive pitches into notes [4, 9, 15]. During this process, non-significant errors in frame-level pitch estimation can cause significant note tracking errors. False alarms in pitch estimates will cause many notes that are too short, while misses can break a long note into multiple short ones. To alleviate these errors, researchers often fill the small gaps to merge two consecutive notes with the same pitch [2, 7], and apply minimum length pruning to remove too-short notes [4, 6, 7]. This idea has also been implemented with more advanced techniques such as hidden Markov models [12]. Besides the abovementioned methods that are entirely based on frame-level pitch estimates, some methods utilize other information in note tracking, such as onset information [10, 14] and musicological information [13, 14].

In this paper, we propose a new note-level music transcription system. It is built based on an existing multi-pitch estimation method [8]. In [8], a multi-pitch likelihood function was defined and concurrent pitches were estimated in a maximum likelihood fashion. This likelihood function tells how well the set of pitches as a whole fit to the audio frame. In this paper, we modify [8] to also define a single-pitch likelihood function. It tells the likelihood (salience) that a pitch is present in the audio frame. Then preliminary note tracking is performed by connecting consecutive pitches into notes and removing too-short notes. The likelihood of each note is calculated as the product of the likelihood of all its pitches. The next step is the key step in the proposed system. We randomly sample subsets of notes according to their likelihood and lengths. Each subset is treated as a possible note-level transcription. The likelihood of such a transcription is then defined as the product of its multi-pitch likelihood in each frame. Finally, the transcription with the maximum likelihood is returned as the output of the system. We carried out experiments on the Bach10 dataset [8] containing Bach chorales

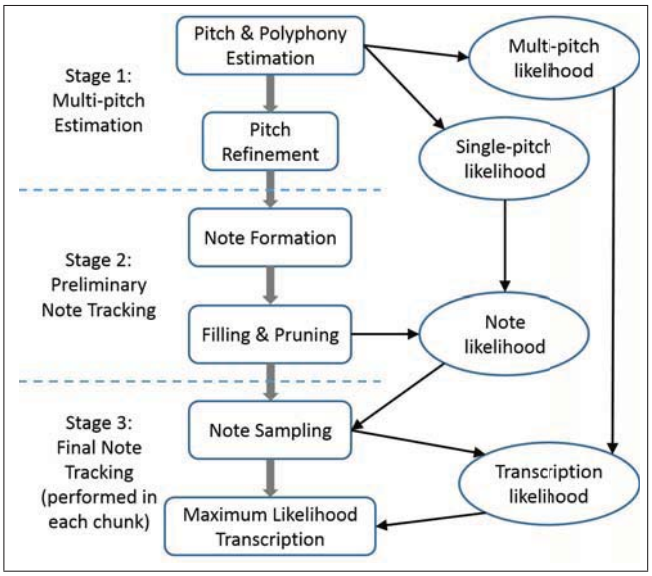


Figure 1. System overview of the proposed note-level transcription system.

with different polyphony. Experiments show that the proposed system significantly improves the transcription performance from the preliminary transcription, and significantly outperforms two state-of-the-art systems at both the note level and frame level on the dataset.

2. PROPOSED SYSTEM

Figure 1 illustrates the overview of the system. It consists of three main stages: multi-pitch estimation, preliminary note tracking, and final note tracking. The first stage is based on [8] with some modifications. The second stage adopts the common filling/prunning strategies used in the literature to convert pitches into notes. The third stage is the main contribution of the paper. Figure 2 shows transcription results obtained at different stages of the system on a piece of J.S. Bach 4-part chorale.

2.1 Multi-pitch Estimation

In [8], Duan et al. proposed a maximum likelihood method to estimate pitches from the power spectrum of each time frame. In the maximum likelihood formulation, pitches (and the polyphony) are the parameters to be estimated while the power spectrum is the observation. The likelihood function $L_{mp}(\{p_1, \cdots, p_N\})$ describes how well a set of N pitches $\{p_1, \cdots, p_N\}$ as a whole fit with the observed spectrum, and hence is called a *multi-pitch likelihood* function. The power spectrum is represented as peaks and the non-peak region, and the likelihood function is defined for both parts. The peak likelihood favors pitch sets whose harmonics can explain peaks, while the non-peak region likelihood penalizes pitch sets whose harmonic positions are in the non-peak region. Parameters of the likelihood function were trained from thousands of musical chords mixed with note samples whose ground-truth pitches were pre-calculated. The maximum likelihood estimation process uses an iterative greedy search strategy. It starts from an empty pitch set, and in each iteration the pitch candidate that results in the highest multi-pitch likelihood increase is selected. The process is terminated by thresholding on the likelihood increase, which also serves for polyphony estimation. After estimating pitches in each frame, a pitch refinement step that utilizes contextual information is performed to remove inconsistent errors.

In this paper, we use the same method to perform MPE in each frame. Differently, we change the instantaneous polyphony estimation parameter settings to achieve a high recall rate of the pitch estimates. This is because the note sampling module in Stage 3 will only remove false alarm notes but cannot add back missing notes (detailed explanation in Section 2.3). In addition, we also calculate a *single-pitch likelihood* $L_{sp}(p)$ for each estimated pitch p. We define it as the multi-pitch likelihood plugged in with the single pitch, i.e., $L_{sp}(p) = L_{mp}(\{p\})$. This likelihood describes how well the single pitch can explain the mixture spectrum, which apparently will not be very good. But from another perspective, this likelihood can be viewed as a salience of the pitch. One important property of multi-pitch likelihood is that it is not additive, i.e., the multi-pitch likelihood of a set of pitches is usually much smaller than the sum of their single-pitch likelihoods:

$$L_{mp}(\{p_1, \cdots, p_N\}) < \sum_{i=1}^{N} L_{mp}(\{p_i\}) = \sum_{i=1}^{N} L_{sp}(p_i) \quad (1)$$

The reason is that the multi-pitch likelihood definition in [8] considers the interaction between pitches. For example, in the peak likelihood definition, a peak will be explained by only one pitch in the pitch set, the one whose corresponding harmonic gives the best fit to the frequency and amplitude of the peak, even if the peak could be explained by multiple pitches. In other words, the single-pitch likelihood considers each pitch independently while the multi-pitch likelihood considers the set as a whole.

The reason of calculating the single-pitch likelihood is because we need to calculate a likelihood (salience) for each note in the second stage, which is further because we need to sample notes using their likelihood in the third stage. Since pitches in the same frame belong to different notes, we need to figure out the likelihood (salience) of each pitch instead of the likelihood of the whole pitch set.

Figure 2(a) shows the MPE result on the example piece. Compared to the ground-truth in (d), it is quite noisy and contains many false alarm pitches, although the main notes can be inferred visually.

2.2 Preliminary Note Tracking

In this stage, we implement a preliminary method to connect pitches into notes, with the ideas of filling and pruning that were commonly used in the literature [2, 4, 6, 7]. We first connect pitches whose frequency difference is less than 0.3 semitones and time difference is less than 100 ms. Each connected component is then viewed as a note. Then notes shorter than 100 ms are removed. The 0.3 semitones threshold corresponds to the range within which the pitch

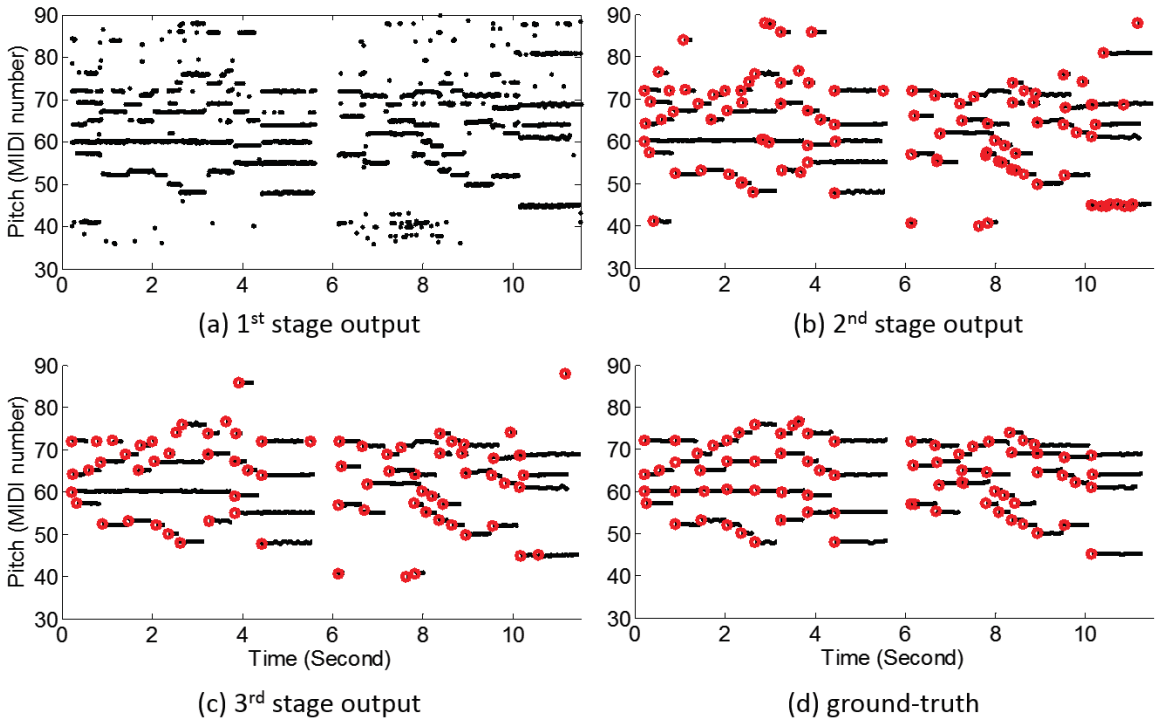


Figure 2. Transcription results on the first 11 seconds of *Ach Lieben Christen*, a piece of 4-part chorales by J.S. Bach. In (a), each pitch is plotted as a point. In (b)-(d), each note is plotted as a line whose onset is marked by a red circle.

often fluctuates within a note, while the 100 ms threshold is a reasonable length of a fast note, as it is the length of a 32nd note in music with a tempo of 75 beats per minute.

Each note is characterized by its pitch, onset, offset, and *note likelihood*. The onset and offset times are the time of the first and last pitch in the note, respectively. The pitch and likelihood are calculated by averaging the pitches and single-pitch likelihood values of all the pitches within the note. Again, this likelihood describes the salience of the note in the audio.

Figure 2(b) shows the preliminary note tracking result. Compared to (a), many noisy isolated pitches have been removed. However, compared to (d), there are still a number of spurious notes, caused by consistent MPE errors (e.g., the long spurious note starting at 10 seconds around MIDI number 80, and a shorter note starting at 4.3 seconds around MIDI number 60). A closer look tells us that both notes and many other spurious notes are higher octave errors of some already estimated notes. This makes sense as octave errors take about half of all errors in MPE [8].

Due to the spurious notes, the instantaneous polyphony constraint is often violated. The example piece has four monophonic parts and at any time there should be no more than four pitches. However, it is often to see more than four notes going simultaneous in Figure 2(b) (e.g., 0-1 seconds, 4-6 seconds, and 10-11 seconds). On the other hand, these spurious notes are hard to remove if we consider them independently: They are long enough from being pruned by the minimum length; They also have high enough likelihood, as the note likelihood is the average likelihood of its pitches. Therefore, we need to consider the interaction between different notes to remove these spurious notes. This leads to the next stage of the system.

2.3 Final Note Tracking

The idea of this stage is quite simple. Thanks to the MPE algorithm in Stage 1, the transcription obtained in Stage 2 inherits the high recall and low precision property. Therefore, a subset of the notes that do not contain many spurious notes but contain almost all correct notes must be a better transcription. The only question now is how can we know which subset is a good transcription. This question can be addressed by an exploration-evaluation strategy: we first explore a number of subsets, and then we evaluate these subsets according to some criterion. But there are two problems of this strategy: 1) how can we efficiently explore the subsets? The number of all subsets is two to the power of the number of notes, hence it is inefficient to enumerate all the subsets. 2) What criterion should we use to evaluate the subsets? If our criterion considers notes independently, then it would not work well, as the spurious notes are hard to distinguish from correct notes in terms of individual note properties such as length and likelihood.

2.3.1 Note Sampling

Our idea to address the exploration problem is to perform note sampling. We randomly sample notes without replacement according to their weights. The weight equals to the product of the note length and the inverse of the negative logarithmic note likelihood. Essentially, longer notes with higher likelihood are more likely to be sampled into

the subset. In this way, we can explore different note subsets, and can guarantee that notes contained in each subset are mostly correct. During the sampling, we also consider the instantaneous polyphony constraint. A note will not be sampled if adding it to the subset would violate the instantaneous polyphony constraint. The sampling process stops if there is no valid note to sample any more.

We perform the sampling process M times to generate M subsets of the notes output in Stage 2. Each subset is viewed as a transcription candidate. We then evaluate the *transcription likelihood* for each candidate and select the one with the highest likelihood. The transcription likelihood is defined as the product of the multi-pitch likelihood of all time frames in the transcription. Since multi-pitch likelihood considers interactions between simultaneous pitches, the transcription likelihood also considers interactions between simultaneous notes. This can help remove spurious notes which are higher octave errors of some correctly transcribed notes. This is because all the peaks that a higher octave error pitch can explain can also be explained by the correct pitch, hence having the octave error pitch in addition to the correct pitch would not increase the multi-pitch likelihood much.

2.3.2 Chunking

The number of subsets (i.e., the sampling space) increases with the number of notes exponentially. If we perform sampling on a entire music piece that contains hundreds of notes, it is likely to require many times of sampling to reach a good subset (i.e., transcription candidate). In order to reduce the sampling space, we segment the preliminary note tracking transcription into one-second long non-overlapping chunks and perform sampling and evaluation in each chunk. Finally, selected transcriptions of different chunks are merged together to get the final transcription of the entire piece. Notes that span across multiple chunks can be sampled in all the chunks, and they will appear in the final transcription if they appear in the selected transcription of some chunk. Depending on the tempo and polyphony of the piece, the number of notes within a chunk can be different. For the 4-part Bach chorales tested in this paper, there are about 12 notes per chunk, and we found sampling 100 subsets gives good accuracy and efficiency.

Figure 2(c) shows the final transcription of the system. We can see that many spurious notes are removed from (b) while most correct notes remain, resulting a much better transcription. The final transcription is very close to the ground-truth transcription.

3. EXPERIMENTS

3.1 Data Set

We use the Bach10 dataset [8] to evaluate the proposed system. This dataset consists of real musical instrumental performances of ten pieces of J.S. Bach four-part chorales. Each piece is about thirty seconds long and was performed by a quartet of instruments: violin, clarinet, tenor saxophone and bassoon. Both the frame-level and note-level ground-truth transcriptions are provided with the dataset. In order to evaluate the system on music pieces with different polyphony, we use the dataset-provided matlab script to create music pieces with different polyphony, which are different combinations of the four parts of each piece. Six duets, four trios and one quartet for each piece was created, totaling 110 pieces of music with polyphony from 2 to 4.

3.2 Evaluation Measure

We evaluate the proposed transcription system with commonly used note-level transcription measures [1]. A note is said to be correctly transcribed, if it satisfies both the pitch condition and the onset condition: its pitch is within a quarter tone from the pitch of the ground-truth note, and its onset is within 50 ms from the ground-truth onset. Offset is not considered in determining correct notes. Then precision, recall, and F-measure are defined as

$$P = \frac{TP}{TP+FP}, R = \frac{TP}{TP+FN}, F = \frac{2PR}{(P+R)}, \quad (2)$$

where TP (true positives) is the number of correctly transcribed notes, FP (false positives) is the number of reported notes not in the ground-truth, and FN (false negatives) is the number of ground-truth notes not reported .

Although note offest is not used in determining correct notes, we do measure the Average Overlap Ratio (AOR) between correctly transcribed notes and their corresponding ground-truth notes. It is defined as

$$AOR = \frac{\min(offsets) - \max(onsets)}{\max(offsets) - \min(onsets)} \quad (3)$$

AOR ranges between 0 and 1, where 1 means that the transcribed note overlaps exactly with the ground-truth note.

To see the improvement of different stages of the proposed system, we also evaluate the system using frame-level measures. Again, we use precision, recall, and F-measures defined in Eq. (2), but here the counts are on the pitches instead of notes. A pitch is considered correctly transcribed if its frequency is within a quarter tone from a ground-truth pitch in the same frame.

3.3 Comparison Methods

3.3.1 Benetos et al.'s System

We compare our system with a state-of-the-art note-level transcription system proposed by Benetos et al. [3]. This system first uses shift-invariant Probabilistic Latent Component Analysis (PLCA) to decompose the magnitude spectrogram of the music audio with a pre-learned dictionary containing spectral templates of all semitone notes of 13 kinds of instruments (including the four kinds used in the Bach10 dataset). The activation weights of the dictionary elements provide the soft version of the frame-level transcription. It is then binarized to obtain the hard version of the frame-level transcription. Note-level transcription is obtained by connecting consecutive pitches, filling short gaps between pitches, and pruning short notes.

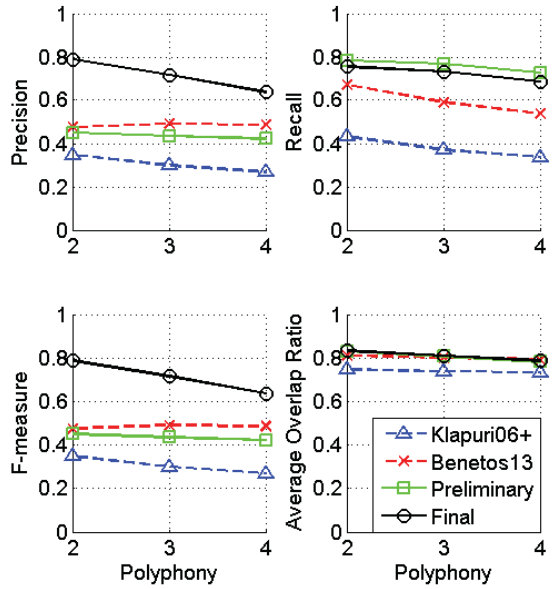


Figure 3. Note-level transcription performances.

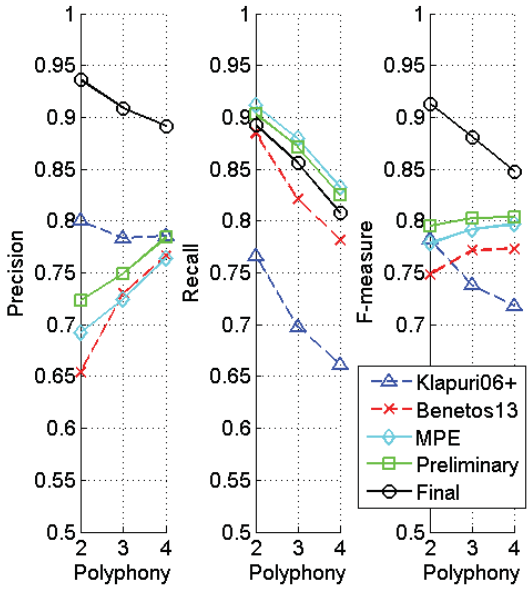


Figure 4. Frame-level transcription performances.

The author's own implementation is available online to generate the soft version frame-level transcription. We then implemented the postprocessing steps according to [3]. Since the binarization threshold is very important in obtaining good transcriptions, we performed a grid search between 1 and 20 with a step size of 1 on the trio pieces. We found 12 gave the best note-level F-measure and used it in all experiments. The time threshold for filling and pruning were set to 100 ms, same as the other comparison methods. We denote this comparison system by "Benetos13".

3.3.2 Klapuri's System

Klapuri's system [11] is a state-of-the-art general-purposed frame-level transcription system. It employs an iterative spectral subtraction approach. At each iteration, a pitch is estimated according to a salience function and its harmonics are subtracted from the mixture spectrum. We use Klapuri's original implementation and suggested parameters. Since Klapuri's system does not output note-level transcriptions, we employ the preliminary note tracking stage in our system to convert Klapuri's frame-level transcriptions into note-level transcriptions. We denote this comparison system by "Klapuri06+".

3.4 Results

Figure 3 compares the note-level transcription performance of the preliminary and final results of the proposed system with Benetos13 and Klapuri06+. It can be seen that the precision of the final transcription of the proposed system is improved significantly from the preliminary transcription for all polyphony. This is accredited to the note sampling stage of the proposed system. As shown in Figure 2, note sampling removes many spurious notes and leads to higher precision. On the other hand, the recall of the final transcription is just slightly decreased (about 3%), which means most correct notes survive during the sampling. Therefore, the F-measure of the final transcription is significantly improved from the preliminary transcription for all polyphony, leading to a very promising performance on this dataset. The average F-measure on the 60 duets is about 79%, which is about 35% higher than the preliminary result in absolute value. The average F-measure on the 10 quartets is about 64%, which is also about 22% higher than the preliminary transcription.

Compared to the two state-of-the-art methods, the final transcription of the proposed system also achieves much higher F-measure. In fact, the preliminary transcription is a little inferior to Benetos13. However, the note sampling stage makes the final transcription surpass Benetos13.

In terms of average overlap ratio (AOR) of the correctly transcribed notes with the ground-truth notes, both preliminary and the final transcription of the proposed system and Benetos13 achieve a similar performance, which is about 80% for all polyphony. This is about 5% higher than Klapuri06+. It is noted that 80% AOR indicates a very good estimation of the note lengths/offsets.

Figure 4 presents the frame-level transcription performance. In this comparison, we also include the MPE result which is the output of Stage 1. There are several interesting observations. First of all, similar to the results in Figure 3, the final transcription of the proposed system improves from the preliminary transcription significantly in both precision and F-measure, and degrades slightly in recall. This is accredited to the note sampling stage. Second, preliminary transcription of the proposed system has actually improved from the MPE result in F-measure. This validates the filling and pruning operations in the second stage, although the increase is only about 3%. Third, the final transcription of the proposed system achieves significantly higher precision and F-measure than the two com-

parison methods, leading to about 91%, 88%, and 85% F-measure for polyphony 2, 3, and 4, respectively. This performance is very promising and may be accurate enough for many other applications.

4. CONCLUSIONS AND DISCUSSIONS

In this paper, we built a note-level music transcription system based on an existing frame-level transcription approach. The system first performs multi-pitch estimation in each time frame. It then employs a preliminary note tracking to connect pitch estimates into notes. The key step of the system is to perform note sampling to generate a number of subsets of the notes, where each subset is viewed as a transcription candidate. The sampling was based on the note length and note likelihood, which was calculated using the single-pitch likelihood of pitches in the note. Then the transcription candidates are evaluated using the multi-pitch likelihood of simultaneous pitches in all the frames. Finally the candidate with the highest likelihood is returned as the system output. The system is simple and effective. Transcription performance was significantly improved due to the note sampling and likelihood evaluation step. The system also significantly outperforms two other state-of-the-art systems on both note-level and frame-level measures on music pieces with polyphony from 2 to 4.

The technique proposed in this paper is very simple, but the performance improvement is unexpectedly significant. We think the main reason is twofold. First, the note sampling step lets us explore the transcription space, especially the good regions of the transcription space. The single-pitch likelihood of each estimated pitch plays an important role in sampling the notes. In fact, we think that probably any kind of single-pitch salience function that have been proposed in the literature can be used to perform note sampling. The second reason is that we use the multi-pitch likelihood, which considers interactions between simultaneous pitches, to evaluate these sampled transcriptions. This is important because notes contained in a sampled transcription must have high salience, however, when considered as a whole, they may not fit with the audio as well as another sampled transcription. One limitation of the proposed sampling technique is that it can only remove false alarm notes in the preliminary transcription but not adding back missing notes. Therefore, it is important to make the preliminary transcription have a high recall rate before sampling.

5. ACKNOWLEDGEMENT

We thank Emmanouil Benetos and Anssi Klapuri for providing the source code or executable program of their transcription systems for comparison.

6. REFERENCES

[1] M. Bay, A.F. Ehmann, and J.S. Downie, "Evaluation of multiple-F0 estimation and tracking systems," in *Proc. ISMIR*, 2009, pp. 315-320.

[2] J.P. Bello, L. Daudet, M.B., Sandler, "Automatic piano transcription using frequency and time-domain information," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 6, pp. 2242-2251, 2006.

[3] E. Benetos, S. Cherla, and T. Weyde, "An efficient shift-invariant model for polyphonic music transcription," in *Proc. 6th Int. Workshop on Machine Learning and Music*, 2013.

[4] E. Benetos and S. Dixon, "A shift-invariant latent variable model for automatic music transcription," *Computer Music J.*, vol. 36, no. 4, pp. 81-94, 2012.

[5] E. Benetos, S. Dixon, D. Giannoulis, H. Kirchhoff, and A. Klapuri, "Automatic music transcription: challenges and future directions," *J. Intelligent Information Systems*, vol. 41, no. 3, pp. 407-434, 2013.

[6] A. Dessein, A. Cont, G. Lemaitre, "Real-time polyphonic music transcription with nonnegative matrix factorization and beta-divergence," in *Proc. ISMIR*, 2010, pp. 489-494.

[7] Z. Duan, J. Han, and B. Pardo, "Multi-pitch streaming of harmonic sound mixtures," *IEEE Trans. Audio Speech Language Processing*, vol. 22, no. 1, pp. 1-13, 2014.

[8] Z. Duan, B. Pardo, and C. Zhang, "Multiple fundamental frequency estimation by modeling spectral peaks and non-peak regions," *IEEE Trans. Audio Speech Language Processing*, vol. 18, no. 8, pp. 2121-2133, 2010.

[9] G. Grindlay and D. Ellis, "Transcribing multi-instrument polyphonic music with hierarchical eigeninstruments," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 6, pp. 1159-1169, 2011.

[10] P. Grosche, B. Schuller, M. Mller, and G. Rigoll, "Automatic transcription of recorded music," *Acta Acustica United with Acustica*, vol. 98, no. 2, pp. 199-215, 2012.

[11] A. Klapuri, "Multiple fundamental frequency estimation by summing harmonic amplitudes," in *Proc. ISMIR*, 2006, pp. 216-221.

[12] G. Poliner, and D. Ellis, "A discriminative model for polyphonic piano transcription," in *EURASIP J. Advances in Signal Processing*, vol. 8, pp. 154-162, 2007.

[13] S.A. Raczyǹski, N. Ono, and S. Sagayama. "Note detection with dynamic bayesian networks as a post-analysis step for NMF-based multiple pitch estimation techniques," in *Proc. WASPAA*, 2009, pp. 49-52.

[14] M. Ryynänen and A. Klapuri, "Polyphonic music transcription using note event modeling," in *Proc. WASPAA*, 2005, pp. 319-322.

[15] E. Vincent, N. Bertin, and R. Badeau, "Adaptive harmonic spectral decomposition for multiple pitch estimation," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 18, no. 3, pp. 528-537, 2010.

DRUM TRANSCRIPTION VIA CLASSIFICATION OF BAR-LEVEL RHYTHMIC PATTERNS

Lucas Thompson, Matthias Mauch and Simon Dixon
Centre for Digital Music, Queen Mary University of London
contact@lucas.im, {m.mauch, s.e.dixon}@qmul.ac.uk

ABSTRACT

We propose a novel method for automatic drum transcription from audio that achieves the recognition of individual drums by classifying bar-level drum patterns. Automatic drum transcription has to date been tackled by recognising individual drums or drum combinations. In high-level tasks such as audio similarity, statistics of longer rhythmic patterns have been used, reflecting that musical rhythm emerges over time. We combine these two approaches by classifying bar-level drum patterns on sub-beat quantised timbre features using support vector machines. We train the classifier using synthesised audio and carry out a series of experiments to evaluate our approach. Using six different drum kits, we show that the classifier generalises to previously unseen drum kits when trained on the other five (80% accuracy). Measures of precision and recall show that even for incorrectly classified patterns many individual drum events are correctly transcribed. Tests on 14 acoustic performances from the ENST-Drums dataset indicate that the system generalises to real-world recordings. Limited by the set of learned patterns, performance is slightly below that of a comparable method. However, we show that for rock music, the proposed method performs as well as the other method and is substantially more robust to added polyphonic accompaniment.

1. INTRODUCTION

The transcription of drums from audio has direct applications in music production, metadata preparation for musical video games, transcription to musical score notation and for musicological studies. In music retrieval, robust knowledge of the drum score would allow more reliable style recognition and more subtle music search by example. Yet like related tasks such as polyphonic piano transcription [1], a versatile, highly reliable drum transcription algorithm remains elusive.

Audio drum transcription methods have been classified into two different strategies [10, 18]: *segment and classify* and *separate and detect*. Systems in the first category detect a regular or irregular event grid in the signal, segment the signal according to the grid, extract features such as MFCCs [19] or multiple low-level features [23] and then classify the segments using Gaussian Mixture Models [19], k nearest neighbour classification [21], or Support Vector Machines [23]. Systems in the second category first detect multiple streams corresponding to drum types, usually via a signal or spectral decomposition approach, e.g. [2, 7], or simpler sub-band filtering [15], and then identify onsets in the individual streams. Other methods combine aspects of both categories, via adaptation [24] or joint detection of onsets and drums [18]. To ensure temporal consistency (smoothness) many approaches make use of high-level statistical models that encode some musical knowledge, e.g. hidden Markov models [18]. The methods greatly differ in terms of the breadth of instruments they are capable of detecting; most detect only bass drum, snare drum and hi-hat [7, 14, 18, 23] or similar variants, probably because these instruments (unlike crash and ride cymbals) can be represented in few frames due to their very fast decay.

Despite the evident diversity of strategies, all existing methods aim directly at detecting individual or simultaneous drum events. As we will see later, our approach is qualitatively different, using higher-level patterns as its classi-

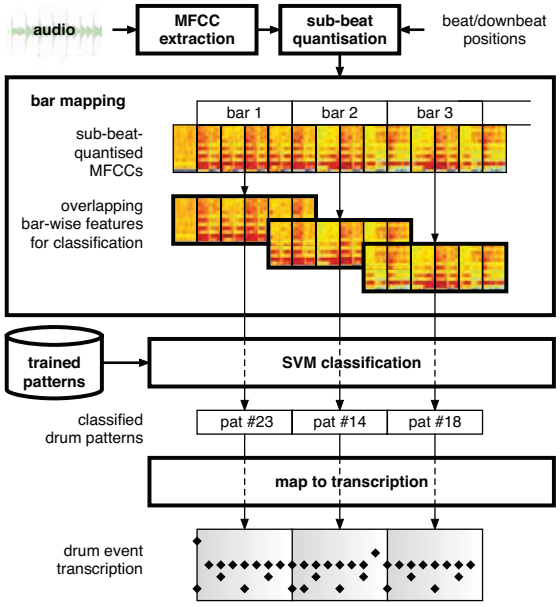


Figure 1. Overview of the system at prediction time.

fication target. One advantage of this is that the long decay of cymbals is naturally modelled at the feature level.

Since drum transcription from polyphonic audio is only partially solved, music information retrieval tasks rely on "soft" mid-level audio features to represent rhythm. Fluctuation patterns [17] summarise the frequency content of sub-band amplitude envelopes across 3-second windows and were used to evaluate song similarity and to classify pop music genres; they have also been used to describe rhythmic complexity [13]. Bar-wise rhythmic amplitude envelope patterns have been shown to characterise ballroom dance music genres [5] and bar-wise pseudo-drum patterns have been shown to correlate with popular music genres [6]. Rhythmic patterns have also formed the basis of beat tracking systems [11] and been used for downbeat detection [8]. These methods share a more musically holistic approach to rhythm, i.e. they summarise rhythmic components in a longer temporal context. Drum tutorials, too, usually focus on complete rhythms, often a bar in length, because "with the command of just a few basic rhythms you can make your way in a rock band" [22]. In fact, we have recently shown that drum patterns are distributed such that a small number of drum patterns can describe a large proportion of actual drum events [12].

Motivated by this result and by the effectiveness of more holistic approaches to rhythm description, we propose a novel drum transcription method based on drum pattern classification. Our main contribution is to show that the classification of bar-length drum patterns is a good proxy for predicting individual drum events in synthetic and real-world drum recordings.

2. METHOD

The proposed method is illustrated in Figure 1. It can broadly be divided into two parts: a feature extraction step, in which MFCC frame-wise features are calculated and formatted into a bar-wise, sub-beat-quantised representation, and a classification step, in which bar-wise drum patterns are predicted from the feature representation and then translated into the desired drum transcription representation. For the sake of this study, we assume that correct beat and bar annotations are given.

2.1 Feature extraction

Following Paulus and Klapuri [19], we choose Mel-frequency cepstral coefficients (MFCCs) as basis features for our experiments. MFCCs are extracted from audio sampled at 44.1 kHz with a frame size of 1024 samples (23ms) and a hop size of 256 samples (6ms), using an adaptation of the implementation provided in the VamPy plugin examples. [1] We extract 14 MFCCs (the mentioned implementation uses a bank of 40 Mel-filters) per frame, but discard the 0^{th} coefficient to eliminate the influence of overall signal level.

In order to obtain a tempo-independent representation, we assume that we know the positions of musical beats and quantise the feature frames into a metrical grid. This is needed for subsequent bar-wise segmentation. Whereas beat-quantised chroma representations usually summarise chroma frames within a whole inter-beat interval [16], drum information requires finer temporal resolution. Hence, following [12] we choose 12 sub-beats per beat, which is sufficient to represent the timing of the most common drum patterns. The MFCC frames belonging to each sub-beat are summarised into a single value by taking the mean over the sub-beat duration to give 12 quantised frames per beat.

Since we assume we know which beat is the downbeat, it is now trivial to extract bar representations from sub-beat-quantised MFCC features. For example, in a $\frac{4}{4}$ time signature, one bar corresponds to $4 \times 12 = 48$ sub-beat-quantised MFCC frames. However, slight deviations in timing and natural decay times of cymbals and drum membranes mean that information on a bar pattern will exist even outside the bar boundaries. For this reason we also add an extra beat either side of the bar lines (further discussion in Section 3), leading to the overlapping bar representations illustrated in Figure 1, each $6 \times 12 = 72$ frames long. The features we are going to use to classify $\frac{4}{4}$ bars into drum patterns will therefore comprise 936 elements (72 frames $\times$ 13 MFCCs).

2.2 Classification and transcription mapping

As our classifier, we use the one-vs-one multi class Support Vector Machine implementation provided in the *sklearn.svm.SVC* [2] package of the Python machine learning library, *scikit-learn* [20], with the default settings using a radial basis kernel, $K(x, x') = e^{-\gamma||x-x'||^2}$, where $\gamma = \frac{1}{N}$ and $N = 936$ is the feature dimension. Once the classifier has predicted a drum pattern for a particular bar, we perform a simple mapping step to obtain a drum transcription: using the information about the actual start and end time of the bar in the recording, each of the drum events that constitute the pattern are assigned to a time stamp within this time interval, according to their position in the pattern.

3. EXPERIMENTS AND RESULTS

We conducted three experiments to test the effectiveness of the proposed method, one with synthesised test data, and two with real recordings of human performances. In all experiments, the drum pattern data for training was encoded as MIDI and then synthesised using the FluidSynth software. Our drum pattern dictionary contains the top 50 most common drum patterns, including the empty pattern, in a collection of 70,000 MIDI files (containing only **bd** - kick, **sd** - snare, **hh** - closed hi-hat, **oh** - open hi-hat, **ri** - ride and **cr** - crash cymbals) [12]. [3] Figure 2 details how each drum class is distributed. Data examples and further

[1] http://www.vamp-plugins.org/vampy.html

[2] http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

[3] http://www.eecs.qmul.ac.uk/~matthiasm/ndrum/patternstats/full_1-2-3-4-5-6/patternvisual_reduced

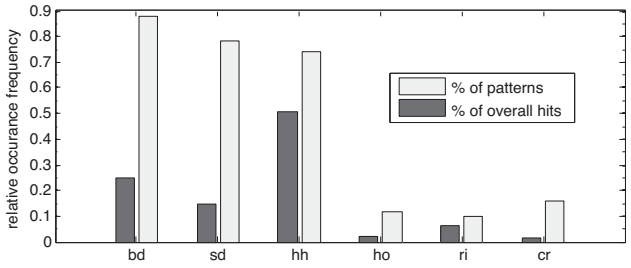


Figure 2. Relative occurrence of the drum classes in terms of overall number of drum events and the number of patterns containing each class. There are 50 patterns with an average of 11 events per pattern.

information can be found on the web page that accompanies this paper. [4]

We evaluate both the pattern classification performance and the quality of transcription of drum events. Pattern accuracy is defined as

$$A = \frac{\text{number of correctly classified bars}}{\text{total number of bars in test set}}. \qquad (1)$$

The transcription of drum events is evaluated using precision, recall and the F-measure (their harmonic mean)

$$P = \frac{N_c}{N_d}, \qquad R = \frac{N_c}{N}, \qquad F = \frac{2PR}{P+R}, \qquad (2)$$

where N_d is the number of detected drum hits, N_c is the number of correctly detected drum hits and N the number of drum hits in the ground truth. The individual drum hits are solely based on the presence or absence of a drum hit at a particular discrete position in the pattern grid used in the dictionary. In Sections 3.2 and 3.3 the ground truth drum hits, given as onset times, are quantised to a position in the grid. Tanghe's method [23] (Sections 3.2 and 3.3) is evaluated against the original ground truth with an acceptance window of 30ms, as in the original paper.

3.1 Multiple Synthesised Drum Kits

The aim of this experiment is to see how well the proposed classifier performs on synthetic data generated using multiple drum kits.

3.1.1 Training and Test Data

In order to create varied training and test data, we first generate 100 unique *songs*, each of which is simply a randomly permuted list of the 50 drum patterns from our dictionary. These *songs* are encoded as MIDI files, and we introduce randomised deviations in note velocity and onset times (velocity range 67–127, timing range $\pm$20 ms) to humanise the performances. All MIDI files are then rendered to audio files (WAV) using 6 drum kits from a set of SoundFonts we collected from the internet. In order to avoid complete silence, which is unrealistic in real-world scenarios, we add white noise over the entirety of each song at a SNR of 55 dB. We then calculate the bar-wise beat-quantised MFCC features as described in section 2.1. This yields a dataset of $6 \times 100 = 600$ files.

We use a random 70:30 train/test split of the 100 songs, where each of the 70 training songs appears in five variations synthesised from different drum kit SoundFonts. The remaining 30 songs, synthesised by the sixth drum kit, are used for testing. In order to assess performance on different drum kits, we cycle the use of the test drum kit in a leave-one-out fashion.

[4] http://www.eecs.qmul.ac.uk/~matthiasm/drummify/

drum-kit	overall drum events R	P	F	classification accuracy
00	98.9 (98.5)	98.9 (98.7)	98.9 (98.6)	91.1 (88.8)
01	97.1 (97.1)	97.6 (97.7)	97.4 (97.4)	89.2 (87.9)
02	98.3 (97.9)	98.3 (98.0)	98.3 (98.0)	87.7 (86.5)
03	84.8 (80.3)	82.3 (85.8)	83.6 (83.0)	50.1 (47.5)
04	92.7 (92.2)	91.2 (90.8)	92.0 (91.5)	72.0 (66.4)
05	97.2 (97.1)	98.5 (98.5)	97.9 (97.8)	91.6 (88.6)
mean	94.8 (93.9)	94.5 (94.9)	94.7 (94.4)	80.3 (77.6)

Table 1. Mean classification accuracy (%) and overall drum event R, P and F metrics for left out drum-kit from leave-one-out cross validation on 6 different drum-kits (see Section 3.1). Results for non-overlapping bars are in brackets.

drum-type	overall drum events R	P	F
bd	96.2 (96.0)	95.4 (95.0)	95.8 (95.5)
sd	96.5 (95.4)	99.3 (99.3)	97.8 (97.0)
hh	96.5 (95.3)	93.7 (94.8)	95.0 (95.0)
ho	59.9 (57.1)	77.3 (77.3)	61.1 (56.8)
ri	86.3 (86.4)	98.3 (99.5)	88.2 (89.0)
cr	84.4 (75.8)	97.0 (96.5)	89.3 (82.5)

Table 2. R, P and F for each drum type, taken over the whole test set and all kits from leave-one-out cross validation on 6 different drum-kits (see Section 3.1). Results for non-overlapping bars are in brackets.

3.1.2 Results

As Table 1 shows, our method achieves a high average accuracy of 80.3%, despite strong variation between drum kits. Irrespective of whether overlapping bar-wise features were used, the accuracy on drum kits 00, 01, 02 and 05 exceeds 85%. Performance is substantially worse on drum kits 03 and 04 (accuracies of 50.1% and 72.0%, respectively). Listening to a subset of the synthesised songs for drum kits 03 revealed that the recording used for the closed hi-hat sounds contains hi-hats that are slightly open, which is likely to cause confusion between the two hi-hat sounds.

To demonstrate the benefit of considering extra beats either side of the bar boundaries, Table 1 includes the results for non-overlapping bars. In this case we can see that the context given by the neighbouring beats increases classification accuracy (mean increase $\approx$3 percentage points). The greatest increase in accuracy ($\approx$6 percentage points) is observed in drum-kit 04.

To gain an insight into the types of patterns being misclassified, we consider those patterns for each drum-kit that are misclassified more than a quarter of the time. Figure 3 contains a few example cases. The single undetected

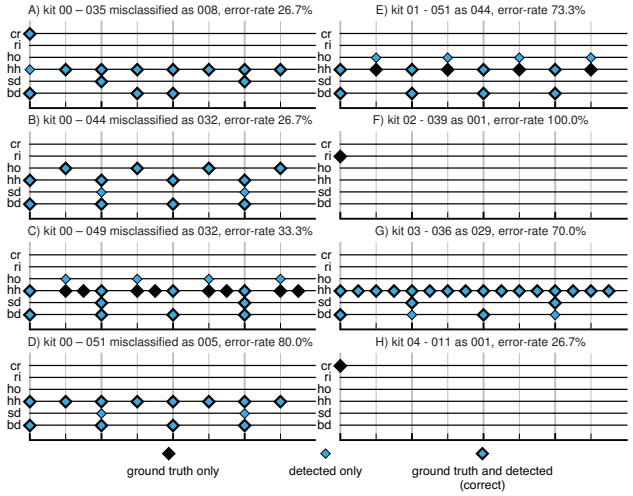


Figure 3. Examples of misclassified patterns (see Section 3.1.2).

ride or crash cymbals on the first beat in the ground-truth (cases F and H) are likely to be caused by the system confusing them for remainders of the previous bar. For cases A, B, D and G, the differences are subtle. In case A, the patterns differ by one hi-hat on the first beat. Cases B, D and G show that on occasions the classifier chooses a pattern where the majority of the drum events are correct, apart from a few inserted bass or snare drum events.

If we compare the individual drum events of the predicted pattern against the ground-truth and use precision and recall measures (see Table 2) we see that the system achieves high F-measures for the majority of drum classes (mean 0.88–0.97 for bd, sd, hh, ri, cr over all kits), but not for the open hi-hat class (mean F-measure 0.61).

Using audio features with overlapping bars leads to a substantial increase of over 8 percentage points in the recall of crash cymbal hits (84.4%) with respect to using no overlap (75.8%). The majority of the crash hits in our pattern dictionary occur on the first beat of the bar, and many of the patterns which were misclassified without the benefit of the overlapping neighbouring beats are such patterns, highlighting that the added context helps distinguish the pattern from those with a decaying hi-hat or other cymbal at the end of the previous bar. Note that since crash cymbals usually occur no more than once per bar, the classification accuracy in Table 1 shows larger improvement than the overall drum event precision and recall values.

3.2 Real drums Without Accompaniment

Having evaluated the performance of our system on synthesised data, we now test its robustness to real acoustic drum data.

3.2.1 Training and test data

We use the set of 100 songs described in the previous experiment (Section 3.1.1) synthesised on all 6 drum kits ($6 \times 100 = 600$ files). Since we have shown that overlapping bar-wise features provide higher accuracy (Section 3.1.2), we use only this feature configuration to train a re-usable model, which is used in the remainder of the experiments.

As test data we use the ENST-Drums database [9], which contains a wide range of drum recordings and ground-truth annotations of drum event onset times. We selected 13 *phrase* performances (15-25 s) which contain a number of similar patterns to ones in our dictionary, with expressional variations and fills, and one song from the *minus-one* category, a 60's rock song, which contains extensive variations and use of drum fills for which there are no similar patterns in our dictionary. In order to convert the provided ground-truth annotations to bar length drum pattern representations of the same format as those in our pattern dictionary, we annotated the beat and downbeat times in a semi-automatic process using Sonic Visualiser [3] and a Vamp-plugin implementation [5] of Matthew Davies' beat-tracker [4].

3.2.2 Results

The results for the ENST-Drums tracks are given in Table 3. The system's performance strongly varies by track. Our system performs particularly well on the disco and rock genre recordings (F-measure 0.479-0.924), for which our pattern dictionary contains very similar patterns. The shuffle-blues and hard-rock patterns perform much worse (F-measure 0.037–0.525), which is largely due to the fact that they utilise patterns outside our dictionary, bringing the mean F-measure down to 0.563. In order to understand the impact of out-of-dictionary patterns, Table 3 also provides the maximum possible F-measure F_{max} calculated from our dictionary by choosing the transcription that results in the highest F-measure for each bar, and computing the overall F-measure of this transcription.

For example, ENST recording 069 only achieves an F score of 0.288, falling short of $F_{max} = 0.583$, as it mostly consists of a typical shuffle drum pattern utilising the ride cymbal which is outside of the dictionary. However, the pattern which the system predicts is in fact one that contains a ride cymbal, from a total of five (see Figure 2). The hard rock recordings make extensive use of the open hi-hat, which is not utilised in the same fashion in our dictionary; here, the classifier most often predicts an empty bar (hence the very low scores). Note that all scores are obtained on a very diverse set of 6 drum and cymbal types.

For comparison, we obtained an implementation of an existing drum transcription method by Tanghe [23] and ran it on the ENST recordings, using the default pre-trained model. Since Tanghe's method only considers bass drum, snare drum and hi-hat, we constrain the evaluation to those drum types, and map the open and closed hi-hat events from our algorithm to single hi-hat events. Table 4 shows that our system has an F-measure of 0.73; Tanghe's system performs better overall (0.82), which is largely due to excellent bass drum detection. Note however that our system obtains better performance for the snare drum (F-measure 0.74 vs 0.70) particularly with respect to precision (0.93 vs

[5] `http://vamp-plugins.org/plugin-doc/qm-vamp-plugins.html#qm-barbeattracker`

	genre	tempo	detected drum events R	P	F	F_{max}
038	disco	slow	72.6	84.9	78.3	86.7
039	disco	medium	90.2	94.8	92.4	95.8
040	disco	fast	93.1	87.1	90.0	100.0
044	rock	slow	48.1	47.6	47.9	59.8
045	rock	medium	52.7	47.5	50.0	58.5
046	rock	fast	54.5	52.5	53.5	63.6
055	disco	slow	75.9	63.8	69.3	98.3
061	rock	slow	93.8	84.3	88.8	92.5
069	SB	slow	25.6	32.8	28.8	58.3
070	SB	medium	50.0	55.4	52.5	59.5
075	HR	slow	1.9	50.0	3.7	58.7
076	HR	medium	3.8	100.0	7.4	53.2
085	SB	slow	49.5	49.0	49.2	79.5
116	minus-one (60s rock)		76.8	77.1	77.0	81.7
mean			56.3	66.2	56.3	74.7

Table 3. Real drums without accompaniment: results in percent for ENST-Drums dataset. SB: shuffle-blues; HR: hard rock.

method	metric	bd	sd	hh	overall
Proposed	R	70.2	62.0	73.1	69.9
	P	60.6	92.7	83.5	76.3
	F	65.1	**74.3**	77.9	73.0
Tanghe et al.	R	87.0	65.0	89.8	83.8
	P	99.3	75.8	73.9	80.6
	F	**92.8**	70.0	81.1	**82.1**

Table 4. Real drums without accompaniment: Results in percent for drum classes reduced to bd, sd, hh (including ho) for comparison with Tanghe et al. [23].

0.76). With a larger dictionary, our method would be able to capture more details, such as drum fills, so we expect a similar system with larger dictionary to perform better.

3.3 Polyphonic Music

For the *minus-one* recording, the ENST-Drums database provides additional non-percussive accompaniment, which allows us to test our system on polyphonic music.

3.3.1 Training and Test Data

As in the previous experiment, we use the pre-trained model from all the synthesised drum data from the experiment described in Section 3.1. The test data consists of the *minus-one* recording considered in the previous experiment. We add the polyphonic accompaniment at different levels: 0dB (fully polyphonic, no attenuation), -6dB, -12dB, -18dB, -24dB and -30dB.

3.3.2 Results

The overall F-measures obtained by the system for the various levels of attenuation are detailed in Figure 4. We provide the performance of the system on the recording with no accompaniment as a baseline (overall F-measure 0.77, as in Table 3). The system's performance on all drums decays rapidly between -24 dB and -18 dB, but then stays relatively robust for the most difficult levels considered (0dB to -18dB, overall F-measure scores of 0.48–0.58).

We compare the performance of our system to Tanghe's method once more on the reduced drum type set (bd, sd, hh). It is interesting to observe that while the F-measure on

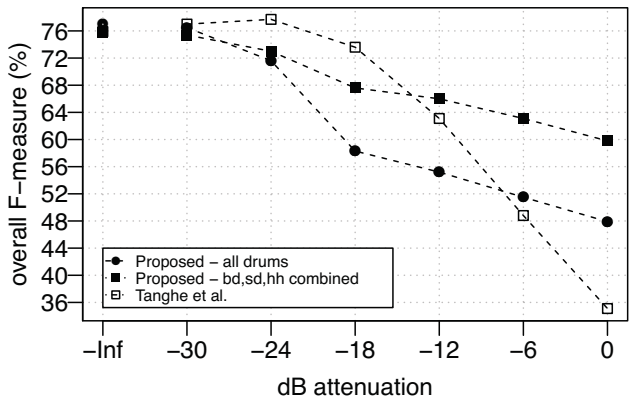


Figure 4. Overall drum events F-measure for ENST recording 116, mixed in with accompaniment at various levels of attenuation.

the pure drums is nearly the same (Tanghe: 0.76, proposed: 0.77), susceptibility to additional instruments strongly differs between the methods. The F-measure of Tanghe's method first increases for low levels of added polyphonic music (attenuation -30, -24 dB), due to the increased recall as a result of the accompaniment being detected as correct drum hits. For increasing levels of added accompaniment, performance rapidly decreases to an overall F-measure of 0.35 for 0 dB. By direct comparison, the proposed method achieves an F-measure of 0.60 even at 0 dB, demonstrating its superior robustness against high levels of accompaniment (-12, -6, 0 dB). Even for the more difficult task of recognising all 6 drum types, the proposed method (F-measure 0.48) outperforms Tanghe's.

4. DISCUSSION

Our results show not only that the proposed bar-wise drum pattern classification method is an effective, robust way to transcribe drums, but also that the first step for immediate improvement should be to increase the dictionary size in order to obtain better coverage. In addition, relaxing the strict holistic pattern approach by classifying patterns of individual instruments would allow for the recognition of combinations of patterns and hence of many new, unseen patterns. Another obvious route for improvement is to train our classifier on drum data with added polyphonic music content, which is likely to further increase robustness in polyphonic conditions.

The general approach of bar-wise drum classification is not exhausted by our particular implementation, and we expect to be able to gain further improvements by exploring different classifiers, different amounts of neighbourhood context or different basic features (e.g. non-negative matrix factorisation activations). Furthermore, to use the method in an interactive annotation system, it would be interesting to investigate bar-wise confidence scores for user guidance. Genre-specific training data could improve the performance of such systems. Finally, using more holistic features instead of single frames may also be applicable to other music informatics tasks such as chord transcription.

5. CONCLUSIONS

We have presented a novel approach to drum transcription from audio using drum pattern classification. Instead of detecting individual drums, our method first predicts whole drum patterns using an SVM classifier trained on a large collection of diverse synthetic data, and then maps the drums from the recognised patterns to the relative timestamps to achieve a transcription. The method performs very well on synthetic data, even with tempo and velocity variations on previously unseen sampled drum kits (mean pattern accuracy: 80%). Even though the pattern accuracy range differs between drum kits (50.1%–91.6%) many drum events are still classified with high precision and recall (F-measure 0.836–0.989). Unlike existing techniques, our drum detection includes open hi-hat, closed hi-hat, crash and ride cymbals, which are all reliably detected in most cases. Extending the bar patterns by one beat either side and thus obtaining overlapping patterns leads to better accuracy, mainly due to improved recognition of crash cymbals. On real drum recordings performance strongly depends on genre (F-measure for rock and disco: 0.479–0.924; hard-rock and shuffle-blues: 0.037–0.525), mainly due to the limited types of drum patterns in our current dictionary. This results in a performance slightly below that of a comparable method. However, we show that for rock music, the proposed method performs as well as the other method (F-measure: 0.77) and is substantially more robust to added polyphonic accompaniment.

6. ACKNOWLEDGEMENTS

Matthias Mauch is supported by a Royal Academy of Engineering Research Fellowship.

7. REFERENCES

[1] E. Benetos, S. Dixon, D. Giannoulis, H. Kirchhoff, and A. Klapuri. Automatic music transcription: Challenges and future directions. *Journal of Intelligent Information Systems*, 41(3):407–434, 2013.

[2] E. Benetos, S. Ewert, and T. Weyde. Automatic transcription of pitched and unpitched sounds from polyphonic music. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2014)*, May 2014.

[3] C. Cannam, C. Landone, M. B. Sandler, and J. P. Bello. The Sonic Visualiser: A visualisation platform for semantic descriptors from musical signals. In *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR 2006)*, pages 324–327, 2006.

[4] M. E. P. Davies and M. D. Plumbley. Context-dependent beat tracking of musical audio. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(3):1009–1020, 2007.

[5] S. Dixon, F. Gouyon, and G. Widmer. Towards characterisation of music via rhythmic patterns. In *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR 2004)*, pages 509–516, 2004.

[6] D. P. W. Ellis and J. Arroyo. Eigenrhythms: Drum pattern basis sets for classification and generation. In *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR 2004)*, pages 101–106, 2004.

[7] D. FitzGerald, R. Lawlor, and E. Coyle. Prior subspace analysis for drum transcription. In *Proceedings of the AES 114th International Convention*, 2003.

[8] D. Gärtner. Unsupervised Learning of the Downbeat in Drum Patterns. In *Proceedings of the AES 53rd International Conference*, pages 1–10, 2014.

[9] O. Gillet and G. Richard. ENST-Drums: An extensive audio-visual database for drum signals processing. In *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR 2006)*, pages 156–159, 2006.

[10] O. Gillet and G. Richard. Transcription and separation of drum signals from polyphonic music. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(3):529–540, 2008.

[11] M. Goto. An audio-based real-time beat tracking system for music with or without drum-sounds. *Journal of New Music Research*, 30(2):159–171, 2001.

[12] M. Mauch and S. Dixon. A corpus-based study of rhythm patterns. In *Proceedings of the 13th International Conference on Music Information Retrieval (ISMIR 2012)*, pages 163–168, 2012.

[13] M. Mauch and M. Levy. Structural change on multiple time scales as a correlate of musical complexity. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, pages 489–494, 2011.

[14] M. Miron, M. E. P. Davies, and F. Gouyon. Improving the real-time performance of a causal audio drum transcription system. In *Proceedings of the Sound and Music Computing Conference (SMC 2013)*, pages 402–407, 2013.

[15] M. Miron, M. E. P. Davies, and Fabien Gouyon. An open-source drum transcription system for Pure Data and Max MSP. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2013)*, pages 221–225. IEEE, 2013.

[16] Y. Ni, M. McVicar, R. Santos-Rodriguez, and T. De Bie. An end-to-end machine learning system for harmonic analysis of music. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(6):1771–1783, 2012.

[17] E. Pampalk, A. Flexer, and G. Widmer. Improvements of audio-based music similarity and genre classificaton. In *Proceedings of the 6th International Conference on Music Information Retrieval*, pages 634–637, 2005.

[18] J. Paulus and A. Klapuri. Drum sound detection in polyphonic music with hidden Markov models. *EURASIP Journal on Audio, Speech, and Music Processing*, 2009:14, 2009.

[19] J. K. Paulus and A. P. Klapuri. Conventional and periodic n-grams in the transcription of drum sequences. In *Proceedings of the International Conference on Multimedia and Expo (ICME 2003)*, volume 2, pages II–737. IEEE, 2003.

[20] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay. Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research*, 12:2825–2830, 2011.

[21] V. Sandvold, F. Gouyon, and P. Herrera. Percussion classification in polyphonic audio recordings using localized sound models. In *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR 2004)*, pages 537–540, 2004.

[22] J. Strong. *Drums For Dummies*. John Wiley & Sons, 2011.

[23] K. Tanghe, S. Degroeve, and B. De Baets. An algorithm for detecting and labeling drum events in polyphonic music. In *Proceedings of the 1st Annual Music Information Retrieval Evaluation Exchange (MIREX 2005)*, pages 11–15, 2005.

[24] K. Yoshii, M. Goto, and H. G. Okuno. Automatic drum sound description for real-world music using template adaptation and matching methods. In *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR 2004)*, pages 184–191, 2004.

Oral Session 3

Symbolic

This Page Intentionally Left Blank

DEVELOPING TONAL PERCEPTION THROUGH UNSUPERVISED LEARNING

Carlos Eduardo Cancino Chacón, Stefan Lattner, Maarten Grachten
Austrian Research Institute for Artificial Intelligence
{carlos.cancino,stefan.lattner,maarten.grachten}@ofai.at

ABSTRACT

The perception of tonal structure in music seems to be rooted both in low-level perceptual mechanisms and in enculturation, the latter accounting for cross-cultural differences in perceived tonal structure. Unsupervised machine learning methods are a powerful tool for studying how musical concepts may emerge from exposure to music. In this paper, we investigate to what degree tonal structure can be learned from musical data by unsupervised training of a Restricted Boltzmann Machine, a generative stochastic neural network. We show that even based on a limited set of musical data, the model learns several aspects of tonal structure. Firstly, the model learns an organization of musical material from different keys that conveys the topology of the circle of fifths (CoF). Although such a topology can be learned using principal component analysis (PCA) when using pitch-only representations, we found that using a pitch-duration representation impedes the extraction of the CoF topology much more for PCA than for the RBM. Furthermore, we replicate probe-tone experiments by Krumhansl and Shepard, measuring the organization of tones within a key in human perception. We find that the responses of the RBM share qualitative characteristics with those of both trained and untrained listeners.

1. INTRODUCTION

Modern approaches in music theory recognize that tonality can be broadly described as the organization of pitch classes into a hierarchical structure of tensions-relaxations around a tonal axis [10, 15, 16]. This conception of tonality is not limited to western tonal classical music, but can also be applied to modal music, popular music (e.g. jazz, rock) and non-western folk music [3]. This notion of tonality is not only a music theoretic construct: perceptual processing of musical stimuli in human listeners has been found to exhibit this type of organization as well [10]. Specific types of hierarchical organization of pitch classes are partly explained by acoustic attributes of pitch, especially the consonance between pairs of pitches [10], suggesting that low-level processing of acoustic stimuli may be relevant for the perception of tonal structure.

However, tonal structure is not only reflected in the physical attributes of pitch, it is also manifest in the statistical properties of music, such as the duration and frequency of occurrence of pitches [17], as illustrated in Figure 1. As Saffran et al. have shown [14], human listeners (including infants) are sensitive to such statistical regularities, and this leads to the view that tonal perception may be shaped by (long time) exposure to music exhibiting statistical regularities regarding frequency of occurrence of pitches, rhythmic emphasis, the position of occurrence within musical phrases, and possibly other aspects [9].

It is this process, the formation of tonal structure through exposure to musical stimuli, that we focus on in this paper. We choose a particularly straightforward approach, using a Restricted Boltzmann Machine (RBM) [6] to learn the probability distribution of melodic sequences, represented as n-grams of notes. In a first explorative experiment, we examine to what degree the feature space learned by the RBM is musically meaningful. Using the resemblance of the feature space to the circle of fifths as a quantitative criterion, we investigate the impact of the n-gram length, and compare pitch-only input representations to input representations that include both pitch and duration. In a second experiment, we use the RBM to simulate listener ratings in a probe tone test, and compare the results to ratings from human listeners of different skill levels.

The structure of the paper is as follows: In Section 2, we discuss prior work on the induction of tonal structure using computational models. Section 3 relates the different aspects of the unsupervised learning task to various perceptual mechanisms that are assumed to be at play in the perception of tonal structure. Section 4 briefly describes the RBM model, the data used for training the model, and representation of the data. The experiments on tonal organization and the organization of pitches are described in Sections 5 and 6, respectively. Conclusions and future directions are presented in Section 7.

2. RELATED WORK

The idea of studying the perception of tonal structure by using computational models to simulate the enculturation process is not new. For example, Tillmann et al. [18] use a hierarchical self-organizing map (SOM) [8] to learn representations of tonal structure from pitch-class representations of chord sequences. They find that their model is able

to develop an organization comparable to that of empirical data gathered from various studies on human perception of tonality. Leman [12] presents an alternative approach to modeling the perception of tonality. He employs a psychoacoustic model in combination with a SOM to learn tonal representations starting from acoustic data. Furthermore, Toiviainen & Krumhansl examined the perception of musical scales by projecting human ratings to the feature space of a SOM, which was trained on scale profiles of Krumhansl [19].

A commonality among the mentioned works is the choice of the self-organizing map as a model for accommodating the learning process. The reason for this preference may be that both the spatial mapping of the data, and the competitive learning algorithm employed by the SOM, are biologically plausible characteristics of the human sensory cortex [7]. The RBM model used in the work presented here, is not explicitly presented as (nor was it designed to be) a biologically plausible model of learning in the brain. Nevertheless RBMs and deep belief nets based on RBMs, in combination with sparseness constraints on the activation of hidden units, are able to learn features from visual data that strongly resemble receptive fields of neurons in the visual cortex [11]. As such, RBMs prove to be a valid computational modeling approach for learning biologically plausible representations from musical data.

A fundamental difference between SOMs and RBMs is that in the former, the hidden units represent points in an explicitly defined low-dimensional feature space. In RBMs, the feature space is defined by the set of all possible combinations of hidden unit activations, such that each hidden unit represents a dimension of the feature space. This allows for representations of data instances as a (non-linear) combination of features. The topology of this high-dimensional feature space can be visualized in a 2-D space using PCA.

3. PERCEPTUAL MECHANISMS

As argued by Smith and Schmuckler [17], perceptual processes like *discrimination*, *differentiation* and *organization* play an important role in the perception of musical tonality. In this Section, we will briefly describe these processes, and show how they can be related to formal aspects of the computational modeling methods, such as the choice of input data representation, and the topology of the feature space being learned.

Perceptual discrimination refers to the sensitivity of a system to differences along some perceivable stimulus dimension. In computational learning models, this relates to the form of input data representation. In general, the type of relevant input features depends heavily on the respective learning task [1]. Musical data comprises much context-dependent information that can not be trivially inferred from low-level representations. To decide on an appropriate representation is thus not always an easy task. For instance, pitch content can be represented in several ways, such as frequency spectra, MIDI note numbers, or pitch classes. In our current experiments, we use MIDI note numbers as well as pitch class representations. Duration is encoded separately from pitch. An advantage of this over combined pitch-duration representations (e.g. piano-roll notation) is that the n-gram size is specified in the number of notes, rather than an absolute time interval. This allows for comparing pitch-only to pitch-duration representations. The input data will be referred to as Input Space (IS), and will be described in more detail in Section 4.3.

Differentiation is a higher order ability that refers to the segregation of the perceived stimuli into elements on the basis of its discriminable differences [17]. In an unsupervised model we can identify this ability as the capacity of the system to segregate the data in the IS into clusters in the Feature Space (FS). In the context of tonality, an example of differentiation would be the capacity of an unsupervised model to cluster the data in the FS in such a way that each cluster represents a musical key. A measure of quality of this clustering would then be the variance of each cluster, as smaller variances imply a better differentiation of the data with respect to each class.

Organization builds on the concept of differentiation, as it establishes relations between the differentiated elements, as well as the nature of the relations themselves. In an unsupervised model, this can be understood as the topology of the FS. In this way, geometric features such as the distance between clusters, as well as the relative position between them can express similarity.

Bharucha [10, cited by Krumhansl] recognizes two types of hierarchies regarding musical tonality. *Event hierarchies* refer to the functional significance of single note events in a specific musical context, while *tonal hierarchies* account for the abstract musical structure in a particular culture or genre, e.g. the functional significance of all elements of a pitch class relative to all other pitch classes.

In our case, we compare the organization of the data in the FS to the circle of fifths, a well known music theoretical construct that explains the relations and the neighborhood of keys [15]. As a measure of quality we use the Procrustes Distance (PD) [4] of the centroids of the clustered data in the feature space with respect to the CoF.

4. METHODS

4.1 Restricted Boltzmann Machine

A Restricted Boltzmann Machine is a stochastic Neural Network (NN) with two layers, a visible layer with units $\mathbf{v} \in \{0,1\}^r$ and a hidden layer with units $\mathbf{h} \in \{0,1\}^q$ [6]. The units of both layers are fully interconnected with weights $\mathbf{W} \in \mathbb{R}^{r \times q}$, while there are no connections between the units within a layer. Given a visible vector $\mathbf{v}$, the free energy of the model can be calculated as:

$$\mathcal{F}(\mathbf{v}) = -\mathbf{a}^\intercal \mathbf{v} - \sum_i \log\left(1 + e^{(b_i + \mathbf{W}_i \mathbf{v})}\right), \quad (1)$$

where $\mathbf{a} \in \mathbb{R}^r$ and $\mathbf{b} \in \mathbb{R}^q$ are bias vectors, and $\mathbf{W}_i$ is the i-th row of the weight matrix.

Given $\mathbf{v}$, a sample of $\mathbf{h}$ can be obtained from its conditional activation probability, given by:

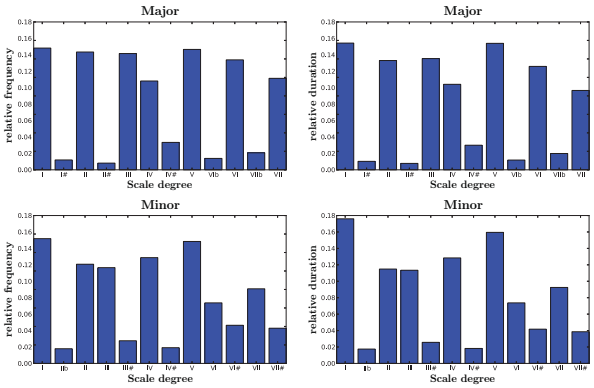


Figure 1. Occurrence and duration distributions of the fugues from Bach's Well Tempered Clavier.

$$p(\mathbf{h} = \mathbf{1} \mid \mathbf{v}) = \sigma\left(\mathbf{b} + (\mathbf{v}^\intercal \mathbf{W})^\intercal\right), \qquad (2)$$

where $\sigma(x) = 1/(1+e^{-x})$ is the logistic sigmoid function.

In experiment 1, we consider the conditional activation probability of vector $\mathbf{h}$ as the result of the projection of $\mathbf{v}$ into the FS. In the second experiment, we calculate the energy using Eq. (1).

4.1.1 Training

We train the model with 200 hidden units for 1000 epochs with Contrastive Divergence (CD) [6], using 3 Gibbs sampling steps and a mini-batch size of 500 for the weight updates. The learning rate is set to 0.01 and the momentum to 0.3. These parameters were empirically selected according to the rules of thumb suggested by Hinton in [5]. In addition, we use the well-known L2 weight-decay regularization which penalizes large weight coefficients.

Based on properties of neural coding, sparsity and selectivity can be used as constraints for the optimization of the training algorithm [2]. Sparsity encourages competition between hidden units, and selectivity prevents overdominance by any individual unit. These constraints are used in our training, with a linear falloff of its influence over the first 200 epochs from 50% to 30%.

4.2 Training Corpus

J. S. Bach's Well Tempered Clavier (WTC), composed between 1722 and 1742, is widely recognized as one of the most influential works in music history [15]. It is also one of the most important works that systematically spans the whole range of major and minor keys, and is therefore well-suited for experiments on tonality. In this paper, we use MIDI versions of the 48 fugues of the WTC as corpus, encoded by David Huron and taken from the KernScores website (http://kern.ccarh.org). Each fugue is decomposed into its voices (two to five), and we consider each voice as a single monophonic melody in its respective key. In Figure 1, the distributions of the occurrence and duration of the notes of the WTC are shown. These distributions are similar to the key profiles by Krumhansl & Kessler [19].

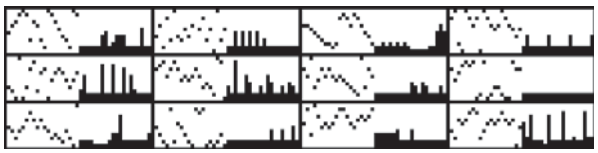


Figure 2. Twelve random *pitch-duration* training instances of the WTC corpus as 20-grams before linearization. Notes are ordered horizontally, the vertical dimension accounts for pitch and duration values, respectively. The left part of each instance shows the one-out-of-m pitch representation of 20 consecutive notes, the right part shows the corresponding duration representation.

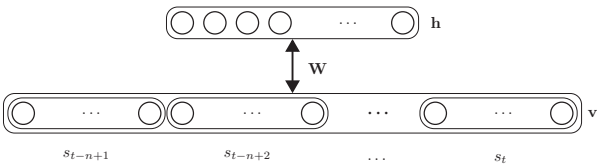


Figure 3. The RBM architecture used. An input vector $\mathbf{v}$ is constituted by a linearized n-gram, where s_j is a binary representation of note j.

4.3 Input Representation

From the monophonic melodies, we construct a set of n-grams by using a sliding window of size n and a step size of 1. Depending on the experiment, we either use only pitch information, or we use both the pitch and duration of the notes. In the first case, an n-gram is a concatenation of n bit vectors of size m, where the i-th bit vector is a one-out-of-m representation of the pitch of note i.

In the second case, n additional vectors are added to the n-gram, where the i-th vector now represents the duration of the i-th note (see the right half of the instances shown in Figure 2). Such a duration vector is constructed by quantizing all durations of a melody into 12 bins and by relating each of those to one of 12 units. A duration that falls into bin k is represented by activating units 1 to k. After linearization, the resulting n-gram constitutes the visible vector $\mathbf{v}$, as illustrated in Figure 3.

5. TONAL ORGANIZATION

In this experiment, we examine the ability of an RBM to learn tonal relationships between n-grams. To that end, we project the FS learned by the RBM into a two-dimensional space using Randomized Principal Component Analysis (rPCA) [13]. As the CoF is the underlying music theoretical construct for the relationships between keys, we are interested to what degree we can approximate the CoF topology. As a baseline, we compare this projection to a direct projection of the IS, again using rPCA.

5.1 Training

We encode the WTC corpus as described in 4.3. As keys are characterized by distributions of pitch classes, the pitch range is set to $m = 12$. In order to examine the organization ability of the RBM under different settings, we use

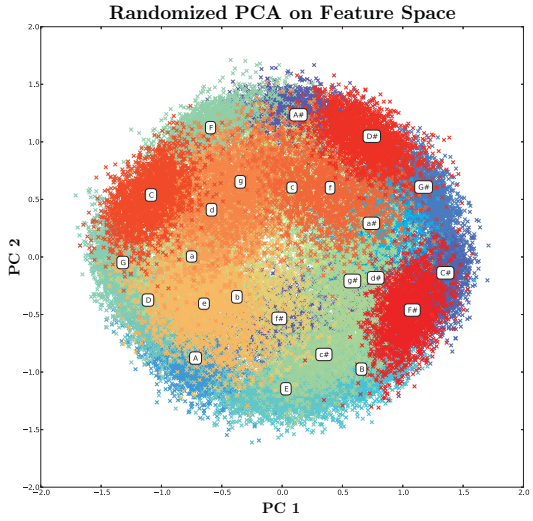


Figure 4. 2-D visualization of n-grams in the FS using rPCA. N-grams belonging to a key have the same color, each centroid is marked with the corresponding cluster's key label. (Best viewed in color)

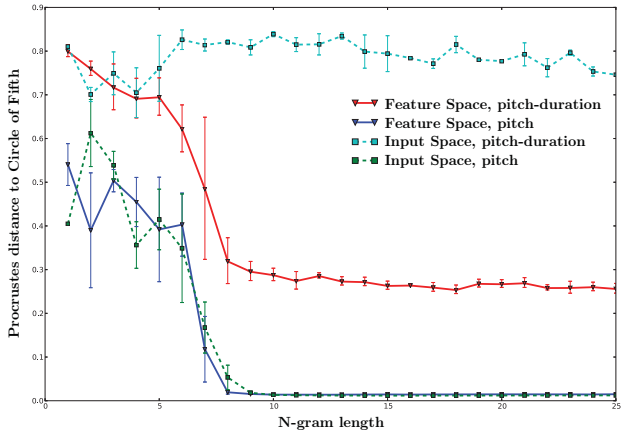


Figure 5. The average Procrustes Distances from major key centroids to the major CoF of 5 runs for different n-gram lengths after rPCA on the IS and on the FS. *pitch* and *pitch-duration* representations are used as input.

n-grams of various lengths, and also compare *pitch* and *pitch-duration* representations.

5.2 Evaluation

We use rPCA to project all n-grams in both the IS and the FS into a two-dimensional space. In this space, for each key we determine the mean of all n-grams created from pieces in that key. The organization of those centroids is then compared to the organization of keys in the CoF by computing the PD of both shapes, separately for major and minor keys. To make different expansions of data points in space comparable, the PD is finally divided by the perimeter of the target CoF.

5.3 Results and Discussion

Figure 4 shows the organization of n-grams in the FS. Cluster centers are organized similarly to how keys are organized in the CoF, which is consistent with the representations of the probe tone ratings obtained by Krumhansl and Kessler [9, 10]. Note that relative minors tend to be shifted counterclockwise with respect to their major counterparts. This occurs in Krumhansl's results as well [10, pp. 43], and can be explained by two factors, namely the alteration of the sixth degree in the melodic minor scale, which is identical to the seventh degree of the dominant of the relative major counterpart (e.g. the melodic Am scale shares the F# with the G major scale, the dominant of C major), and due to the tonal modulations concerning the form of the piece (e.g. fugues in minor keys tend to have certain passages in the relative major, while fugues in major keys tend to have passages in both the relative minor and the dominant).

Figure 5 shows, that the Procrustes Distance to the CoF tends to stabilize at a minimum with an n-gram length of about nine. This can be explained by the fact that n-grams of that length contain enough information to obtain the respective distribution of a key well enough. Adding duration information clearly impedes the organization of clusters in a CoF topology. As the occurrence of notes in the WTC is strongly correlated to their absolute duration (see Figure 1), and rhythmic information is not directly linked to the CoF organization, this is not unexpected. Interestingly, for larger n-gram sizes the FS of the RBM is not disrupted as much by the inclusion of distractive information as the rPCA on the IS.

6. ORGANIZATION OF PITCHES

A probe-tone test, proposed by Krumhansl et al. [9, 10], consists of a set of *musical stimuli* (such as scales, chord cadences, or musical pieces) that unambiguously instantiate a specific key, and a set of *probe tones*, typically the set of 12 pitch classes. Listeners are then required to rate on a numerical scale, from 1 ("very bad") to 7 ("very good"), how well the probe tones fit the musical stimulus. In order to explore the hierarchical event organization of pitches induced by the RBM, we compare our model with a particular probe tone test conducted by Krumhansl and Shepard [10, cited by Krumhansl]. In this specific experiment, the musical stimulus consisted of an incomplete C major scale (in both ascending and descending contexts), and listeners were asked to give a numerical rating of the degree to which each probe tone fits the scale. The stimuli of this particular setup are illustrated in part Figure 6 a), while the probe tones are shown in Figure 6 c). The participants of the experiment were divided in three groups according to their number of years of formal musical training.

6.1 Training

As we are only interested in the ability of the model to learn tonal hierarchies in major and minor mode, we transpose all melodies to C major and C minor, respectively. In order to remain consistent with the aforementioned experiment of Krumhansl & Shepard, rather than using pitch-classes, we allow the training data to be in a range of three octaves, ranging from MIDI pitch numbers 48 to 74 (such that both the stimuli and the probe tones can be represented

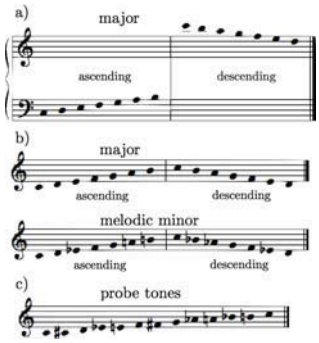


Figure 6. Stimuli/probe tones used in the probe-tone test.

without wrapping). Most of the n-grams of the transposed WTC data fall in that range, or can be transposed octave-wise to fall in that range. N-grams for which this is not possible are ignored.

Since the fugues from the WTC contain certain tonal modulations, in order to train the RBM with prototypical examples of major and minor scales, all n-grams are classified using the Krumhansl & Kessler key-finding algorithm [10, cited by Krumhansl] and those whose annotated key is not the same as that identified by the classifier (ca. 53% of the corpus) are removed. The training is executed as described in 4.1.1.

6.2 Evaluation

Two different probe tone tests are conducted. The first test aims to reproduce the setup by Krumhansl and Shepard, and thus, the stimuli consist of the major ascending (starting from C3) and descending scales (starting from C5) shown in Figure 6 a). For the second experiment, the stimuli consist of ascending and descending major and melodic minor scales, but this time both are generated in the middle C octave, as shown in Figure 6 b). For both tests, the set of probe tones consist of all notes of the chromatic scale (starting from C4) as shown in Figure 6 c). We construct n-grams of length 8, consisting of the 7 notes of the target stimulus and a probe tone as the last note. This results in visible vectors $\mathbf{v}_{pt}$ of length 36×8. The free energy corresponding to each combination of stimulus and probe tone is calculated using Eq. (1). In order to compare our results to those of human listeners, these energies are scaled using an affine transformation as follows:

$$\text{Judgment}(\mathbf{v}_{pt}) = \alpha \mathcal{F}(\mathbf{v}_{pt}) - \beta, \qquad (3)$$

where the constants α, β are selected such that the mean and the variance of the scaled energy are equal to those of the judgments reported in [10].

6.3 Results and Discussion

Figure 7 shows the results of the probe tone test, and in Table 1 the correlations of the RBM judgments with respect to those of expert and untrained listeners are presented. These results suggest that the model can learn some event hierarchy structures, such as the prevalence of diatonic over chromatic notes, similar to the judgment of trained listeners. In addition, the model develops a sense for melodic direction, preferring probe tones close to the final notes of the stimulus, which is consistent with the ratings of untrained listeners. Stimulated in the middle octave, the model is able to distinguish major and minor modes, especially the major and minor thirds reflect the characteristics of the respective diatonic triads. The model responses do not show explicit octave equivalence, since C and C' are not equally emphasized. Still it is interesting to note that a stimulus in the lower octave has implications on the pitch expectations in the middle octave, and that these implications are in correspondence with the tonal hierarchy of the key implied by the stimulus.

Group	r	p-value
Expert ascending	0.7213	0.0054
Untrained ascending	0.7942	0.0012
Expert descending	0.7985	0.0011
Untrained descending	0.8344	0.0004

Table 1. Pearson correlations and p-values for the judgments of the probe tone tests.

7. CONCLUSION

In this paper we show that tonal structure can be learned from musical data with an RBM using unsupervised training with a limited set of monophonic melodies. The model is able to reproduce the topology of the CoF using *pitch* n-gram representations of the input data. We found that for successful inference of the CoF, a minimal n-gram length of nine notes is needed, and that longer n-grams do not lead to better representations. Furthermore, although duration information profoundly disturbs the learning of tonal structure through the baseline rPCA method, the RBM model is less affected by distracting duration information.

By way of a probe tone test, we explored the organization of pitches in the context of major and minor modes. Our results show the model was able to learn several aspects of tonal structure, in particular the hierarchical prevalence of diatonic over chromatic tones. Comparing results with Krumhansl's probe tone experiments on human subjects with different levels of musical training do not yield a conclusive classification of the model: the model displays aspects of both untrained and trained subjects.

An important feature of tonal perception in trained subjects is octave equivalence. This feature was not well-reproduced by the model. It is possible that a pre-condition for octave-equivalence is the harmonic overlap of octaves. In our current setup, the overtone structure of tones is not represented. To test this hypothesis, we intend to investigate whether using harmonic tone representations leads to stronger octave-equivalence in the the model.

Furthermore we wish to investigate which factors induce more expert-like perception of tonal structure. Possible factors include the size of the training data, and the depth of the model (in terms of hidden layers).

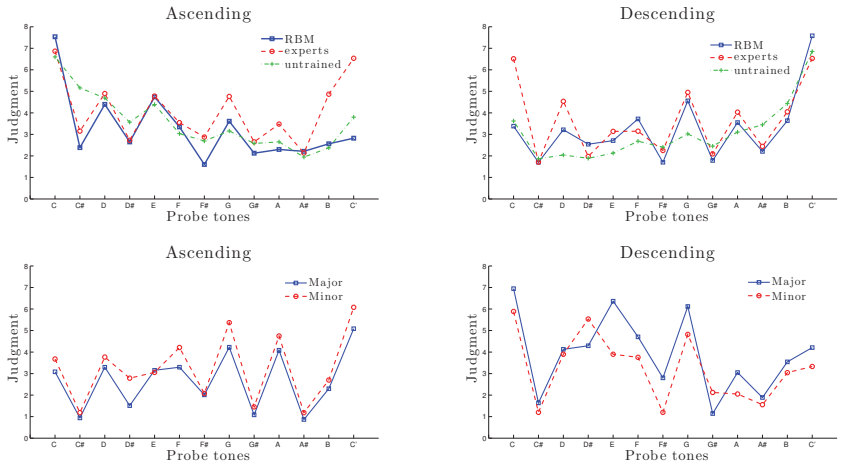


Figure 7. (Top) Comparison of the judgments for the probe tones between the RBM and human listeners for both ascending (left) and descending (right) major stimulus in the lower and upper octave, respectively. (Bottom) Comparison of the judgments for the probe tones of the RBM for both major and melodic minor stimulus in the middle octave. In all cases, responses are measured in the middle octave.

8. ACKNOWLEDGMENTS

This work is supported by the European Union Seventh Framework Programme, through the Lrn2Cre8 project (FET grant agreement no. 610859). We thank Geraint Wiggins, Kat R. Agres and Jamie Forth for valuable suggestions and commentaries on this work.

9. REFERENCES

[1] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer Verlag, 2009.

[2] H. Goh, N. Thome, and M. Cord. Biasing restricted Boltzmann machines to manipulate latent selectivity and sparsity. *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, pages 1–8, 2010.

[3] E. Gómez. *Tonal description of music audio signals*. PhD thesis, Universitat Pompeu Fabra, Barcelona, 2006.

[4] C. Goodall. Procrustes Methods in the Statistical Analysis of Shape. *Journal of the Royal Statistical Society. Series B (Methodological)*, 53(2):285–339, 1991.

[5] G. E. Hinton. A practical guide to training restricted Boltzmann machines. Tech. Report UTML TR 2010-003, Department of Computer Science, University of Toronto, 2010.

[6] G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

[7] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[8] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biol. Cybernetics*, 43:59–69, 1982.

[9] C. L. Krumhansl and L. L.Cuddy. A theory of tonal hierarchies in music. In *Handbook of Auditory Research*. Springer, New York, 2010.

[10] C. L. Krumhansl. *Cognitive foundations of musical pitch*. Cognitive foundations of musical pitch. Oxford University Press, New York, 1990.

[11] H. Lee, C. Ekanadham, and A. Y. Ng. Sparse deep belief net model for visual area V2. In *Advances in Neural Information Processing Systems 20*, pages 873–880. 2008.

[12] M. Leman. A model of retroactive tone center perception. *Music Perception*, 12(4):439–471, 1995.

[13] V. Rokhlin, A. Szlam, and M. Tygert. A randomized algorithm for principal component analysis. *arXiv.org*, page 2274, 2008.

[14] J. R. Saffran, E. K. Johnson, R. N. Aslin, and E. L. Newport. Statistical learning of tone sequences by human infants and adults. *Cognition*, 70(1):27 – 52, 1999.

[15] F. Salzer. *Structural hearing; tonal coherence in music*. New York, Dover Publications, 1962.

[16] H. Schenker. *Harmony*. University of Chicago Press, 1980.

[17] N. A. Smith and M. A. Schmuckler. The Perception of Tonal Structure Through the Differentiation and Organization of Pitches. *Journal of Exp. Psych.: Human Perception and Performance*, 30(2):268–286, 2004.

[18] B. Tillmann, J. J. Bharucha, and E. Bigand. Implicit learning of tonality: a self-organizing approach. *Psychological review*, 107(4):885–913, 2000.

[19] P. Toiviainen and C. L. Krumhansl. Measuring and modeling real-time responses to music: The dynamics of tonality induction. *Perception*, 32(6):741–766, 2003.

EXPLOITING INSTRUMENT-WISE PLAYING/NON-PLAYING LABELS FOR SCORE SYNCHRONIZATION OF SYMPHONIC MUSIC

Alessio Bazzica
Delft University of Technology
a.bazzica@tudelft.nl

Cynthia C. S. Liem
Delft University of Technology
c.c.s.liem@tudelft.nl

Alan Hanjalic
Delft University of Technology
a.hanjalic@tudelft.nl

ABSTRACT

Synchronization of a score to an audio-visual music performance recording is usually done by solving an audio-to-MIDI alignment problem. In this paper, we focus on the possibility to represent both the score and the performance using information about which instrument is active at a given time stamp. More specifically, we investigate to what extent instrument-wise "playing" (P) and "non-playing" (NP) labels are informative in the synchronization process and what role the visual channel can have for the extraction of P/NP labels. After introducing the P/NP-based representation of the music piece, both at the score and performance level, we define an efficient way of computing the distance between the two representations, which serves as input for the synchronization step based on dynamic time warping. In parallel with assessing the effectiveness of the proposed representation, we also study its robustness when missing and/or erroneous labels occur. Our experimental results show that P/NP-based music piece representation is informative for performance-to-score synchronization and may benefit the existing audio-only approaches.

1. INTRODUCTION AND RELATED WORK

Synchronizing an audio recording to a symbolic representation of the performed musical score is beneficial to many tasks and applications in the domains of music analysis, indexing and retrieval, like audio source separation [4, 9], automatic accompaniment [2], sheet music-audio identification [6] and music transcription [13]. As stated in [7], "sheet music and audio recordings represent and describe music on different semantic levels" thus making them complementary for the functionalities they serve.

The need for effective and efficient solutions for audio-score synchronization is especially present for genres like symphonic classical music, for which the task remains challenging due to the typically long duration of the pieces and a high number of instruments involved [1]. The existing solutions usually turn this synchronization problem into an audio-to-audio alignment one [11], where the score is rendered in audio form using its MIDI representation.

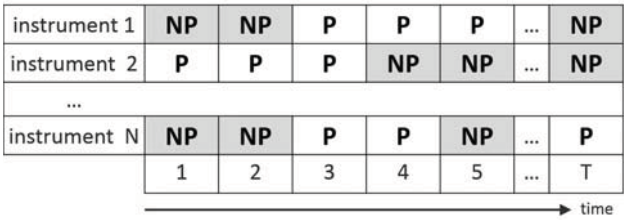


Figure 1: An illustration of the representation of a symphonic music piece using the matrix of playing/non-playing labels.

In this paper, we investigate whether sequences of playing (P) and non-playing (NP) labels, extracted per instrument continuously over time, can alternatively be used to synchronize a recording of a music performance to a MIDI file. At a given time stamp, the P (NP) label is assigned to an instrument if it is (not) being played. If such labels are available, a representation of the music piece as illustrated in Figure 1 can be obtained: a matrix encoding the P/NP "state" for different instruments occurring in the piece at subsequent time stamps. Investigating the potential of this representation for synchronization purposes, we will address the following research questions:

- RQ1: How robust is P/NP-based synchronization in case of erroneous or missing labels?
- RQ2: How does synchronizing P/NP labels behave at different time resolutions?

We are particularly interested in this representation, as P/NP information for orchestra musicians will also be present in the signal information of a recording. While such information will be hard to obtain from the audio channel, it can be obtained from the visual channel. Thus, in case an audio-visual performance is available, using P/NP information opens up possibilities for video-to-score synchronization as a means to solve a score-to-performance synchronization problem.

The rest of the paper is structured as follows. In Section 2, we formulate the performance-to-score synchronization problem in terms of features based on P/NP labels. Then, we explain how the P/NP matrix is constructed to represent the score (Section 3) and we elaborate on the possibilities for extracting the P/NP matrix to represent the analyzed performance (Section 4). In Section 5 we propose an efficient method for solving the synchronization problem. The experimental setup is described in Section 6 and in Section 7 we report the results of our

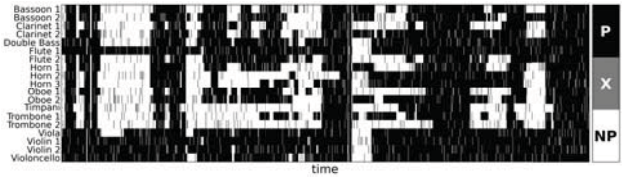


Figure 2: Example of a M_{PNP} matrix with missing labels.

experimental assessment of the proposed synchronization methodology and provide answers to our research questions. The discussion in Section 8 concludes the paper.

2. PROBLEM DEFINITION

Given an audio-visual recording of a performance and a symbolic representation of the performed scores, we address the problem of synchronizing these two resources by exploiting information about the instruments which are active over time.

Let $L = \{-1, 0, 1\}$ be a set encoding the three labels non-playing (NP), missing (X) and playing (P). Let $M_{\mathrm{PNP}} = \{m_{ij}\}$ be a matrix of $N_{\mathrm{I}} \times N_{\mathrm{T}}$ elements where N_{I} is the number of instruments and N_{T} is the number of time points at which the P/NP state is observed. The value of $m_{ij} \in L$ represents the state of the i-th instrument observed at the j-th time point ($1 \leq i \leq N_{\mathrm{I}}$ and $1 \leq j \leq N_{\mathrm{T}}$). An example of M_{PNP} is given in Figure 2.

We now assume that the matrices $M^{\mathrm{AV}}_{\mathrm{PNP}}$ and $M^{\mathrm{S}}_{\mathrm{PNP}}$ are given and represent the P/NP information respectively extracted by the audio-visual recording and the sheet music. The two matrices have the same number of rows and each row is associated to each instrumental part. The number of columns, i.e. observations over time, is in general different. The synchronization problem can be then formulated as the problem of finding a time map $f_{sync} : \{1 \dots N^{\mathrm{AV}}_{\mathrm{T}}\} \rightarrow \{1 \dots N^{\mathrm{S}}_{\mathrm{T}}\}$ linking the observation time points of the two resources.

3. SCORE P/NP REPRESENTATION

For a given piece, we generate one P/NP matrix $M^{\mathrm{S}}_{\mathrm{PNP}}$ for the score relying on the corresponding MIDI file as the information source.

We start generating the representation of the score by parsing the data of each available track in the given MIDI file. Typically, one track per instrument is added and is used as a symbolic representation of the instrumental part's score. More precisely, when there is more than one track for the same instrument (e.g. Violin 1, Violin 2 - which are two different instrumental parts), we keep both tracks as separate. In the second step, we use a sliding window that moves along the MIDI file and derive a P/NP label per track and window position. A track receives a P label if there is at least one note played within the window. We work with the window in order to comply with the fact that a played note has a beginning and end and therefore lasts for an interval of time. In this sense, a played note is registered when there is an overlap between the sliding window and the play interval of that note.

The length of the window is selected such that short rests within a musical phrase do not lead to misleading P-NP-P switches. We namely consider a musician in the "play" mode if she is within the "active" sequence of the piece with respect to her instrumental part's score, independently whether at some time stamps no notes are played. In our experiments, we use a window length of 4 seconds which has been determined by empirical evaluation, and a step-size of 1 second. This process generates one label per track every second.

In order to generalize the parameter setting for window length and offset, we also related them to the internal MIDI file time unit. For this purpose, we set a reference value for the tempo. Once the value is assigned, the sliding window parameters are converted from seconds to beats. The easiest choice is adopting a fixed value of tempo for every performance. Alternatively, when an audio-visual recording is available, the reference tempo can be estimated as the number of beats in the MIDI file divided by the length of the recording expressed in minutes. A detailed investigation of different choices of the tempo is reported in [6].

4. PERFORMANCE P/NP REPRESENTATION

While an automated method could be thought of to extract the P/NP matrix $M^{\mathrm{AV}}_{\mathrm{PNP}}$ from a given audio-visual recording, developing such a method is beyond the scope of this paper. Instead, our core focus is assessing the potential of such a matrix for synchronization purposes, taking into account the fact that labels obtained from real-world data can be noisy or even missing. We therefore deploy two strategies which mimic the automated extraction of the $M^{\mathrm{AV}}_{\mathrm{PNP}}$ matrices. We generate them: (i) artificially, by producing (noisy) variations of the P/NP matrices derived from MIDI files (Section 4.1), and (ii) more realistically, by deriving the labels directly from the visual channel of a recording in a semi-automatic way (Section 4.2).

4.1 Generating synthetic P/NP matrices

The first strategy produces synthetic P/NP matrices by analyzing MIDI files as follows. Similarly to the process of generating a P/NP matrix for the score, we apply a sliding window to the MIDI file and extract labels per instrumental track at each window position. This time, however, time is randomly warped, i.e. the sliding window moves over time with non-constant velocity. More specifically, we generate random time-warping functions by randomly changing slope every 3 minutes and by adding a certain amount of random noise in order to avoid perfect piecewise linear functions. In a real audio-visual recording analysis pipeline, we expect that erroneous and missing P/NP labels will occur. Missing labels may occur if musicians cannot be detected, e.g. because of occlusion or leaving the camera's angle of view in case of camera movement. In order to simulate such sources of noise, we modify the generated P/NP tracks by randomly flipping and/or deleting predetermined amounts of labels at random positions of the P/NP matrices.

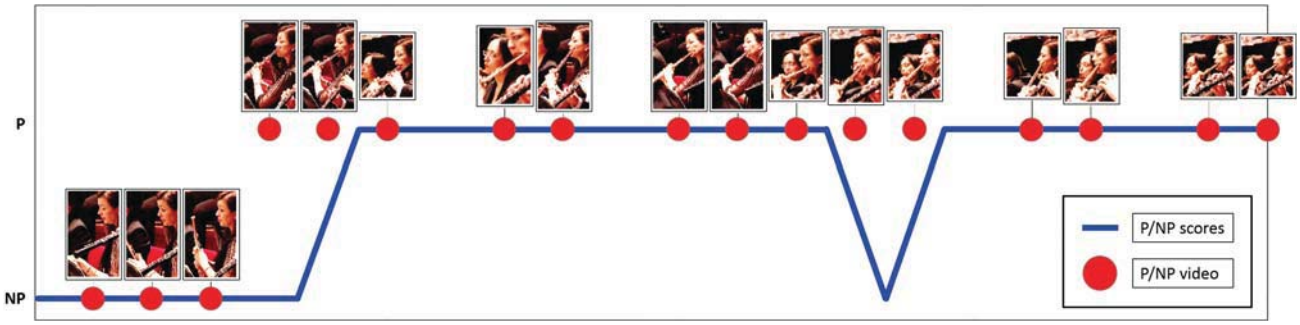


Figure 3: Example of P/NP labels extracted from the visual channel (red dots) and compared to labels extracted by the score (blue line).

4.2 Obtaining P/NP matrices from a video recording

The second strategy more closely mimics the actual video analysis process and involves a simple, but effective method that we introduce for this purpose. In this method, we build on the fact that video recordings of a symphonic music piece are typically characterized by regular close-up shots of different musicians. From the key frames representing these shots, as illustrated by the examples in Figure 4, it can be inferred whether they are using their instrument at that time stamp or not, for instance by investigating their body pose [14].

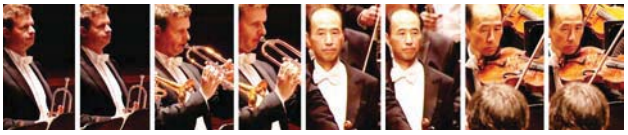


Figure 4: Examples of body poses indicating playing/non-playing state of a musician.

In the first step, a key frame is extracted every second in order to produce one label per second, as in the case of the scores. Faces are detected via off-the-shelf face detectors and upper-body images are extracted by extending the bounding box's areas of face detector outputs. We cluster the obtained images using low-level global features encoding color, shape and texture information. Clustering is done using k-means with the goal to isolate images of different musicians. In order to obtain high precision, we choose a large value for k. As a result, we obtain clusters mostly containing images of the same musician, but also multiple clusters for the same musician. Noisy clusters (those not dominated by a single musician) are discarded, while the remaining are labeled by linking them to the correspondent track of the MIDI file (according to the musician's instrument and position in the orchestra, i.e. the instrumental part). In order to label the upper-body images as P/NP, we generate sub-clusters using the same features as those extracted in the previous (clustering) step. Using once again k-means, but now with k equal to 3 (one cluster meant for P labels, one for NP and one extra label for possible outliers), we build sub-clusters which we label as either playing (P), non-playing (NP) or undefined (X). Once the labels for every musician are obtained, they are aggregated by instrumental part (e.g. the labels from all the Violin 2 players are combined by majority voting). An example of a P/NP subsequence extracted by visual analysis is given in Figure 3.

5. SYNCHRONIZATION METHODOLOGY

In this section, we describe the synchronization strategy used in our experiments. The general idea is to compare configurations of P/NP labels for every pair of performance-score time points and produce a distance matrix. The latter can then serve as input into a synchronization algorithm, for which we adopt the well-known dynamic time warping (DTW) principle. This implies we will not be able to handle undefined amounts of repeats of parts of the score. However, this is a general issue for DTW also holding for existing synchronization approaches, which we consider out of the scope of this paper.

In order to find the time map between performance and score, we need to solve the problem of finding time links between the given $M^{\mathrm{AV}}_{\mathrm{PNP}}$ and $M^{\mathrm{S}}_{\mathrm{PNP}}$ matrices. To this end, we use a state-of-the-art DTW algorithm [12].

5.1 Computing the distance matrix

Ten Holt et. al. [12] compute the distance matrix through the following steps: (i) both dimensions of the matrices are normalized to have zero mean and unit variance, (ii) optionally a Gaussian filter is applied, and (iii) pairs of vectors are compared using the city block distance. In our case, we take advantage of the fact that our matrices contain values belonging to the finite set of 3 different integers, namely the set L introduced in Section 2. This enables us to propose an alternative, just as effective, but more efficient method to compute the distance matrix.

Let $\boldsymbol{m}^{\mathrm{AV}}_j$ and $\boldsymbol{m}^{\mathrm{S}}_k$ be two column vectors respectively belonging to M^{AV}_{PNP} and M^{S}_{PNP}. To measure how (dis-)similar those two vectors are, we define a *correlation score* s_{jk} as follows:

$$s_{jk} = \mathrm{corr}(\boldsymbol{m}^{\mathrm{AV}}_j, \boldsymbol{m}^{\mathrm{S}}_k) = \sum_{i=1}^{N_{\mathrm{I}}} m^{\mathrm{AV}}_{ij} \cdot m^{\mathrm{S}}_{ik}$$

From such definition, it follows that a pair of observed matching labels add a positive unitary contribution. If the observed labels do not match, the added contribution is unitary and negative. Finally, if one or both labels are not observed (i.e. at least one of them is X), the contribution is 0. Hence, it also holds $-N_{\mathrm{I}} \leq s_{jk} \leq +N_{\mathrm{I}}$. The maximum is reached only if the two vectors are equal. Correlation scores can be efficiently computed as dot-product of the given P/NP matrices, namely as $(M^{\mathrm{AV}}_{\mathrm{PNP}})^{\top} M^{\mathrm{S}}_{\mathrm{PNP}}$.

The distance matrix $D = \{d_{jk}\}$, whose values are zero when the compared vectors are equal, can now be computed as $d_{jk} = N_{\mathrm{I}} - s_{jk}$. As a result, D will have $N^{\mathrm{AV}}_{\mathrm{T}}$

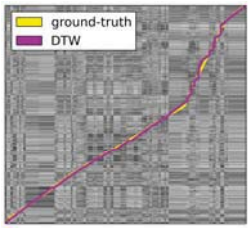
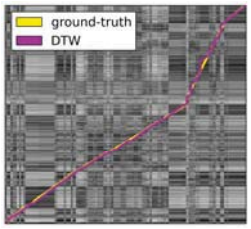
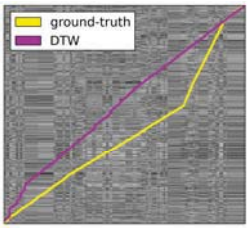
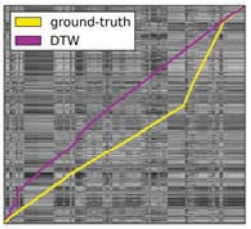
noisy M_{PNP}		very noisy M_{PNP}	
			
Ten Holt et. al.	our method	Ten Holt et. al.	our method

Table 1: Comparing our distance matrix definition to Ten Holt et. al. [12]. By visual inspection, we observe comparable alignment performances. However, the computation of our distance matrix is much faster.

rows and N_{T}^{S} columns. When the correlation is the highest, namely equal to N_{I}, the distance will be zero.

Our approach has two properties that make the computation of D fast: D is computed via the dot product and it contains integer values only (as opposed to standard methods based on real-valued distances). As shown in Table 1, both the distance matrix proposed in [12] and using our definition produce comparable results. Since our method allows significantly faster computation (up to 40 times faster), we adopt it in our experiments.

5.2 Dynamic Time Warping

Once the distance matrix D is computed, the time map between $M_{\text{PNP}}^{\text{AV}}$ and $M_{\text{PNP}}^{\text{S}}$ is determined by solving the optimization problem: $P^{\star} = \arg\min_P \text{cost}(D, P)$ where $P = \{(p_\ell \rightsquigarrow p_{\ell+1})\}$ is a path through the items of D having a cost defined by the function $\text{cost}(D, P)$. More specifically, $p_\ell = (i_\ell^{\text{AV}}, i_\ell^{\text{S}})$ is a coordinate of an element in D. The cost function is defined as $\text{cost}(D, P) = \sum_{\ell=1}^{|P|} d_{i_\ell^{\text{AV}}, i_\ell^{\text{S}}}$ The aforementioned problem is efficiently solved via dynamic programing using the well-known dynamic time warping (DTW) algorithm. Examples of $P^{\star}$ paths computed via DTW are shown in the figures of Table 1.

Once $P^{\star}$ is found, the time map f_{sync} is computed through the linear interpolation of the correspondences in $P^{\star}$, i.e. the set of coordinates $\{p_\ell^{\star} = (i_\ell^{\text{AV}}, i_\ell^{\text{S}})\}$. This map allows to define correspondences between the two matrices, as shown in the example of Figure 5.

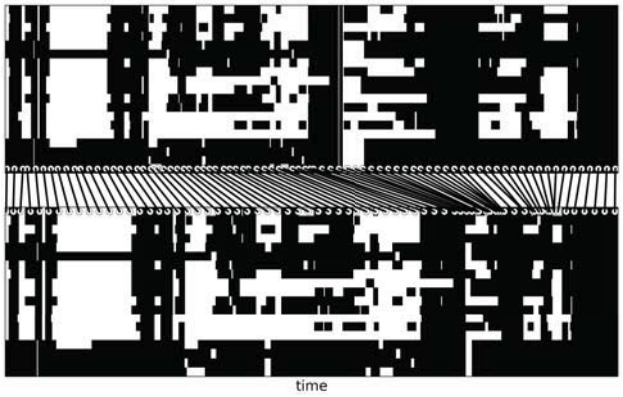


Figure 5: Example of produced alignment between two fully-observed M_{PNP} matrices.

6. EXPERIMENTAL SETUP

In this section, we describe our experimental setup including details about the dataset. In order to ensure the reproducibility of the experiments, we release the code and share the URLs of the analyzed freely available MIDI files [1] .

We evaluate the performances of our method on a set of 29 symphonic pieces composed by Beethoven, Mahler, Mozart and Schubert. The dataset consists of 114 MIDI files. Each MIDI file contains a number of tracks corresponding to different parts performed in a symphonic piece. For instance, first and second violins are typically encoded in two different parts (e.g. "Violin 1" and "Violin 2"). In such a case, we keep both tracks separate since musicians in the visual channel can be labeled according to the score which they perform (and not just by their instrument). We ensured that the MIDI files contain tracks which are mutually synchronized (i.e. MIDI files of type 1). The number of instrumental parts, or MIDI tracks, ranges between 7 and 31 and is distributed as shown in Figure 7.

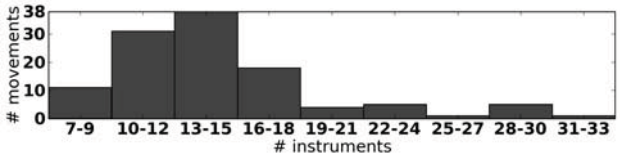
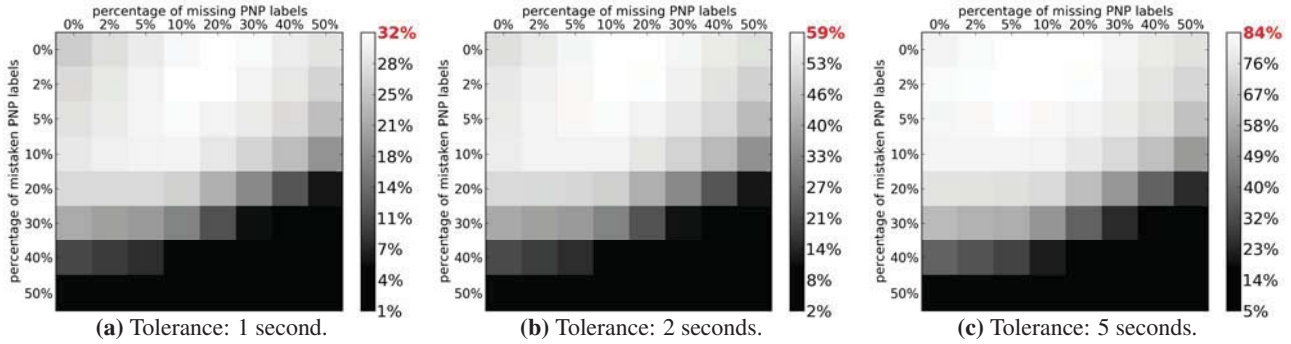


Figure 7: Distribution of the number of instrumental parts across performances in the data set.

For each MIDI file, we perform the following steps. First, we generate one $M_{\text{PNP}}^{\text{S}}$ matrix using a fixed reference tempo of 100 BPM. The reason why we use the same value for every piece is that we evaluate our method on artificial warping paths, hence we do not need to adapt the sliding window parameters to any actual performance. Then we generate one random time-warping function which has two functions: (i) it is used as ground-truth when evaluating the alignment performance, and (ii) it is used to make one time-warped P/NP matrix $M_{\text{PNP}}^{\text{AV}}$. The latter is used as template to build noisy copies of $M_{\text{PNP}}^{\text{AV}}$ and evaluate the robustness of our method. Each template P/NP matrix is used to generate a set of noisy P/NP matrices which are affected by different pre-determined amounts of noise. We consider two sources of noise: mistaken and missing labels. For both sources, we generate

[1] http://homepage.tudelft.nl/f8j6a/ISMIR2014baz.zip



(a) Tolerance: 1 second. **(b)** Tolerance: 2 seconds. **(c)** Tolerance: 5 seconds.

Figure 6: Average matching rates as a function of the percentage of mistaken and/or missing labels at different tolerance thresholds.

the following percentages of noisy labels: 0% (noiseless), 2%, 5%, 10%, 20%, 30%, 40% and 50%. For every pair of noise percentages, e.g. 5% mistaken + 10% missing, we create 5 different noisy versions of the original P/NP matrix [2]. Therefore, for each MIDI file, the final set of matrices has the size $1 + (8 \times 8 - 1) \times 5 = 316$. Overall, we evaluate the temporal alignment of $316 \times 114 = 36024$ P/NP sequences.

For each pair of M_{PNP} matrices to be aligned, we compute the matching rate by sampling f_{sync} and measuring the distance from the true alignment. A match occurs when the distance between linked time points is below a threshold. In our experiments, we evaluate the matching rate using three different threshold values: 1, 2 and 5 seconds.

Finally, we apply the video-based P/NP label extraction strategy described in Section 4.2 to a multiple camera video recording of the 4th movement of Symphony no. 3 op. 55 of Beethoven performed by the Royal Concertgebouw Orchestra (The Netherlands). For this performance, in which 54 musicians play 19 instrumental parts, we use the MIDI file and the correspondent performance-score temporal alignment file which are shared by the authors of [8]. The latter is used as ground truth when evaluating the synchronization performance.

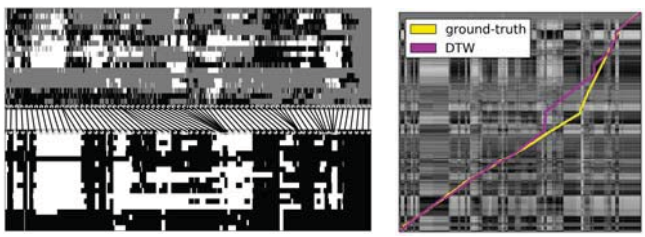
7. RESULTS

In this section, we present the obtained results and provide answers to the research questions posed in Section 1. We start by presenting in Figure 6 the computed matching rates in 3 distinct matrices, one for each threshold value. Given a threshold, the overall matching rates are reported in an 8×8 matrix since we separately compute the average matching rate for each pair of mistaken-missing noise rates. Overall, we see two expected effects: (i) the average matching rate decreases for larger amounts of noise, and (ii) the performance increases with the increasing threshold. What was not expected is the fact that the best performance is not obtained in the noiseless case. For instance, when the threshold is 5 seconds, we obtained an average matching rate of 81.7% in the noiseless case and 85.0% in the case of 0% mistaken and 10% missing labels. One possible explanation is that 10% missing labels could give more "freedom" to the DTW algorithm than the noiseless case. Such freedom may lead to a better global optimization. In order to fully understand the reported outcome, however, further investigation is needed, which we leave for future work.

As for our first research question, we conclude that the alignment through P/NP sequences is more robust to missing labels than to mistaken ones. We show this by the fact that the performance for 0% mistaken and 50% missing labels are higher than in the opposite case, namely for 50% mistaken and 0% missing labels. In general the best performance is obtained for up to 10% mistaken and 30% missing labels.

In the second research question we address the behavior at different time resolutions. Since labels are sampled every second, it is clear why acceptable matching rates are only obtained at coarse resolution (namely for a threshold of 5 seconds).

Finally, we comment on the results obtained when synchronizing through the P/NP labels assigned via visual analysis. The P/NP matrix, shown in Figure 8a, is affected by noise as follows: there are 53.95% missing and 8.65% mistaken labels.



(a) $M_{\mathrm{PNP}}^{\mathrm{AV}}$ and $M_{\mathrm{PNP}}^{\mathrm{S}}$ **(b)** DTW

Figure 8: Real data example: P/NP labels by analysis of video

We immediately notice the large amount of missing labels. This is mainly caused by the inability to infer a P/NP label at those time points when all the musicians of a certain instrumental part are not recorded. Additionally, some of the image clusters generated as described in Section 4.2 are not pure and hence labeled as X.

The obtained synchronization performance at 1, 2 and 5 seconds of tolerance are respectively 18.74%, 34.49% and 60.70%. This is in line with the results obtained with synthetic data whose performance at 10% of mistaken labels and 50% of missing for the three different tolerances are 24.3%, 44.2% and 65.9%. Carrying out the second exper-

[2] We do not add extra copies for the pair (0%,0%), i.e. the template matrix.

iment was also useful to get insight about the distribution of missing labels. By inspecting Figure 8a, we notice that such a type of noise is not randomly distributed. Some musicians are sparsely observed over time hence leading to missing labels patterns which differ from uniform distributed random noise.

8. DISCUSSION

In this paper, we presented a novel method to synchronize score information of a symphonic piece to a performance of this piece. In doing this, we used a simple feature (the act of playing or not) which trivially is encoded in the score, and feasibly can be obtained from the visual channel of an audio-visual recording of the performance. Unique about our approach is that both for the score and the performance, we start from measuring individual musician contributions, and only then aggregate up to the full ensemble level to perform synchronization. This makes a case for using the visual channel of an audio-visual recording. In the audio channel, which so far has predominantly been considered for score-to-performance synchronization, even if separate microphones are used per instrument, different instruments will never be fully isolated from each other in a realistic playing setting. Furthermore, audio source separation for polyphonic orchestral music is far from being solved. However, in the visual channel, different players are separated by default, up to the point that a first clarinet player can be distinguished from a second clarinet player, and individual contributions can be measured for both.

Our method still works at a rough time resolution, and lacks the temporal sub-second precision of typical audio-score synchronization methods. However, it is computationally inexpensive, and thus can quickly provide a rough synchronization, in which individual instrumental part contributions are automatically marked over time. Consequently, interesting follow-up approaches could be devised, in which cross- or multi- modal approaches might lead to stronger solutions, as already argued in [3, 10].

For the problem of score synchronization, a logical next step is to combine our analysis with typical audio-score synchronization approaches, or approaches generally relying on multiple synchronization methods, such as [5], to investigate whether a combination of methods improves the precision and efficiency of the synchronization procedure. Our added visual information layer can further be useful for e.g. devising structural performance characteristics, e.g. the occurrence of repeats. Our general synchronization results will also be useful for source separation procedures, since the obtained P/NP annotations indicate active sound-producing sources over time. Furthermore, results of our method can serve applications focusing on studying and learning about musical performances. We can easily output an activity map or multidimensional time-scrolling bar, visualizing which orchestra parts are active over time in a performance. Information about expected musical activity across sections can also help directing the focus of an audience member towards dedicated players or the full ensemble.

Finally, it will be interesting to investigate points where P/NP information in the visual and score channel clearly disagree. For example, in Figure 3, some time after the flutist starts playing, there is a moment where the score indicates a non-playing interval, while the flutist keeps a playing pose. We hypothesize that this indicates that, while a (long) rest is notated, the musical discourse actually still continues. While this also will need further investigation, this opens up new possibilities for research in performance analysis and musical phrasing, broadening the potential impact of this work even further.

Acknowledgements The research leading to these results has received funding from the European Union Seventh Framework Programme FP7 / 2007–2013 through the PHENICX project under Grant Agreement no. 601166.

9. REFERENCES

[1] A. D'Aguanno and G. Vercellesi. Automatic Music Synchronization Using Partial Score Representation Based on IEEE 1599. *Journal of Multimedia*, 4(1), 2009.

[2] R.B. Dannenberg and C. Raphael. Music Score Alignment and Computer Accompaniment. *Communications of the ACM*, 49(8):38–43, 2006.

[3] S. Essid and G. Richard. Fusion of Multimodal Information in Music Content Analysis. *Multimodal Music Processing*, 3:37–52, 2012.

[4] S. Ewert and M. Muller. Using Score-informed Constraints for NMF-based Source Separation. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 129–132. IEEE, 2012.

[5] S. Ewert, M. Müller, and R.B. Dannenberg. Towards Reliable Partial Music Alignments Using Multiple Synchronization Strategies. In *Adaptive Multimedia Retrieval. Understanding Media and Adapting to the User*, pages 35–48. Springer, 2011.

[6] C. Fremerey, M. Clausen, S. Ewert, and M. Müller. Sheet Music-Audio Identification. In *ISMIR*, pages 645–650, 2009.

[7] C. Fremerey, M. Müller, and M. Clausen. Towards Bridging the Gap between Sheet Music and Audio. *Knowledge Representation for Intelligent Music Processing*, (09051), 2009.

[8] M. Grachten, M. Gasser, A. Arzt, and G. Widmer. Automatic Alignment of Music Performances with Structural Differences. In *ISMIR*, pages 607–612, 2013.

[9] Y. Han and C. Raphael. Informed Source Separation of Orchestra and Soloist Using Masking and Unmasking. In *ISCA-SAPA Tutorial and Research Workshop, Makuhari, Japan*, 2010.

[10] C.C.S. Liem, M. Müller, D. Eck, G. Tzanetakis, and A. Hanjalic. The Need for Music Information Retrieval with User-centered and Multimodal Strategies. In *Proceedings of the 1st international ACM workshop MIRUM*, pages 1–6. ACM, 2011.

[11] M. Müller, F. Kurth, and T. Röder. Towards an Efficient Algorithm for Automatic Score-to-Audio Synchronization. In *ISMIR*, 2004.

[12] G.A. Ten Holt, M.J.T. Reinders, and E.A. Hendriks. Multi-dimensional Dynamic Time Warping for Gesture Recognition. In *13th annual conference of the Advanced School for Computing and Imaging*, volume 119, 2007.

[13] R.J. Turetsky and D.P.W. Ellis. Ground-truth Transcriptions of Real Music from Force-aligned MIDI Syntheses. *ISMIR 2003*, pages 135–141, 2003.

[14] B. Yao, J. Ma, and L. Fei-Fei. Discovering Object Functionality. In *ICCV*, pages 2512–2519, 2013.

MULTI-STRATEGY SEGMENTATION OF MELODIES

Marcelo Rodríguez-López
Utrecht University
m.e.rodriguezlopez@uu.nl

Anja Volk
Utrecht University
a.volk@uu.nl

Dimitrios Bountouridis
Utrecht University
d.bountouridis@uu.nl

ABSTRACT

Melodic segmentation is a fundamental yet unsolved problem in automatic music processing. At present most melody segmentation models rely on a 'single strategy' (i.e. they model a single perceptual segmentation cue). However, cognitive studies suggest that multiple cues need to be considered. In this paper we thus propose and evaluate a 'multi-strategy' system to automatically segment symbolically encoded melodies. Our system combines the contribution of different single strategy boundary detection models. First, it assesses the perceptual relevance of a given boundary detection model for a given input melody; then it uses the boundaries predicted by relevant detection models to search for the most plausible segmentation of the melody. We use our system to automatically segment a corpus of instrumental and vocal folk melodies. We compare the predictions to human annotated segments, and to state of the art segmentation methods. Our results show that our system outperforms the state-of-the-art in the instrumental set.

1. INTRODUCTION

In Music Information Retrieval (MIR), segmentation refers to the task of dividing a musical fragment or a complete piece into smaller cognitively-relevant units (such as notes, motifs, phrases, or sections). Identifying musical segments aids (and in some cases enables) many tasks in MIR, such as searching and browsing large music collections, or visualising and summarising music. In MIR there are three main tasks associated with music segmentation: (1) the segmentation of musical audio recordings into notes, as part of transcription systems, (2) the segmentation of symbolic encodings of music into phrases, and (3) the segmentation of both musical audio recordings and symbolic encodings into sections. In this paper we focus on the second task, i.e. identifying segments resembling the musicological concept of *phrase*. Currently automatic segmentation of music into phrases deals mainly with monophony. Thus, this area is commonly referred to as *melody segmentation*.

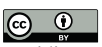
When targeting melodies, segmentation is usually reduced to identifying *segment boundaries*, i.e. locate the points in time where one segment transitions into another. [1] Computer models of melody segmentation often focus on modelling *boundary cues*, i.e. the musical factors that have been observed or hypothesised to trigger human perception of boundaries. Two common examples of boundary cues are: (a) the perception of 'gaps' in a melody (e.g. the sensation of a 'temporal gap' due to long note durations or rests) and (b) the perception of repetitions (e.g. recognising a melodic figure as a modified instance of a previously heard figure). The first cue mentioned is thought to signal the *end* of phrases, and conversely the second one is thought to signal the *start* of phrases.

Findings in melodic segment perception studies suggest that, even in short melodic excerpts, listeners are able to identify multiple cues, and what is more, that the role and relative importance of these cues seems to be contextual [3,6]. Yet, most computer models of melody segmentation rely on a *single strategy*, meaning that they often focus on modelling a single type of cue. For instance, [4] focuses on modelling cues related only to melodic gaps, while [1, 5] aim to modelling cues related only to melodic repetitions.

In this paper we propose and evaluate a *multi-strategy system* that combines single strategy models of melodic segmentation. In brief, our system first estimates the cues (and hence the single strategy models) that might be more 'relevant' for the segmentation of a particular input melody, combines the boundaries predicted by the models estimated relevant, and then selects which boundaries result in the 'most plausible' segmentation of the input melody.

Contribution: first, we bring together single strategy models that have not been previously tested in combination; second, our evaluation results show that our system outperforms the state-of-the-art of melody segmentation in instrumental folk songs.

The remainder of this paper is organised as follows: §2 reviews music segmentation related work using multi-strategy approaches, §3 presents a theoretical overview of the proposed system, §4 describes implementation details of the system, §5 describes and discusses our evaluation of the system, and finally, §6 provides conclusions and outlines possibilities of future work.



[1] Other subtasks associated to segmentation such as boundary pairing, as well as labelling of segments, are not considered.

2. RELATED WORK

Melody segmentation models often focus on modelling a single cue (e.g. [1, 4, 5]), leaving only a handful of models that have proposed ways to combine different cues. Perhaps the best known multi-strategy model is Grouper [11], which relies on three cues: temporal gaps, metrical parallelism, and segment length. Grouper employs temporal gap detection heuristics to infer a set of candidate boundaries, and uses dynamic programming to find an 'optimal' segmentation given the candidate boundaries and two regularisation constraints (metrical parallelism and segment length). Grouper constitutes the current state-of-the-art in melodic segmentation. However, Grouper relies entirely on temporal information, and as such might have difficulties segmenting melodies with low rhythmic contrast or no discernible metric.

Another multi-strategy model is ATTA [7], which merges gap, metrical, and self-similarity related cues. In ATTA the relative importance of each cue is assigned manually, requiring the tuning of over 25 parameters. Parameter tuning in ATTA is time consuming (estimated to be ~10 mins per melody in [7]). Moreover, the parameters are non-adaptive (set at initialization), and thus make the model potentially insensitive to changes in the relative importance of a given cue during the course of a melody.

The main differences between the research discussed and ours are: (a) our system integrates single strategy models that have not been previously used (and systematically tested) in combination, and (b) our system provides ways to select which single strategy models to use for a particular melody. In §5.3.2 we compare our system to the two models that have consistently performed best in comparative studies, namely Grouper [11] and LBDM [4]. [2]

3. THEORETICAL OVERVIEW OF OUR SYSTEM

In this section we describe our system, depicted in Figure 1. In module 1, our system takes a group of single strategy segmentation models (henceforth 'cue models'), selects which might be more relevant to segment the current input melody, and combines the estimated boundary locations into a single list. In module 2, the system assesses the segmentation produced by combinations of the selected boundary candidates in respect to corpus-learnt priors on segment contour and segment length. Below we describe in more detail the input/output characteristics of our system, as well as each processing module.

3.1 Input/Output

The input to our system consists of a melody and a set of boundaries predicted by cue models. The melody is encoded as a sequence of temporally ordered note events $e = e_1, \dots e_i, \dots, e_n$. In e each note event is represented by its chromatic pitch and quantized duration (onset, offset) values. The output of our system is a set of 'optimum' boundary locations b_{opt} of length m, constituting

[2] The manual tuning feature of ATTA made it impossible to include it in our evaluation.

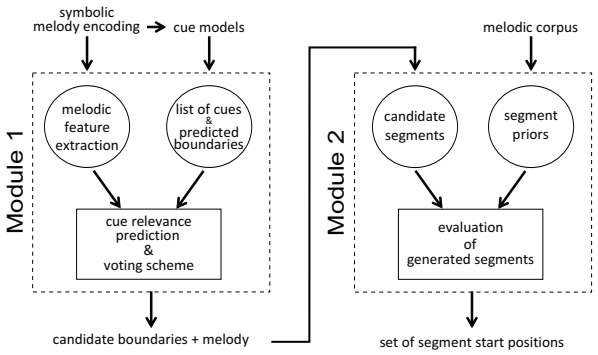


Figure 1. General diagram of our system. Within the modules ○ = input elements, and □ = processing stages.

a set of segments $S_{opt} = \{s_i\}_{1 \leq i < m}$, where each segment $s_i = [b_i, b_{i+1})$.

3.1.1 Cue Models Characteristics

Each cue model transforms e into a set of sequences, each representing a melodic attribute (e.g. pitch class, inter-onset-interval, etc.). The specific set of attribute sequences produced by each cue model used within our system is discussed in §4.2. Each cue model processes the attribute sequences linearly, moving in steps of one event, producing a *boundary strength profile*. A boundary strength profile is simply a normalized vector of length n, where each element value encodes the strength with which a cue model 'perceives' a boundary at the temporal location of the element. In these profiles segment boundaries correspond to local maxima, and thus candidate boundary locations are obtained via peak selection. The method used to select peaks is discussed in §4.2.

3.2 Module 1: Multiple-Cue Boundary Detection

Module 1 takes as input a set of features describing the melody, and a set of boundary locations predicted by cue models. Module 1 is comprised of two processing stages, namely 'cue relevance prediction' and 'voting scheme'. The first uses the input melodic features to estimate the 'relevance' of a given cue for the perception of boundaries in the input melody, and the second merges and filters the predicted boundary locations.

3.2.1 Cue Relevance Prediction

For a given set of k cue models $C = \{c_i\}_{1 \leq i \leq k}$, and a set of h features describing the melodies $F = \{f_j\}_{1 \leq j \leq h}$, we need to estimate how well a given cue model might perform under a given performance measure M as $P(M|C_i, F_j)$. In this paper we use the common F1, $precision$, and $recall$ measures to evaluate performance (see §5.2). In module 1 we focus on predicting a cue model's $precision$ (assuming high $recall$ can be achieved by the combined set of candidate boundaries).

3.2.2 Voting Scheme

Once we have estimated the relevance value of each cue model for the input melody $P(M|C_i, F_j)$, we combine the

candidate boundaries by simply adding the relevance values of candidate boundaries in close proximity (i.e. ± 1 note event apart). We assume that if boundaries from different cues are located ± 1 note event apart, one of them might be identifying a beginning and the other an end of segment, and thus for the processing in module 2 is beneficial to keep both.

The final output of this module is a single list of boundary locations b, each boundary with its own relevance value.

3.3 Module 2: Optimality-Based Segment Formation

Module 2 takes as input b, e, and length/contour [3] priors computed from a melodic corpus. The task of this module is to find the 'most plausible' set of segments S_{opt} from the space of all possible candidate segmentations. The idea is to evaluate segmentations according to two empirical constraints: one, melodic segments tend to show small deviations from a 'typical' segment length, and two, melodic segments tend to show a reduced set of prototypical melodic contour shapes. We address the task of finding the most plausible set of segments given these two constraints as an optimisation problem. Thus, for a given candidate segmentation $S_c = \{s_i\}_{1 \leq i < t}$, derived from a subset of t candidate boundaries $c \in b$, where $s_i = [c_i, c_{i+1})$, our cost function is defined as:

$$C(S_c) = \sum_{i=1}^{t-1} T(s_i) \tag{1}$$

with

$$T(s_i) = \Phi(s_i) + \alpha(\Upsilon(s_i) + \Psi(s_i)) \tag{2}$$

Where,

- $\Phi(s_i)$ is the cost associated to each candidate boundary demarcating s_i (i.e. the inverse of the relevance value of each candidate boundary).
- $\Upsilon(s_i)$ is a cost associated to the deviation of s_k from an expected phrase contour. The cost of $\Upsilon(s_i)$ is computed as $-log(\cdot)$ of the probability of the contour of the candidate phrase segment s_i.
- $\Psi(s_i)$ is a cost of the deviation from the length of s_i from an expected length. The cost of $\Psi(s_i)$ is computed as $-log(\cdot)$ of the probability of the length of the candidate phrase segment s_i.
- α is a user defined parameter that balances the boundary related costs against the segment related costs.

Details for the computation of S_{opt} and priors on segment length/contour are given in §4.4.

[3] Melodic contour can be seen as an overall temporal development of pitch height

4. SYSTEM IMPLEMENTATION

In this section we first describe the selection and tuning of the cue models used within our system, then provide some details on the implementation of modules 1 and 2.

4.1 Cue Models: Selection

We selected and implemented four cue models based on two conditions: (a) the models have shown relatively high performance in previous studies, (b) the cues modelled have been identified as being important for melody segmentation within music cognition studies. All implemented models follow the same processing chain, described in §3.1, i.e. each model derives a set of melodic attribute sequences, processes each sequence linearly, and outputs a boundary strength profile bsp. Below we list and briefly describe the cue models used within our system.

CM1 - gap detection: Melodic gap cues are assumed to correspond to points of significant local change, e.g. a pitch or duration interval that is perceived as 'overly large' in respect to its immediate vicinity. We implemented a model of melodic gap detection based on [4]. The model uses a distance metric to measure local change, [4] and generates a bsp where peaks correspond to large distances between contiguous melodic events. Large local distances are taken as boundary candidates.

CM2 - contrast detection: Melodic contrast cues are assumed to correspond to points of significant change (which require a mid-to-large temporal scale to be perceptually discernible), e.g. a change in melodic pace, or a change of mode. We implemented a contrast detection model based on [9]. The model employs a probabilistic representation of melodic attributes and uses an information-theoretic divergence measure to determine contrast. The model generates a bsp where peaks correspond to large divergences between attribute distributions representing contiguous sections of the melody. The model identifies boundaries by recursively locating points of maximal divergence.

CM3 - repetition detection: Melodic repetition cues are assumed to correspond to salient (exact or approximate) repetitions of melodic material. We implemented a model to locate salient repeated fragments of a melody based on [5]. The model uses an exact-match string pattern search algorithm to extract repeated melodic fragments, and includes a method to score the salience of repetitions based on the length, frequency, and temporal overlap of the extracted fragments. The model generates a bsp where peaks correspond to the starting points of salient repetitions.

CM4 - closure detection: Tonal closure cues are assumed to correspond to points where an ongoing cognitive process of melodic expectation is disrupted. One way in which expectation of continuation might be disrupted is when a melodic event following a given context is unexpected. We implemented an unexpected-event detection model based on [8]. [5] The model employs unsupervised probabilistic

[4] The model employs both pitch and temporal information, but in our tests only temporal information is used

[5] Our implementation is however less sophisticated than that of [8], as it requires the user to provide an upper limit for context length (specifed

learning and prediction to measure the degree of unexpectedness of each note event in the input melody, given a finite preceding context. The model generates a bsp where peaks correspond to significant increases in (information-theoretic) surprise. Candidate boundaries are placed before surprising note events.

4.2 Cue Models: Tuning

We tuned the cue models used within our system to achieve maximal precision. This involved a selection of melody representation (choice of melodic attribute sequences to be processed), [6] tuning of parameters exclusive to the cue model, and choice and tuning of a peak selection mechanism.

The choice of attribute sequence selection and parameter tuning per cue model is listed in Table 1. The abbreviations of melodic attributes correspond to: `cp`: chromatic pitch, `ioi`: inter onset interval, `ooi`: onset to offset interval, `cpiv`: chromatic pitch interval, `pcls`: pitch class. To select peaks as boundary candidates, we experimented with several peak selection algorithms, settling for the algorithm proposed in [8]. [7] This peak selection algorithm has only one parameter k. The optimal values of k for each cue model are given in the rightmost column of Table 1. We also provide details on the choice of parameters exclusive to each cue model, for an elaboration on their interpretation we refer the reader to the original publications.

Cue model	attribute sequence set	parameters
CM1	{`cpiv`, `ioi`, `ooi`}	$k = 2$
CM2	{`pcls`, `ioi`}	-
CM3	{`cp`, `ioi`}	$F = 3$
		$L = 3$
		$O = 1$
		$k = 3$
CM4	{`cp`, `pcls`, `cpiv`}	PPM-C, with exclusion
		STM: order 5
		LTM: order 2
		LTM: 400 EFSC melodies
		$k = 2.5$

Table 1. Attributes and parameter settings of cue models.

4.3 Module 1: Predictors and Feature Selection

To evaluate cue relevance prediction, we first select a subset of 200 boundary annotated melodies from the melodic corpora used in this paper (see §5.1), and then run the cue models to obtain precision performance values for each melody. To allow an estimation of precision we partition its range into a discrete set of categories. [8]

To determine cue relevance prediction, we experimented with several off-the-shelf classifiers available as part of *Weka* [9] . We selected features using the common *BestFirst* with a 10-fold cross validation. The selected features were those used in all folds.

The melodic features used to predict precision by the classifiers where taken from the *Fantastic* [10] and *jSymbolic* [11] feature extractor libraries, which add up to over 200. After selection, 17 features are kept: 'melody length', 'pitch standard deviation, skewness, kurtosis, and entropy', 'pitch interval standard deviation, skewness, kurtosis, and entropy', 'duration standard deviation, skewness, kurtosis, and entropy', 'tonal clarity', 'm-type mean entropy', 'm-type Simpson's D', 'm-type productivity' (please refer to the Fantastic library documentation for definitions).

The classifiers we experimented with are Sequential Minimal Optimization (*SMO*, with the radial basis function kernel), K-Nearest Neighbours (*K**) and Bayesian Network (*BNet*). To evaluate each classifier we use 10-fold cross validation. The classifier with the best performance-to-efficiency ratio is *SMO* for models `CM2-CM4`, with an average accuracy of 72.21%, and the simple *K** for `CM1` with an average accuracy of 66.37%.

4.4 Module 2: Computation of Priors and Choice of α

To compute the optimal sequence of segments S_{opt} we minimise the cost function in Eq. 1 using a formulation of the Viterbi algorithm based on [10]. The minimisation of Eq. 1 is subject to constraints on segment contour and segment length, and to a choice for parameter α. We tuned α manually (a value of 0.6 worked best in our experiments). To model constraints in segment contour and segment length we use probability priors. Below we provide details on their computation.

A prior $P(contour(s_k))$ is computed employing a Gaussian Mixture Model (GMM). Phrase contours are computed using the polynomial contour feature of the Fantastic library. A contour model with four nodes was selected. The GMM (one Gaussian per node) is fitted to contour distributions obtained from a subset of 1000 phrases selected randomly from the boundary annotated corpora used in this paper (see §5.1).

A prior of segment length $P(l_k)$ is computed employing a Gaussian fitted to a distribution of lengths obtained from the same 1000 phrase subset used to derive contours.

5. EVALUATION

In this section we describe our test database and evaluation metrics, and subsequently describe experiments and results obtained by our system. A prototype of our system was implemented using a combination of Matlab, R, and Python. Source files and test data are available upon request.

as the Markov order in Table 1).

[6] While some cue models, e.g. [4, 11] have already a preferred choice of melodic attribute representation, the other cue models used within our system allow for many choices, and where thus selected through experimental exploration.

[7] This algorithm proved to work better than the alternatives for all models but CM3, for which its own peak selection heuristic worked best.

[8] In our experiments we used a set dividing a model's precision into two categories (1:*poor*, 2:*good*). The exact mapping $precision : [0, 1] \rightarrow \{1, 2\}$ was selected manually for each cue model, to ensure a sufficient number of melodies representing each performance category is available for training.

[9] `http://www.cs.waikato.ac.nz/ml/weka/`

[10] `http://www.doc.gold.ac.uk/~mas03dm/`

[11] `http://jmir.sourceforge.net/jSymbolic.html`

5.1 Melodic Corpora

To evaluate our system we employed a set of 100 instrumental folk songs randomly sampled from the Liederenbank collection [12] (LC) and 100 vocal folk songs randomly sampled from the German subset of the Essen Folk Song Collection [13] (EFSC). We chose to use the EFSC due to its benchmark status in the field of melodic segmentation. Additionally, we chose to use the LC to compare the performance of segmentation models in vocal and non-vocal melodies. [14]

The EFSC consists of ∼6000 songs, mostly of German origin. The EFSC data was compiled and encoded from notated sources. The songs are available in EsAC and **kern formats. The origin of phrase boundary markings in the EFSC has not been explicitly documented (yet it is commonly assumed markings coincide with breath marks or phrase boundaries in the lyrics of the songs).

The instrumental (mainly fiddle) subset of the LC consists of ∼2500 songs. The songs were compiled and encoded from notated sources. The songs are available in MIDI and **kern formats. Segment boundary markings for this subset comprise two levels: 'hard' and 'soft'. Hard (section) boundary markings correspond with structural marks found in the notated sources. Soft (phrase) boundary markings correspond to the musical intuition of two experts annotators. [15]

5.2 Evaluation Measures

To evaluate segmentation results, we encode both predicted and human-annotated phrase boundary markings as binary vectors. Using these vectors we compute the number of true positives tp (hits), false positives fp (insertions), and false negatives fn (misses). [16] We then quantify the similarity between predictions and human annotations using the well known $F1 = \frac{2 \cdot p \cdot r}{p+r}$, where precision $p = \frac{tp}{tp+fp}$ and recall $r = \frac{tp}{tp+fn}$. While the $F1$ has its downsides (it assumes independence between boundaries), [17] it has been used extensively in the field and thus allows us to establish a comparison to previous research.

5.3 Experiments & Results

In our experiments we compare our system to the melody segmentation models that have consistently scored best in comparative studies: GROUPER [11] and LBDM [4]. The first is a multi-strategy model, and the second a single strategy (gap detection) model. [18] We also compare our system to its performance when only one module is active. Additionally we compare to two naïve baselines: *always*, which predicts a segment boundary at every melodic event position, and *never* which does not make predictions.

Table 2 shows the performance results of all models over the instrumental and vocal melodic sets. We refer to our model as COMPLETE, and to the configurations when either module 1 or 2 are active as MOD1ON and MOD2ON, respectively.

We tested the statistical significance of the paired F1 differences between the three configurations of our system, the two state-of-the-art models, and the baselines. For the statistical testing we used a non-parametric Friedman test ($\alpha = 0.05$). Furthermore, to determine which pairs of measurements significantly differ, we conducted a post-hoc Tukey HSD test. All pair-wise differences among configurations were found to be statistically significant, except those between MOD1ON and MOD2ON in the vocal set and between LBDM and MOD2ON in the instrumental set. In Table 2 the highest performances are highlighted in bold.

Database	Instrumental			Vocal		
Model	$\overline{R}$	$\overline{P}$	$\overline{F1}$	$\overline{R}$	$\overline{P}$	$\overline{F1}$
COMPLETE	0.56	0.62	**0.54**	0.49	0.67	0.56
GROUPER	0.81	0.31	0.44	0.60	0.62	**0.61**
LBDM	0.57	0.49	0.45	0.56	0.55	0.52
MOD2ON	0.51	0.49	0.44	0.48	0.45	0.47
MOD1ON	0.52	0.47	0.42	0.63	0.42	0.46
always	0.06	1.00	0.09	0.08	1.00	0.12
never	0.00	0.00	0.00	0.00	0.00	0.00

Table 2. Performance of models and baselines sorted in order of mean recall $\overline{R}$, precision $\overline{P}$, and $\overline{F1}$ for instrumental and vocal melodies. The results presented in this table were obtained comparing predictions to the 'soft' boundary markings of the LC.

5.3.1 Summary of Main Results

In general, F1 performances obtained by the segmentation models in the vocal set are consistently higher than in the instrumental set. This might be simply an indication that in the instrumental set melodies constitute a more challenging evaluation scenario. However, the F1 differences might also be an indication that relevant perceptual boundary cues are not covered by the evaluated models.

In the instrumental set, COMPLETE outperforms both LBDM and GROUPER by a relatively large margin ($\geq 10\%$). In the vocal set, on the other hand, GROUPER obtains the best performance. Below we discuss the three configurations of our system (COMPLETE, MOD1ON, MOD2ON).

5.3.2 Discussion

In both melodic sets MOD1ON shows considerably higher recall than precision. These recall/precision differences agree with intuition, since the output of MOD1ON consists of the combination of all boundaries predicted by

[12] http://www.liederenbank.nl/

[13] http://www.esac-data.org

[14] Vocal music has dominated previous evaluations of melodic segmentation (especially large-scale evaluations), which might give an incomplete picture of the overall performance and generalisation capacity of segmentation models

[15] Instructions to annotate boundaries were related to performance practice (e.g. "where would you change movement of bow").

[16] The first and last boundaries are treated as trivial cases which correspond, respectively, to the beginning and ending notes of a melodic phrase. These trivial cases are excluded from the evaluation. Also, we allow a tolerance of ± 1 note event for the computation of tp.

[17] By assuming independence between boundaries aspects such as segment length and position are discarded from the evaluation

[18] For our tests we ran GROUPER and LBDM with their default settings.

the cue models, and can hence be expected to contain a relatively large number of false positives. On the other hand, MOD2ON shows smaller differences between precision and recall values, and shows higher F1 performances than MOD1ON in both melodic sets (although the difference between performances is significant only for the instrumental set). This last result highlights the robustness of the optimisation procedure driving MOD2ON. [19]

The large F1 differences between MOD1ON and MOD2ON in respect to COMPLETE suggest that segmentation at the phrase level is a perceptual process which, despite happening in 'real time' (i.e. as music unfolds itself, represented more closely by module 1), might still require repeated exposure and retrospective listening (represented more closely by module 2).

Manual examination COMPLETE reveals that, when segmenting the vocal melody set, the prediction stage of module 1 tends to overestimate the importance of cue models (i.e. it often misclassifies models as relevant when they are not). However, when altering the settings of COMPLETE so that the prediction stage of model 1 is more conservative (i.e. so that it predicts fewer boundaries), there is no significant improvement in performance. Closer analysis of these results points to a trade-off in performance, i.e. while a conservative setting increases precision (predictions have fewer 'false positives'), it also decreases recall (predictions have fewer 'correct positives'). This suggests that the prediction stage of module 1 might require estimation of cue relevance at a local level, i.e. on subsections of the melody rather than on the whole melody.

6. CONCLUSION

In this paper we introduce a multi-strategy system for the segmentation of symbolically encoded melodies. Our system combines the contribution of single strategy models of melody segmentation. The system works in two stages. First, it estimates how relevant the boundaries computed by each selected single strategy model are to the melody being analysed, and then combines boundary predictions using heuristics. Second, it assesses the segmentation produced by combinations of the selected boundary candidates in respect to corpus-learnt priors on segment contour and segment length.

We tested our system on 100 vocal and 100 instrumental folk song melodies. The performance of our system showed a considerable (10% $F1$) improvement upon the state-of-the-art in melody segmentation for instrumental folk music, and showed to perform second best in the case of vocal folk songs.

In future work we will test if the relevance of cue models can be accurately estimated for sections of the melody (and not the whole melody as it is done in this paper). This 'local' account of relevance might play a major role in improving the system's precision. Also, we will incorporate a more advanced model of prior segment knowledge of segment structure in our system. We hypothesise that a model of the characteristics of [2] could constitute a good alternative to model not only segment length and contour, but also to incorporate knowledge of 'template' phrase structure forms. Lastly, we will continue testing our model's generalisation capacity by evaluating on larger sample sizes and genres other than folk (for the latter the authors are currently in the process of annotating a corpus of Jazz melodies).

Acknowledgments: We thank Frans Wiering, Remco Veltkamp, and the anonymous reviewers for the useful comments on earlier drafts of this document. Marcelo Rodríguez-López and Anja Volk (NWO-VIDI grant 276-35-001) and Dimitrios Bountouridis (NWO-CATCH project 640.005.004) are supported by the Netherlands Organization for Scientific Research.

[19] If we consider that (with MOD1ON bypassed) the number of candidate boundaries taken as input to MOD2ON often exceeds 'correct' (human annotated) boundaries by a factor 2 or 3, then the number of possible segmentations of the melody shows an exponential increase, leading to local minima issues, and so it would be reasonable to expect a performance equal or worse than that of MOD1ON.

7. REFERENCES

[1] S. Ahlbäck. Melodic similarity as a determinant of melody structure. *Musicae Scientiae*, 11(1):235–280, 2007.

[2] R. Bod. Probabilistic grammars for music. In *Belgian-Dutch Conference on Artificial Intelligence (BNAIC)*, 2001.

[3] M. Bruderer, M. McKinney, and A. Kohlrausch. The perception of structural boundaries in melody lines of western popular music. *Musicae Scientiae*, 13(2):273–313, 2009.

[4] E. Cambouropoulos. The local boundary detection model (LBDM) and its application in the study of expressive timing. In *Proceedings of the International Computer Music Conference (ICMC01)*, pages 232–235, 2001.

[5] E. Cambouropoulos. Musical parallelism and melodic segmentation. *Music Perception*, 23(3):249–268, 2006.

[6] E. Clarke and C. Krumhansl. Perceiving musical time. *Music Perception*, pages 213–251, 1990.

[7] M. Hamanaka, K. Hirata, and S. Tojo. ATTA: Automatic time-span tree analyzer based on extended GTTM. In *ISMIR Proceedings*, pages 358–365, 2005.

[8] M. Pearce, D. Müllensiefen, and G. Wiggins. The role of expectation and probabilistic learning in auditory boundary perception: A model comparison. *Perception*, 39(10):1365, 2010.

[9] M. Rodríguez-López and A. Volk. Melodic segmentation using the jensen-shannon divergence. In *International Conference on Machine Learning and Applications (ICMLA12)*, volume 2, pages 351–356, 2012.

[10] G. Sargent, F. Bimbot, E. Vincent, et al. A regularity-constrained Viterbi algorithm and its application to the structural segmentation of songs. In *ISMIR Proceedings*, 2011.

[11] D. Temperley. *The cognition of basic musical structures*. MIT Press, 2004.

A DATA SET FOR COMPUTATIONAL STUDIES OF SCHENKERIAN ANALYSIS

Phillip B. Kirlin
Department of Mathematics and Computer Science, Rhodes College
kirlinp@rhodes.edu

ABSTRACT

Schenkerian analysis, a kind of hierarchical music analysis, is widely used by music theorists. Though it is part of the standard repertoire of analytical techniques, computational studies of Schenkerian analysis have been hindered by the lack of available data sets containing both musical compositions and ground-truth analyses of those compositions. Without such data sets, it is difficult to empirically study the patterns that arise in analyses or rigorously evaluate the performance of intelligent systems for this kind of analysis. To combat this, we introduce the first publicly available large-scale data set of computer-processable Schenkerian analyses. We discuss the choice of musical selections in the data set, the encoding of the music and the corresponding ground-truth analyses, and the possible uses of these data. As an example of the utility of the data set, we present an algorithm that transforms the Schenkerian analyses into hierarchically-organized data structures that are easily manipulated in software.

1. CORPUS-DRIVEN RESEARCH

Corpus-driven research is now commonplace in the music informatics community. With the wealth of raw musical information now available in digital form, in many cases, it is straightforward to construct and use data sets containing numerous musical compositions. However, the problem of collecting ground-truth metadata about the content of the music still exists, especially where high-level features are concerned. This is a problem that effects researchers working with music in audio or symbolic formats.

Ground-truth data sets that include features specifically relating to music theory or music analysis are particularly labor-intensive to construct. Information about the high-level harmonic or melodic structure of compositions is often only found scattered throughout textbooks or individual research publications, and so there are few publicly-available corpora containing such information in a computer-processable format. Some data sets are created only for specific research projects and then discarded, are not in an easy-to-use format, or are simply never made widely available.

The lack of varied ground-truth musical metadata relating to theory and analysis — especially data sets specifically designed to align with symbolic music data — hinders corpus-driven research studies because time must be spent collecting data. Sometimes the researchers must perform the music analysis themselves, possibly inadvertently introducing biases into the data. Without widely available comprehensive data sets, it is extremely difficult to conduct large-scale experiments on the structure of musical compositions in symbolic form, or quantitatively evaluate the performance of computational systems that emulate a music analysis process.

There is a particular dearth of empirical data available in the realm of *Schenkerian analysis*, a widely used analytical system that illustrates a hierarchical structure among the notes of a composition. Though Schenkerian analysis is one of the most comprehensive methods for music analysis that we have available today [1], there are no large-scale digital repositories of analyses available to researchers. In addition to the reasons stated above for the lack of corpora, Schenkerian analysis presents a number of unique challenges to creating a useful data set. First, a Schenkerian analysis for a composition is illustrated using the musical score of the composition itself, and commonly requires multiple staves to show the hierarchical structure uncovered. This requires substantial space on the printed page and thus is a deterrent to retaining large sets of analyses. Second, there is no established computer-interpretable format for Schenkerian analysis storage, and third, even if there were a format, it would take a great deal of effort to encode a number of analyses into processable computer files.

The lack of data has kept the number of computational studies of Schenkerian analysis requiring ground-truth data to a bare minimum; some examples include studies using corpora with six [7] or eight [6] pieces. Though these studies are useful, the results would likely carry more weight if the data sets used were larger.

With all of these ideas in mind, in this paper we introduce the first large-scale data set of musical compositions along with corresponding ground-truth Schenkerian analyses, called SCHENKER41 [1] . The 41 musical selections included constitute the largest-known corpus of Schenkerian analyses in a machine-readable format. The musical selec-

[1] Available at www.cs.rhodes.edu/~kirlinp/schenker41.

tions are standardized in mode, length, and instrumentation, and the analyses are stored in a novel text-based representation designed to be easily processed by a computer. We created these data with the hope that they would be useful to researchers (a) studying the Schenkerian analysis process itself from a quantitative standpoint (for instance, detecting patterns in the way analysis is done), (b) needing a data set of analyses for use with supervised machine learning techniques, and (c) performing any sort of quantitative evaluation requiring ground-truth hierarchical music analyses.

2. THE DATA SET

2.1 Creation and Content

In order to create a data set of musical compositions and corresponding ground-truth Schenkerian analyses that would be useful to researchers with a wide variety of goals, we restricted ourselves to music from the common practice period of European art music, and selected 41 excerpts from works by J. S. Bach, G. F. Handel, Joseph Haydn, M. Clementi, W. A. Mozart, L. van Beethoven, F. Schubert, and F. Chopin. All of the compositions were either for a solo keyboard instrument (or arranged for such an instrument) or for voice with keyboard accompaniment. All were in major keys and did not modulate.

The musical excerpts were also selected for the ease of locating a Schenkerian analysis for each excerpt done by an outside expert. Analyses for the 41 excerpts chosen came from four places: Forte and Gilbert's textbook *Introduction to Schenkerian Analysis* [4] and the corresponding instructor's manual [3], Cadwallader and Gagné's textbook *Analysis of Tonal Music* [2], Pankhurst's handbook *SchenkerGUIDE* [9], and a professor of music theory who teaches a Schenkerian analysis class. These four sources are denoted by the labels F&G, C&G, SG, and Expert in Table 1, which lists the excerpts in the corpus.

From a Schenkerian standpoint, we also chose excerpts such that the analyses of the excerpts would all share some commonalities. All the analyses contained a single linear progression as the fundamental background structure: either an instance of the *Ursatz* or a rising linear progression. Some excerpts contained an *Ursatz* with an *interruption*: a Schenkerian construct that occurs when a musical phrase ends with an incomplete instance of the *Ursatz*, then repeats with a complete version.

We put these restrictions on the musical content in place because we expected that if SCHENKER41 were to be used for supervised machine learning, such algorithms would be able to better model a corpus with less variability among the pieces.

Overall, SCHENKER41 contains 253 measures of music and 907 notes. The lengths of individual excerpts ranged from 6 to 53 notes.

2.2 Encoding

With our selected musical excerpts and our corresponding analyses in hand, we needed to translate the musical information into machine-readable form. Musical data has many established encoding schemes; we used MusicXML, a format that preserves more information from the original score than say, MIDI.

Translating the Schenkerian analyses proved harder because there is no current standard for storing such analyses in a format that a computer could easily process and manipulate. Therefore, we devised a text-based encoding scheme to represent the various notations found in a Schenkerian analysis. Each analysis is stored in a single text file that is linked to a specific MusicXML file containing the musical excerpt being analyzed.

Schenkerian analyses are primarily based on the concept of a *prolongation*, a situation where an analyst determines that a group of notes is elaborating a group of more structurally fundamental notes. Consider the descending melodic pattern D–C–B–F♯–G all occurring over G major harmony, as is shown in Figure 1. We could imagine that an analyst would determine that this passage outlines a descending G-major triad (D–B–G), with the second note C (a passing tone) serving to melodically connect the preceding D to the following B. We would say the note C *prolongs* the motion from D to B. Similarly, the F♯ prolongs the motion from B to G. Schenkerian analysis hypothesizes that any tonal composition is structured as a nested collection of prolongations; identifying them is a important component of the analysis procedure.

Every prolongation identified in an analysis is encoded in the analysis text file using the syntax $X\ (Y)\ Z$, where X and Z are individual notes in the score and Y is a non-empty list of notes. Such a statement means that the notes in Y prolong the motion from note X to note Z. Additionally, we permit incomplete prolongations in the text file representation: one of X or Z may be omitted. The notes of X, Y and Z are transcribed in the text file as is shown in Figure 1, with a measure number, followed by a pitch and octave, followed by a integer to distinguish between repeated notes in the same measure. Figure 2 shows how the prolongations of Figure 1 would be encoded. Note that the prolongation involving the F♯ is encoded with no X component; this tell us that there is no strong melodic connection from the B to the F♯, only from the F♯ to the G.

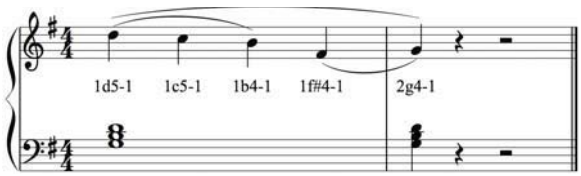


Figure 1. A melodic sequence with note names.

```
1d5-1 (1c5-1) 1b4-1
(1f#4-1) 2g4-1
1d5-1 (1b4-1) 2g4-1
```

Figure 2. An encoding of the prolongations present.

This text format easily supports encoding prolongations at differing hierarchical levels in the music. We can see

how Figure 2 encodes both the "surface-level" prolongations D–C–B and F♯–G, but also the deeper prolongation D–B–G which outlines the fifth relationship in the G-major chord.

Aside from prolongations, the encoding system supports describing repetitions of notes that may be omitted in the analysis on the printed page; any linear progressions, including instances of the *Ursatz*; and the harmonic context present at any point in the analysis.

2.3 Compromises

An additional challenge not previously mentioned in creating the SCHENKER41 analyses is choosing an appropriate level of detail of the material to encode. Because the main objects in analyses are prolongations, it is natural to attempt to group them into categories like "neighbor tone" and "passing tone." However, not all prolongations identified in analyses are easily categorized, and so category labels are often omitted in analyses not found in an educational context. This raises the question of whether or not to attempt to encode the category of prolongations in this corpus. To avoid the risk of incorrectly interpreting analyses, we have chosen to encode only what is *directly observable on the printed page* — the hierarchical relationship between groups of notes — and not categorize the prolongations found in the analyses. We recognize that this is a compromise between staying true to the data and encoding all potentially useful information.

3. USAGE OF THE DATA

The SCHENKER41 data set enables the undertaking of a wide variety of tasks and studies. In addition to the already-discussed endeavors of using the corpus for supervised machine learning or for quantitative evaluation, we theorize that with these data it could be possible to address the following questions:

- Do analysts identify certain types of prolongations more often than others under certain circumstances? These circumstances may involve the composer, musical genre, or even the analysis source.
- Does Schenkerian analysis align well with other forms of music analysis, such as Narmour's implication-realization model of melodic expectation [8]?
- How well do Schenkerian analyses align with expressive performances of the music [10]? Do features of a performance such as phrasing, volume, or other quantifiable measures of musicality correspond to various Schenkerian annotations in an analysis?

Besides answering questions about Schenkerian analysis itself, we hope that that the availability of SCHENKER41 will spur others to study the utility of Schenkerian analysis in other areas of music informatics. For instance, we suspect hierarchical analyses could prove useful in constructing musical similarity metrics, because Schenkerian analyses may highlight a common melodic pattern residing under the surface in two different musical excerpts.

Though the SCHENKER41 analyses can be directly processed by software, the nature of the flat text file format in which the data are encoded makes it difficult to see hierarchical relationships between notes not directly related by a single prolongation. Therefore, in this section we describe an algorithm to translate the analysis text files into hierarchical graph structures known as MOPs. It is possible to use the SCHENKER41 data in MOP form to automatically learn characteristics of Schenkerian analysis [5].

3.1 Maximal Outerplanar Graphs

Maximal outerplanar graphs, or *MOPs*, were first proposed by Yust [11] as elegant structures for representing a set of musical prolongations in a Schenkerian-style hierarchy. A MOP represents a hierarchy of melodic intervals located in a monophonic sequence of notes, though Yust proposed some extensions for polyphony. For example, the prolongations mentioned in Figures 1 and 2 are represented by the MOP shown in Figure 3.

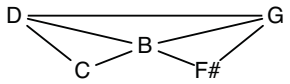


Figure 3. A MOP representation of the music in Figure 1.

Formally, a MOP is a complete triangulation of a polygon, where the vertices of the polygon are notes and the outer perimeter of the polygon consists of the melodic intervals between consecutive notes of the original music, except for the edge connecting the first note to the last, which is called the *root edge*. Each triangle in the polygon specifies a prolongation. For instance, in Figure 3, the presence of triangle D–C–B means that the melodic motion from D to B is prolonged by the C. By expressing the hierarchy in this fashion, each edge (x, y) carries the interpretation that notes x and y are "consecutive" at some level of abstraction of the music. Edges closer to the root edge express more abstract relationships than edges farther away.

Outerplanarity is a property of a graph that can be drawn such that all the vertices are on the perimeter of the graph. Such a condition is necessary for us to enforce the strict hierarchy among the prolongations. A *maximal* outerplanar graph cannot have any additional edges added to it without destroying the outerplanarity; such graphs are necessarily polygon triangulations, and under this interpretation, all prolongations must occur over triples of notes.

There are three representational issues with MOPs we must address before discussing the algorithm to convert an analysis text file into MOPs. First, Schenkerian analyses as commonly encountered often include prolongations involving more than three notes. The analysis sources used in SCHENKER41 are no exception. For this reason, we relax the "maximal" qualifier for MOPs and permit prolongations involving any number of notes in our MOP representation. A prolongation involving more than three notes

will be translated into a polygon with more than three edges in the MOP representation.

Second, MOPs do not have a direct way to represent a prolongation with only a single "parent" note. Because MOPs model prolongations as a way of moving *from* one musical event *to* another event, every prolongation must have two parent notes. Music sometimes presents situations, however, that an analyst would model with a one-parent prolongation, such as an incomplete neighbor tone (we encountered this situation in Figure 2). Yust interprets such prolongations as having a "missing" origin or goal note that has been elided with a nearby structural note, which substitutes in the MOP for the missing note.

The third representational issue stems from trying to represent prolongations involving the first or last note in the music. Prolongations necessarily take place over time, and in a MOP, we interpret the temporally middle notes as prolonging the motion from the earliest note (the left parent) to the latest (the right parent). Following this temporal logic, we can infer that the root edge of a MOP must therefore necessarily be between the first note of the music and the last, implying these are the two most structurally important notes of a composition. As this is not always true in compositions, we add two pseudo-events to every MOP: an initiation event that is located temporally before the first note of the music, and a termination event, which is temporally positioned after the last note. The root edge of a MOP is fixed to always connect the initiation event and the termination event. These extra events allow for any melodic interval — and therefore any pair of notes in the music — to be represented as the most structural event in the composition. For instance, in Figure 4, which shows the D–C–B–F♯–G pattern with initiation and termination events (labeled START and FINISH), the analyst has indicated that the G is the most structurally significant note in the passage, as this note prolongs the motion along the root edge.

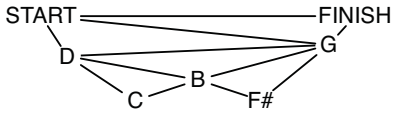


Figure 4. A MOP containing initiation and termination events.

3.2 Converting the Corpus to MOPs

We now present an algorithm to convert a text file analysis like those in SCHENKER41 to a collection of MOPs. Because a single MOP only represents a monophonic sequence of notes, we may need multiple MOPs to store all of the prolongations in a single text file analysis. Most of the analyses in SCHENKER41 contain at least two MOPs, one representing the structure of the main melody, and one representing structure of the bass line.

The algorithm operates in three phases. In the first phase, we make a pass through the analysis text file to identify which notes will belong to which MOPs. We do this by creating a temporary graph structure consisting of all the notes present in the analysis and initially no edges. For each prolongation in the analysis file $X\ (Y)\ Z$, we add the edges (X, Z) and (i, Z) for each note i in the set of notes Y. After processing every prolongation, every connected component in the graph will correspond to a single MOP.

Phase two adds edges to the MOP for all two-parent prolongations. For each MOP graph identified in phase one, we first remove all the edges, then create a "skeleton" MOP structure consisting of edges connecting only consecutive notes in the music, plus the additional edges involving the START and FINISH vertices. Figure 5(a) illustrates this skeletal structure for the prolongations described in Figure 2. We then create edges in the MOP corresponding to all prolongations in the analysis text file that have two parent notes. Adding appropriate edges is straightforward: for a prolongation $X\ (Y)\ Z$, we add an edge from note X to the first note of the set of notes Y, an edge from the last note of Y to note Z, and an edge from X to Z. If the consecutive notes of Y are not already connected to each other by edges, we also add such edges. At the end of phase two, we would have a structure like in Figure 5(b).

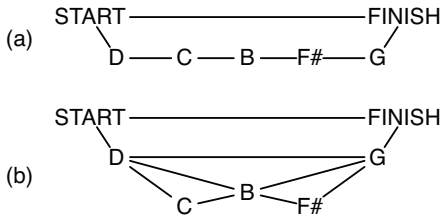


Figure 5. The (a) beginning and (b) end of phase two of creating a MOP.

Phase three involves adding edges in the MOP for one-parent prolongations, i.e., prolongations in the analysis text file of the form $X\ (Y)$ or $(Y)\ Z$. We begin by adding edges between consecutive notes of Y as in phase two. The next step is identifying any additional edges necessary to enforce that the notes of Y should be lower in the hierarchy than X or Z, whichever parent note is present. Fortunately, it is guaranteed that every one-parent prolongation will fall into one of the six categories described below, each of which we handle separately. We briefly describe the six categories here, and their processing steps are fully described in the pseudocode of Algorithm 1. The code refers to the "smallest interior polygon" for a one-parent prolongation p, which is the smallest polygon in the MOP containing all the notes of p (the parent note and all of the child notes). This interior polygon will always exist in a MOP because MOPs express a strict hierarchy among the notes, and therefore all the notes of a prolongation will be found within a single polygon.

Category 1 corresponds to a one-parent prolongation missing a right parent, where the MOP already contains an edge connecting the left parent X to the first note of Y, and the edge in question already implies a hierarchical relationship between X and Y. In this situation, there are no extra edges to add because the necessary hierarchical relationship already exists. Category 2 corresponds to the same situation as Category 1, but reversed for a missing

Algorithm 1

procedure PROCESS-ONE-PARENT-PROLONGATIONS
  Let S be the set of one-parent prolongations.
  while $S \neq \emptyset$ **do**
    $p \leftarrow$ shortest length prolongation in S
    $I \leftarrow$ identify smallest interior polygon containing all notes of p
    Assume vertices of I are numbered $0...m-1$
    if leftParent(p) $= I[0]$ and firstChildNote(p) $= I[1]$ **then** ▷ Category 1
      $S \leftarrow S - \{p\}$ ▷ No additional edges needed; p's children are already lower in the hierarchy
    else if rightParent(p) $= I[m-1]$ and lastChildNote(p) $= I[m-2]$ **then** ▷ Category 2
      $S \leftarrow S - \{p\}$ ▷ No additional edges needed; p's children are already lower in the hierarchy
    else if leftParent(p) $= I[0]$ **then** ▷ Category 3
      Add edge (leftParent(p), firstChildNote(p)) to MOP; $S \leftarrow S - \{p\}$
    else if rightParent(p) $= I[m-1]$ **then** ▷ Category 4
      Add edge (rightParent(p), lastChildNote(p)) to MOP; $S \leftarrow S - \{p\}$
    else if rightParent(p) is missing **then** ▷ Category 5
      newRight $\leftarrow$ earliest $I[x]$ such that $I[x]$ is later than all of p's children
      if choice of newRight increases length of prolongation p **then**
        Update p's length in S; defer processing
      else
        Add edge (leftParent(p), newRight) to MOP; $S \leftarrow S - \{p\}$
    else if leftParent(p) is missing **then** ▷ Category 6
      newLeft $\leftarrow$ latest $I[x]$ such that $I[x]$ is earlier than all of p's children
      if choice of newLeft increases length of prolongation **then**
        Update p's length in S; defer processing
      else
        Add edge (newLeft, rightParent(p)) to MOP; $S \leftarrow S - \{p\}$

left parent note.

Category 3 corresponds to a one-parent prolongation missing a right parent, where the the MOP does *not* contain an edge connecting the left parent X to the first note of Y, but other nearby edges already imply a hierarchical relationship between X and Y. Here, we only need to add an edge from X to the first child note of Y. Category 4 corresponds to the same situation as Category 3, but reversed for a missing left parent.

Category 5 corresponds to a one-parent prolongation missing a right parent, where the the MOP does *not* contain an edge connecting the left parent X to the first note of Y, and *no other edges* in the MOP already imply a hierarchical relationship between X and Y. In this situation we must explicitly find a suitable right parent note, which we choose to be the temporally earliest note on the interior polygon that is later than all the notes of Y. Category 6 corresponds to the same situation as Category 5, but reversed for a missing left parent.

4. CONCLUSIONS

In this paper, we presented SCHENKER41, the first large-scale data set of musical compositions and corresponding Schenkerian analyses in a computer-processable format. We anticipate that with the rise of corpus-driven research in music informatics, this data set will be of value to researchers investigating various characteristics of Schenkerian analysis, using machine learning techniques to study the analytical procedure, or harnessing the analyses for use in other music informatics tasks. We also presented an algorithm for translating the analyses into MOPs, which serve as useful data structures for representing the hierarchical organization of the analyses.

5. REFERENCES

[1] Matthew Brown. *Explaining Tonality*. University of Rochester Press, 2005.

[2] Allen Cadwallader and David Gagné. *Analysis of Tonal Music: A Schenkerian Approach*. Oxford University Press, Oxford, 1998.

[3] Allen Forte and Steven E. Gilbert. *Instructor's Manual for* Introduction to Schenkerian Analysis. W. W. Norton and Company, New York, 1982.

[4] Allen Forte and Steven E. Gilbert. *Introduction to Schenkerian Analysis*. W. W. Norton and Company, New York, 1982.

[5] Phillip B. Kirlin. *A Probabilistic Model of Hierarchical Music Analysis*. PhD thesis, University of Massachusetts Amherst, 2014.

[6] Phillip B. Kirlin and David D. Jensen. Probabilistic modeling of hierarchical music analysis. In *Proceedings of the 12th International Society for Music Information Retrieval Conference*, pages 393–398, 2011.

[7] Alan Marsden. Schenkerian analysis by computer: A proof of concept. *Journal of New Music Research*, 39(3):269–289, 2010.

[8] Eugene Narmour. *Beyond Schenkerism: The Need for Alternatives in Music Analysis*. University of Chicago Press, 1977.

[9] Tom Pankhurst. *SchenkerGUIDE: A Brief Handbook and Website for Schenkerian Analysis*. Routledge, New York, 2008.

[10] Christopher Raphael. Symbolic and structural representation of melodic expression. In *Proceedings of the 10th International Society for Music Information Retrieval Conference*, pages 555–560, 2009.

[11] Jason Yust. *Formal Models of Prolongation*. PhD thesis, University of Washington, 2006.

Composer	Excerpt name	Analysis source
Bach	Minuet in G major, BWV Anh. 114, mm. 1–16	Expert
Bach	Chorale 233, Werde munter, mein Gemute, mm. 1–4	Expert
Bach	Chorale 317 (BWV 156), Herr, wie du willt, so schicks mit mir, mm. 1–5	F&G manual
Beethoven	Seven Variations on a Theme by P. Winter, WoO 75, Variation 1, mm.1–8	C&G
Beethoven	Seven Variations on a Theme by P. Winter, WoO 75, Theme, mm. 1–8	C&G
Beethoven	Ninth Symphony, Ode to Joy theme from finale (8 measures)	SG
Beethoven	Piano Sonata in F minor, Op. 2, No. 1, Trio, mm. 1–4	SG
Beethoven	Seven Variations on God Save the King, Theme, mm. 1–6	SG
Chopin	Mazurka, Op. 17, No. 1, mm. 1–4	SG
Chopin	Grande Valse Brilliante, Op. 18, mm. 5–12	SG
Clementi	Sonatina for Piano, Op. 38, No. 1, mm. 1–2	SG
Handel	Trio Sonata in B-flat major, Gavotte, mm. 1–4	Expert
Haydn	Divertimento in B-flat major, Hob. 11/46, II, mm. 1–8	F&G
Haydn	Piano Sonata in C major, Hob. XVI/35, I, mm. 1–8	F&G
Haydn	Twelve Minuets, Hob. IX/11, Minuet No. 3, mm. 1–8	SG
Haydn	Piano Sonata in G major, Hob. XVI/39, I, mm. 1–2	SG
Haydn	Hob. XVII/3, Variation I, mm. 19–20	SG
Haydn	Hob. I/85, Trio, mm. 39–42	SG
Haydn	Hob. I/85, Menuetto, mm. 1–8	SG
Mozart	Piano Sonata 11 in A major, K. 331, I, mm. 1–8	F&G
Mozart	Piano Sonata 13 in B-flat major, K. 333, III, mm. 1–8	F&G manual
Mozart	Piano Sonata 16 in C major, K. 545, III, mm. 1–8	F&G manual
Mozart	Six Variations on an Allegretto, K. Anh. 137, mm. 1–8	F&G manual
Mozart	Piano Sonata 7 in C major, K. 309, I, mm. 1–8	C&G
Mozart	Piano Sonata 13 in B-flat major, K. 333, I, mm. 1–4	F&G
Mozart	7 Variations in D major on “Willem van Nassau,” K. 25, mm. 1–6	SG
Mozart	Twelve Variations on “Ah vous dirai-je, Maman,” K. 265, Var. 1, mm. 23–32	SG, C&G
Mozart	12 Variations in E-flat major on “La belle Françoise,” K. 353, Theme, mm. 1–3	SG
Mozart	Minuet in F for Keyboard, K. 5, mm. 1–4	SG
Mozart	8 Minuets, K. 315, No. 1, Trio, mm. 1–8	SG
Mozart	12 Minuets, K. 103, No. 4, Trio, mm. 15–16	SG
Mozart	12 Minuets, K. 103, No. 3, Trio mm. 7–8,	SG
Mozart	Untitled from the London Sketchbook, K. 15a, No. 1, mm. 12–14	SG
Mozart	9 Variations in C major on “Lison dormait,” K. 264, Theme, mm. 5–8	SG
Mozart	12 Minuets, K. 103, No. 12, Trio, mm. 13–16	SG
Mozart	12 Minuets, K. 103, No. 1, Trio, mm. 1–8	SG
Mozart	Piece in F for Keyboard, K. 33B, mm. 7–12	SG
Schubert	Impromptu in B-flat major, Op. 142, No. 3, mm. 1–8	F&G manual
Schubert	Impromptu in G-flat major, Op. 90, No. 3, mm. 1–8	F&G manual
Schubert	Impromptu in A-flat major, Op. 142, No. 2, mm. 1–8	C&G
Schubert	Wanderer’s Nachtlied, Op. 4, No. 3, mm. 1–3	SG

Table 1. The musical excerpts contained in SCHENKER41.

SYSTEMATIC MULTI-SCALE SET-CLASS ANALYSIS

Agustín Martorell
Universitat Pompeu Fabra
agustin.martorell@upf.edu

Emilia Gómez
Universitat Pompeu Fabra
emilia.gomez@upf.edu

ABSTRACT

This work reviews and elaborates a methodology for hierarchical multi-scale set-class analysis of music pieces. The method extends the systematic segmentation and representation of Sapp's 'keyscapes' to the description stage, by introducing a set-class level of description. This provides a systematic, mid-level, and standard analytical lexicon, which allows the description of any notated music based on fixed temperaments. The method benefits from the representation completeness, the compromise between generalisation and discrimination of the set-class spaces, and the access to hierarchical inclusion relations over time. The proposed *class-matrices* are multidimensional time series encoding the pitch content of every possible music segment over time, regardless the involved time-scales, in terms of a given set-class space. They provide the simplest information mining methods with the ability of capturing sophisticated tonal relations. The proposed *class-vectors*, quantifying the presence of every possible set-class in a piece, are discussed for advanced explorations of corpora. The compromise between dimensionality and informativeness provided by the class-matrices and class-vectors, is discussed in relation with standard content-based tonal descriptors, and music information retrieval applications.

1. INTRODUCTION

Pitch-class set theory has been used in music analysis practice since decades. However, its general applicability to post-tonal music has contributed, and still contributes, to be perceived as for specialists only. This apparent difficulty is far from real, and just a matter of the application context. The systematic and objective nature of the theory, together with the compactness of the basic representations, constitutes a powerful and flexible descriptive framework suited for any kind of pitch-based music. [1] This description level is purposeful for several music information retrieval (MIR) applications, such as structural analysis, similarity, pattern finding, classification, and generation of content-based metadata. More interestingly, it provides a means for approaching complex topics, such as similarity, in alternative and insightful musically-grounded scenarios. In addition, the basic descriptors are trivial to compute, and they can be readily exploited by standard information mining techniques.

The remaining of this work is organised as follows. Section 2 introduces the basic set-theoretical concepts, and contextualise them in terms of our systematic analysis endeavour. Section 3 describes the computational approach. Sections 4 and 5 discuss the method in several application contexts. Section 6 summarises the proposed method, and points to future extensions.

2. BACKGROUND

2.1 Set-class description

Pitch class [1] is defined, in the twelve-tone equal tempered system (TET), as an integer representing the residue class modulo 12 of a pitch, that is, any pitch is mapped to a pitch class by removing its octave information. A *pitch-class set* (henceforth *pc-set*) is a set of pitch classes without repetitions in which the order of succession of the elements in the set is not of interest. In the TET system, there exist $2^{12} = 4096$ distinct pc-sets, so a vocabulary of 4096 symbols is required for describing any possible segment of music. Any pc-set can also be represented by its intervallic content [5]. Intervals considered regardless of their direction are referred to as *interval classes*. The total count of interval classes in a pc-set can be arranged as a six-dimensional data structure called an *interval vector* [4].

Relevant relational concepts for analysis are the *set-class equivalences*, whereby two pc-sets are considered equivalent if and only if they belong to the same *class*. As pointed out by Straus, equivalence is not the same thing as identity, rather it is a link between musical entities that have something in common. This commonality underlying the surface may eventually lend unity and/or coherence to musical works [12]. In this respect, the class equivalences can be conceived as *all or nothing* similarity measures between two pc-sets. In the context of pc-sets, the number of pitch classes in a set is referred to as its *cardinality*. This is perhaps the coarsest measure of similarity. Despite its theoretical relevance, cardinality is too general a notion of similarity to be of use in many analytical situations. Among the many equivalence systems in the set-theoretical

[1] In which the concepts of octave equivalence and fixed temperaments are applicable. Although the pitch relations of interest may be quite different, depending on the temperament and the applied context, any discrete pitch organization of the octave can be handled by the general mathematical framework. In this work, we bound to the twelve-tone equal temperament.

literature, three of them are particularly useful:

1. *Interval vector equivalence* (iv-equivalence), which groups all the pc-sets sharing the same interval vector. There exist 197 different iv-types.

2. *Transpositional equivalence* (T_n-equivalence), which groups all the pc-sets related to each other by transposition. There exist 348 distinct T_n-types.

3. *Inversional and transpositional equivalence* (T_nI-equivalence), which groups all the pc-sets related by transposition and/or inversion. There exist 220 different T_nI-types (also referred to as T_n/T_nI-types).

Aside the comprehensive coverage of every possible pc-set, the compromise between discrimination and generalisation of these class-equivalence systems fits a wide range of descriptive needs, thus their extensive usage in general-purpose music analysis. From them, iv-equivalence is the most general (197 classes). It shares most of its classes with T_nI-equivalence (220 classes), with some exceptions, named *Z-relations* [4], for which the same interval vector groups pc-sets which are not T_nI-equivalent [7]. The most specific from the three systems is T_n-equivalence.

2.2 Systematic approaches to set-class analysis

To date, one of the most systematic approaches to set-class surface analysis is proposed in [6], under the concept of 'tail-segment array', whereby every note in a composition is associated with all the possible segments of a given cardinality that contains it. This segmentation is combined with certain set-class-based 'detector functions', in order to obtain summarized information from music pieces and collections. The usefulness of the method is comprehensively discussed in the context of style characterization. Some limitations of this technique are addressed in [8], by first identifying the segmentation, description and representation stages of the method, and extending systematization to all of them simultaneously. This is done by combining the exhaustive segmentation and representation of Sapp's 'keyscapes' [11], with a systematic description of the segments in terms of set-classes. The multidimensional, massive and overlapping information resulting from this method, is managed by summarising features and interfacing design, targeting specific analytical tasks.

3. MULTI-SCALE SET-CLASS ANALYSIS

This work elaborates directly upon [8], in which detailed and extended discussions can be consulted. A description of our general method follows.

3.1 Segmentation

The input to the system is a sequence of MIDI events, which can be of any rhythmic or polyphonic complexity. This signal is processed by the segmentation stage, for which two different algorithms are used: a) an approximate technique, non comprehensive but practical for interacting with the data; b) a fully systematic method, which exhausts all the segmentation possibilities.

The approximate method applies many overlapping sliding windows, each of them scanning the music at a different time-scale. The minimum window size and the number of time-scales are user parameters, and can be fine tuned as a trade-off between resolution and computational cost. The same hop size is applied for all the time-scales, in order to provide a regular grid for visualisation and interfacing purposes. Each segment is thus indexed by its centre location (time) and its duration (time-scale).

The fully systematic method is required for the quantitative descriptors in which completeness of representation is necessary. It is computed by finding every change in the pc-set content, whether the product of onsets or offsets, and segmenting the piece by considering all the pairwise combinations among these boundaries.

3.2 Description

Denoting pitch-classes by the ordinal convention (C=0, . . . , B=11), each segment is analysed as follows. Let $b_i = 1$ if the pitch-class i is contained (totally or partially) in the segment, or 0 otherwise. The pc-set in the segment is encoded as an integer $p = \sum_{i=0}^{11} b_i \cdot 2^{11-i} \in$ [0,4095]. This integer serves as an index for a precomputed table of set classes,[2] including the iv-, T_nI- and T_n-equivalences (discussed in Section 2.1). For systematisation completeness, the three class spaces are extended to include the so-called *trivial forms*.[3] With this, the total number of interval vectors rises to 200, while the T_nI- and T_n-equivalence classes sum to 223 and 351 categories respectively. In this work, we use Forte's cardinality-ordinal convention [4] to name the classes, as well as the usual A/B suffix for referring to the prime/inverted forms under T_n-equivalence. We also follow the conventional notation to name the Z-related classes, by inserting a 'Z' between the hyphen and the ordinal. As an example, a segment containing the pitches {G5,C3,E4,C4} is mapped to the pc-set {0,4,7} and coded as $p = 2192$ (100010010000 in binary). The precomputed table is indexed by p, resulting in the interval vector ⟨001110⟩ (iv-equivalence, grouping all the sets containing exactly 1 minor third, 1 major third, and 1 fourth), the class 3-11 (T_nI-equivalence, grouping all the major and minor trichords), and the class 3-11B (T_n-equivalence, grouping all the major trichords). The discrimination between major and minor trichords is thus possible under T_n-equivalence (3-11A for minor, 3-11B for major), but not under iv- or T_nI-equivalences.

3.3 Representation

The main data structure, named *class-scape*, is the set-class equivalent of Sapp's 'keyscapes' [11]. It represents the class content of every possible segment, indexed by

[2] As formalised in [4]. See *Supplemental material* (Section 7).

[3] The null set and single pitch classes (cardinalities 0 and 1, containing no intervals), the undecachords (cardinality 11) and the universal pc-set (cardinality 12, also referred to as the *aggregate*).

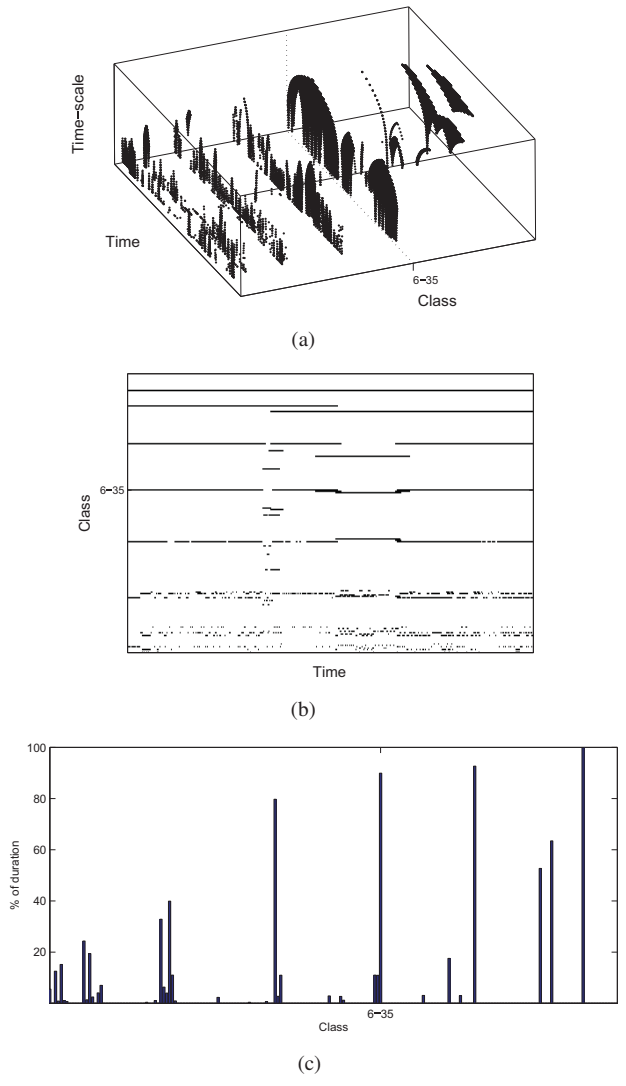


Figure 1: Debussy's *Voiles*. a) class-scape; b) class-matrix; c) class-vector.

their time position and duration. The dimensionality of the class-scapes (time, time-scale and class) is then reduced to more manageable, yet informative, data structures. The first reduction consists on projecting the class-scape to the time-class plane, which results in the concept of *class-matrix*. This is done by realising in time each point in the class-scape, thus retaining a substantial information from the lost dimension (time-scale). A further reduction summarizes the class-matrix in a single vector, named *class-vector*, by quantifying the presence of every possible class in the piece as a percentage of the piece's duration. The class-scape, class-matrix and class-vector, computed from Debussy's *Voiles* are depicted in Figure 1, with the prominent whole-tone scale (class 6-35) labelled as a reference.

4. MINING CLASS-MATRICES

In this section, we will review and elaborate upon the information conveyed by the class-matrices. Even with the loss of information, the reduction process from the class-scape to the class-matrix guarantees that every instantiation of every class is represented in the class-matrix, regardless the involved time-scales. The class-matrix represents the temporal *activation* of every possible class over time. A time *point* activated for a given class in the matrix means that it exist at least one segment containing this time point which belongs to such class. As the representation guarantees a strict class-wise separation, the class matrix constitutes a time-series of a special kind. It does not only capture evidence from every class instantiation over time, but it also informs about their set-class *inclusion relations*. The class-matrix, thus, embeds a considerable hierarchical information, allowing the analysis of the specific *constructions* of the class instantiations.

4.1 Case study: subclass analysis

An example of this analytical potential is depicted in Figure 2. It shows the comparison between the *pure diatonicisms* in Victoria's parody masses in Ionian mode [4] and Bach's preludes and fugues in major mode from the Well Tempered Clavier. This is done by first isolating the diatonic segments (activation of 7-35 in the class-matrix) of each movement, and constructing a *subclass-matrix* with the subset content of these segments. The differences can be quantified by computing the corresponding *subclass-vectors* out of the subclass-matrices, and averaging them across pieces in the corpora. This tells about what the particular diatonicisms (and only the diatonicisms) are made of. Some relevant differences stand out from the comparison. Victoria's larger usage of major and minor triads (3-11) and cadential chord sequences (5-27) stands out. On the other hand, Bach makes more prominent usage of the scalar formation 6-Z25: aside its instantiations as perfect cadences, it is recurrent in many motivic progressions, which are not idiomatic in Victoria's contrapuntal writing.

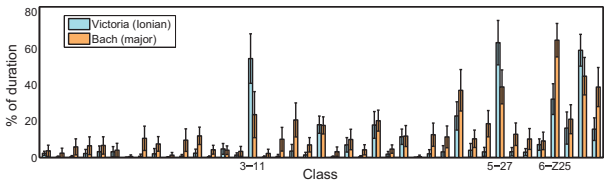


Figure 2: Diatonicism in Victoria and Bach. Mean subclass-vectors under 7-35.

4.2 Case study: structural analysis

Self-similarity matrices (SSM) are a simple standard tool used for structural analysis [3]. Classical inputs to the SSM are spectral or chroma feature time series. Some of the SSM-based methods can handle different time-scales, and some of the chroma methods allows transpositional invariance [9]. These functionalities are usually implemented at the SSM computation stage, or as a post processing. In the class-matrices, both the equivalence mappings (including their inherent hierarchies) and the multi-scale na-

[4] Including *Alma Redemptoris Mater*, *Ave Regina Caelorum*, *Laetatus Sum*, *Pro Victoria*, *Quam Pulchri Sunt*, and *Trahe Me Post Te*. See (Rive, 1969) for a modal classification.

ture of the information are *already* embedded in the feature time-series, so a plain SSM can be used for finding sophisticated recurrences. For instance, a passage comprised of a chord sequence can be recognized as similar than a restated passage with different arpeggiations and/or inversions of the chord intervals (e.g. from major to minor triads). A *vertical* chord and its arpeggiated version may not be recognized as very similar at the lowest cardinalities, but their common T_nI-sonority will certainly do at their corresponding time-scales. Moreover, any sonority containing the chords (*supersets*) will also be captured at their proper time-scales, climbing up the hierarchy until reaching the whole-piece segment, everything indexed by a common temporal axis. A quantification of similarity between variations may thus be possible at the level of embedded sonorities.

This is discussed next for large-scale recurrence finding in Webern's *Variations for piano*, op.27/I. This serial piece presents an *A-B-A*' structure, built upon several instantiations of the main twelve-tone row, at different transpositional and/or inversional levels. Figure 3 (top) depicts the class-scape of the piece, filtered by the prominent hexachordal iv-sonority ⟨332232⟩, and Figure 3 (bottom) shows the well-known (extensively analysed in literature) structure of the row instantiations, annotated according to [2]. Figure 4 depicts the output of a plain SSM, computed from three different inputs: a) the pc-set time series;[5] b) the class-matrix under T_n; c) the class-matrix under T_nI. The pc-equivalence does not capture any large-scale recurrence. The restatement of the first two phrases in A is captured by the T_n-equivalence, as these phrases are mainly related by transposition in A'. Finally, the T_nI-equivalence reveals the complete recapitulation, including the last two phrases of A, which are restated in A' in both transposed and inverted transformations. It is worth noting that the method does not limit to compare the general sonority, the ubiquitous ⟨332232⟩, but its specific construction down the subclass hierarchy. This allows the discrimination of the *B* section, built upon the same kind of row instantiations than A and A', but presented in distinct harmonisations.

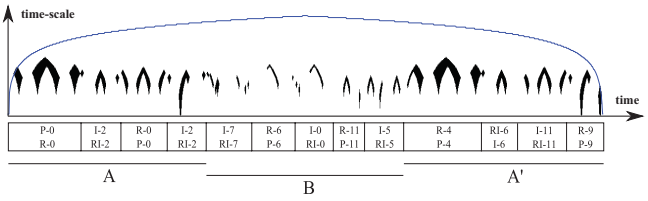


Figure 3: Webern's op.27/I. Top: class-scape filtered by ⟨332232⟩; Bottom: structure.

A relevant advantage of the pc-set-based spaces, with respect to *continuous* ones,[6] is that music can be analysed in terms of different class systems at no extra computational cost. Being *finite and discrete* spaces (4096 classes at most for the TET system), the whole equivalence systems, including their inner metrics, can be precomputed.

[5] In some respect, the discrete *equivalent* of the chroma features.

[6] Such as chroma features, a *finite, but continuous* space.

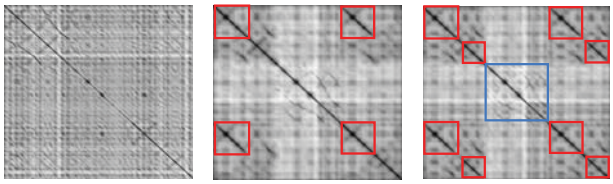


Figure 4: SSM from Webern's op.27/I. a) pc-equivalence; b) T_n-equivalence); c) T_nI-equivalence).

The mapping from pc-sets to set-classes, as well as the distances between any pair of music segments, can thus be implemented by table indexing. Once the pc-set of each possible segment has been computed (which constitutes the actual bottleneck of the method), the rest of the process is inexpensive, and multiple *set-class lenses* can be changed in real time, allowing fast interactive explorations of the massive data. This feature, alongside with a variety of filtering options for visual exploration, can be tested with our proof-of-concept set-class analysis tool.[7]

5. MINING CLASS-VECTORS

In this section, we will review and elaborate upon the information conveyed by the class-vectors. For each class, the corresponding value in the vector accounts for the relative duration of the piece which is *interpretable* in terms of the specific class, that means, the proportion of time points which are contained in some (at least one) instance of the class. A dataset of class-vectors, thus, can be exploited in a variety of ways. Finding specific sonorities in large datasets can be combined with the extraction of the actual segments from the MIDI files. This can be exploited in varied applications, ranging from corpora analysis to music education.

A dataset of class-vectors was computed from 13480 MIDI tracks, including works by Albéniz, Albinoni, Alkan, Bach, Beethoven, Brahms, Bruckner, Busoni, Buxtehude, Byrd, Chopin, Clementi, Corelli, Couperin, Debussy, Dowland, Frescobaldi, Gesualdo, Guerrero, Haydn, Josquin, Lasso, Liszt, Lully, Mahler, Morales, Mozart, Pachelbel, Palestrina, Satie, Scarlatti, Shostakovich, Schumann, Scriabin, Soler, Stravinsky, Tchaikovsky, Telemann, Victoria and Vivaldi. It also includes anonymous medieval pieces, church hymns, and the Essen folksong collection.

5.1 Case study: query by set-class

A simple but useful application is querying the dataset for a given set-class sonority. It can be used, for instance, to find pieces with a relevant presence of *exotic* scales. Table 1 shows 10 retrieved pieces with a notable presence (relative duration) of the sonority 7-22, usually referred to as the Hungarian minor scale.[8] Both monophonic and polyphonic pieces are retrieved, ranging different styles

[7] See *Supplemental material* (Section 7).

[8] Sometimes also called Persian, major gypsy, or double harmonic scale, among other denominations.

and historic periods, as the unique requisite for capturing a given sonority it its existence as a temporal segment.

retrieved piece	7-22 (%)
Scriabin - Prelude op.33 n.3	68.61
Busoni - 6 etudes op.16 n.4	63.22
Essen - 6478	62.50
Liszt - Nuages gris	42.41
Essen - 531	36.67
Scriabin - Prelude op.51 n.2	31.74
Lully - Persee act-iv-scene-iv-28	29.73
Alkan - Esquisses op.63 n.19	28.87
Satie - Gnossienne n.1	28.15
Scriabin - Mazurka op.3 n.9	24.61

Table 1: Retrieved pieces: 7-22

5.1.1 Query by combined set-classes

The strict separation of classes in the class-vectors, allows the exploration of any class combination, whether common or unusual. For instance, the first movement of Stravinsky's *Symphony of psalms* is retrieved by querying for music containing substantial diatonic (7-35) and octatonic (8-28) material, certainly an uncommon musical combination. The class-vector also reveals the balance between both sonorities, as 30.18 % and 29.25 % of the piece duration, respectively. As discussed in Section 4, the class-matrices allow the hierarchical analysis of specific sonorities. The class-vectors, on the other hand, summarise the information in a way in which it is not possible, in general, to elucidate the subclass content under a given class. However, if the queried sonorities have a substantial presence (or absence) in the piece, the class-vectors alone can often account for some hierarchical evidence. Table 2 shows 10 retrieved pieces, characterised by a notable presence of the so-called *suspended* trichord (3-9),[9] constrained to cases of mostly diatonic contexts (7-35). This situation, as reflected in the results, is likely to be found in medieval melodies, early counterpoint, or works composed as reminiscent of them. It is worth noting that the 3-9 instantiations appear in quite different settings, whether in monophonic voices, as a combination of melody and tonic-dominant drones, and as actual suspended (voiced) chords.

retrieved piece	3-9 (%)	7-35 (%)
Anonym - Angelus ad virginem 1	56.79	100
Anonym - Instrumental dances 7	50.41	100
Lully - Persee prologue-3c	47.11	100
Lully - Phaeton acte-i-scene-v	45.95	90.81
Lully - Persee prologue-3b	44.36	100
Anonym - Ductia	43.82	100
Anonym - Danse royale	35.58	100
Anonym - Cantigas de Santa Maria 2	32.06	100
Anonym - Instrumental dances 9	27.43	100
Frescobaldi - Canzoni da sonare-11	26.69	81.34

Table 2: Retrieved pieces: mostly 7-35 with 3-9.

As non-existing sonorities may also reveal important characteristics of music, the dataset can be queried for combinations of present and absent classes. For instance, the sonority of fully diatonic (7-35) pieces depends on whether they contain major or minor trichords (3-11) or not. Retrieved pieces in the latter case (diatonic, not triadic) are mostly medieval melodies or early polyphonic pieces, prior to the establishment of the triad as a common sonority.

These results point to interesting applications related with music similarity, such as music recommendation and music education. We find of particular interest the potential of retrieving pieces sharing relevant tonal-related properties, but pertaining to different styles, composers, or historical periods. Music similarity is, to a great extent, a human construct, as it depends on cultural factors and musical background. It would thus be possible to *learn* to appreciate non familiar similarity criteria, which could be suggested by music discovery or recommendation systems.

5.2 On dimensionality and informativeness

In feature design, the ratio between the size of the feature space and the informativeness of description is a relevant factor. The class content of a piece, as described by its class-vector, have 200, 223 or 351 dimensions, depending on the chosen equivalence (iv, T_nI or T_n). Compared with other tonal feature spaces, these dimensions may seem quite large. However, the benefits of class vectors are the systematicity, specificity and precision of the description. Several relevant differences with respect to other tonal-related features are to be noticed. A single class-vector, computed after a fully systematic segmentation, accounts for:

1. Every different segment in the piece, regardless of their time position or duration. No sampling artefacts of any kind are introduced.

2. Every possible sonority among the set-class space, which is *complete*. Every instantiation of every class is captured and represented.

3. An objective and precise description of the *set-class sonority*. No probabilities or estimations are involved.

4. A description in (high level) music theoretical terms, readable and interpretable by humans. Set-classes constitute a standard lexicon in music analysis.

5. An objective quantification of every possible sonority in terms of relative duration in the piece. No probabilities or estimations are involved.

6. A content-based, model-free, description of the piece. Neither statistics nor properties learned from datasets are involved.

7. In cases of large presence or notable absences of some sonorities, an approximation to the hierarchical inclusion relations (fully available through the class-matrices only).

In contrast, the most common tonal piecewise and labelwise feature (global key estimation) conveys:

[9] A major trichord with the third degree substituted by the fourth.

1. A single label for the whole piece, often misleading for music which modulates.

2. 24 different labels, but actually two different sonorities (major and minor), non representative of a vast amount of music.

3. An *estimation* of the key: not only because of the inherent ambiguity of tonality, but also because the (most often) limited tonal *knowledge* of the algorithms.

4. A description in (high level) music theoretical terms, but conveying very little musical information (e.g. at compositional level).

5. No quantification, just a global label. At most, including an indicator of *confidence* (in the descriptor terms), usually the key strength.

6. A description based on specific models (e.g. profiling methods or rule-based), which do not generalize. Some models are trained from specific datasets, biasing the actual *meaning* of the descriptor.

7. No access to the (very rich) hierarchical relations of the piece's tonality.

With this in mind, it seems to us that a piecewise description in 200 dimensions is a reasonable trade-off between size and informativeness. Considering the somewhat sophisticated tonal information conveyed by the class-vectors, they may constitute a useful complementary feature for existing content-based metadata.

6. CONCLUSIONS

The proposed systematic methodology for multi-scale set-class analysis is purposeful for common music information retrieval applications. An appropriate mining of the class-matrices can bring insights about the hierarchical relations among the sets, informing about the specific construction of the class sonorities. In combination with simple recurrence finding methods, the class-matrices can be used for music structure analysis of complex music, beyond the scope of mainstream tonal features. The proposed class-vectors, as piecewise tonal summaries, convey a rich information in terms of every possible class sonority. They can be mined for querying tasks of some sophistication. Their compromise between dimensionality and informativeness, point to potential advances in music similarity and recommendation applications. The examples in this work show that set-classes can inform about very different music compositions, ranging simple folk tunes, early polyphony, common-practice period, *exotic* or uncommon scales, and atonal music. Besides our ongoing musicological analyses, and current research with chroma-based transcriptions from audio, future work may explore the potential of these methods in actual classification and recommendation systems.

7. SUPPLEMENTAL MATERIAL

The interactive potential of the methods discussed in this work can be tested by our multi-scale set-class analysis prototype for Matlab, freely available from http://agustinmartorell.weebly.com/set-class-analysis.html. A comprehensive table of set-classes, and a growing dataset of class-vectors, are also available at this site.

8. ACKNOWLEDGEMENTS

This work was supported by the EU 7th Framework Programme FP7/2007-2013 through PHENICX project [grant no. 601166].

9. REFERENCES

[1] M. Babbit: "Some Aspects of Twelve-Tone Composition," *The Score and I.M.A. Magazine*, Vol. 12, pp. 53–61, 1955.

[2] N. Cook: *A Guide to Music Analysis*, J. M. Dent and Sons, London, 1987.

[3] J. Foote: "Visualizing Music and Audio Using Self-Similarity," *Proceedings of the ACM Multimedia*, pp. 77–80, 1999.

[4] A. Forte: "A Theory of Set-Complexes for Music," *Journal of Music Theory*, Vol. 8, No. 2 pp. 136–183, 1964.

[5] H. Hanson: *The Harmonic Materials of Modern Music: Resources of the Tempered Scale*, Appleton-Century-Crofts, New York, 1960.

[6] E. Huovinen and A. Tenkanen: "Bird's-Eye Views of the Musical Surface: Methods for Systematic Pitch-Class Set Analysis," *Music Analysis*, Vol. 26, No. 1–2 pp. 159–214, 2007.

[7] D. Lewin: "Re: Intervallic Relations between Two Collections of Notes," *Journal of Music Theory*, Vol. 3, No. 2 pp. 298–301, 1959.

[8] A. Martorell and E. Gómez: "Hierarchical Multi-Scale Set-Class Analysis," *Journal of Mathematics and Music*, Online pp. 1-14, 2014.

[9] M. Müller: *Information Retrieval for Music and Motion*, Springer, Berlin, 2007.

[10] T. N. Rive: "An Examination of Victoria's Technique of Adaptation and Reworking in his Parody Masses - with Particular Attention to Harmonic and Cadential Procedure," *Anuario Musical*, Vol. 24, pp. 133–152, 1969.

[11] C. S. Sapp: "Visual Hierarchical Key Analysis," *Computers in Entertainment*, Vol. 4, No. 4 pp. 1–19, 2005.

[12] J. N. Straus: *Introduction to Post-Tonal Theory*, Prentice-Hall, Upper Saddle River, NJ, 1990.