

An Introduction to Audio Content Analysis v2

Confidential Manuscript (all rights reserved)

DO NOT SHARE
DO NOT DISTRIBUTE

Alexander Lerch

September 16, 2021

Preface

- state of ACA and MIR
 - steadily growing (compare ISMIR numbers)
 - ...
- goal: provide starting point through baseline approaches and relevant references and resources
- deep learning prevalent
- more datasets out there but still far from enough
- (knowledge of) baseline systems matters as
 - informs system and parametrization choices
 - gives a better understanding of the tasks
 - gives better understanding of underlying (design) principles
 - allows for more hands-on exercises

What's new

- code examples, now also with Python
- Evaluation section
- several new topics
- questions and assignments
- rewritten and restructured (some parts had unnecessary detail, others unhelpful shortness, evolution of field makes it easier to formalize and structure, approaches that might have seemed important then are not anymore, others which might have seems unimportant became more important, blablabla)
- more online resources (code, datasets, slides)

Overall: sometimes limited accuracy for the sake of focusing on practical challenges

Preface of the 1st Edition

The growing amount of audio and music data on the Internet and in user databases leads to an increasing need for intelligent browsing, retrieving, and processing of this data with automated methods. *Audio content analysis*, a subfield of the research field *music information retrieval*, aims at extracting (musical and perceptual) properties directly from the audio signal to support these tasks. Knowledge of these properties allows us to improve the interaction of humans or machines with digital audio signals. It enables new ways of assessing, processing, and visualizing music.

Although analysis of audio signals covers other research areas such as automatic speech recognition, we will restrict ourselves to the analysis of music signals in the context of this book.

When preparing classes on audio content analysis with a focus on music recordings it became quickly clear that — although there is a vast and growing amount of research literature available — there exists no introductory literature. This observation led to writing this book in the hope it might assist students, engineers, and developers who have basic knowledge of digital signal processing. The focus lies on the signal processing part of audio content analysis, but wherever it may improve the understanding of either algorithmic design choices or implementation details some basic characteristics of human perception, music theory, and notation as well as machine learning will be summarized.

Chapter A starts by introducing some definitions and offers a short reiteration of the most important tools of digital signal processing for the analysis of audio signals. The following chapters encompass the basic four technical content categories timbre, level, pitch, and rhythm. A fifth category is reserved for purely technical and statistical signal descriptions. Chapter 3.6 introduces low-level or short-term features that are widely used in systems for signal analysis. A large part of the chapter deals with timbre representations of a signal, accompanied by the introduction of statistical features. The chapter concludes with a summary of approaches to feature selection and post-processing. Chapter 8 focuses on intensity-related features. It covers envelope features and simple models of human loudness perception. The extraction of pitch-related information such as the detection of fundamental frequency, harmony, key, etc. is described in Chap. 7. Chapter 9 focuses on the temporal and rhythmic aspects of the audio signal. It explains the segmentation of audio signals into musical events and covers higher level information such as the detection of tempo and meter. The remaining chapters deal with analysis systems using combinations of timbre, loudness, onset, and pitch features to derive higher level information. Chapter 10 describes the automatic synchronization of two similar audio sequences or an audio and a score sequence. Musical genre classification, one of the most prominent research fields of audio content analysis, is explained in Chap. 12. Chapter 11 is about audio fingerprinting which is probably the commercially most successful application in audio content analysis. The concluding chapter, targeting classical music, covers the analysis of music performance. It is not a core field in audio content analysis but emphasizes the differentiation between performance aspects and musical aspects of recordings and elaborates on the manual and automated analysis methods used for musicological music performance analysis. The appendices provide details and derivations of some of the most important signal processing tools as well as a short survey on available software solutions for audio content analysis.

Downloadable MATLAB files are available at: <http://www.audiocontentanalysis.org>.

Contents

Acronyms	19
1 Introduction	25
1.1 A Short History of Audio Content Analysis	26
1.2 Applications and Use Cases	27
1.2.1 Music Browsing and Music Discovery	27
1.2.2 Music Consumption	27
1.2.3 Music Production	27
1.2.4 Music Education	27
1.2.5 Generative Music	28
1.3 Structural Choices and Information	28
I Fundamentals of Audio Content Analysis	29
2 The Audio Content Analysis Process	31
2.1 Audio Content	31
2.2 Audio Content Analysis Flow-Chart	32
2.3 Exercises	33
2.3.1 Questions	33
3 Input Representation	35
3.1 Audio Signals	35
3.1.1 Periodic Signals	35
3.1.2 Random Signals	37
3.1.3 Statistical Signal Description	37
3.1.3.1 Arithmetic Mean	38
3.1.3.2 Geometric Mean	39
3.1.3.3 Harmonic Mean	39
3.1.3.4 Variance and Standard Deviation	39
3.1.3.5 Quantiles and Quantile Ranges	40
3.1.4 Digital Audio Signals	41
3.2 Block-Based Processing	42
3.3 Audio Pre-Processing	43
3.3.1 Down-Mixing	43
3.3.2 DC Removal	44
3.3.3 Normalization	45
3.3.4 Sample Rate Conversion	45
3.3.5 Other Pre-Processing Options	46
3.4 Time-Frequency Representations	46
3.4.1 Fourier Transform	46

3.4.2	Constant Q Transform	47
3.4.3	Log-Mel Spectrogram	50
3.4.4	Auditory Filterbanks	50
3.5	Other Input Representations	52
3.6	Instantaneous Features	52
3.6.1	Spectral Centroid	55
3.6.2	Spectral Spread	56
3.6.3	Spectral Skewness and Spectral Kurtosis	58
3.6.4	Spectral Rolloff	59
3.6.5	Spectral Decrease	60
3.6.6	Spectral Slope	61
3.6.7	Mel Frequency Cepstral Coefficients	63
3.6.8	Spectral Flux	63
3.6.9	Spectral Crest Factor	67
3.6.10	Spectral Flatness	68
3.6.11	Tonal Power Ratio	69
3.6.12	Maximum of Autocorrelation Function	71
3.6.13	Zero Crossing Rate	72
3.6.14	Feature Learning	73
3.7	Feature Post-Processing	75
3.7.1	Derived Features	75
3.7.2	Normalization and Mapping	75
3.7.3	Feature Aggregation	77
3.7.4	Feature Dimensionality Reduction	78
3.7.4.1	Feature Subset Selection	78
3.7.4.2	Feature Space Transformation	80
3.8	Exercises	81
3.8.1	Questions	81
3.8.2	Assignments	82
4	Inference	83
4.1	Classification	83
4.2	Regression	87
4.3	Clustering	88
4.4	Distance and Similarity	88
4.5	Exercises	90
4.5.1	Questions	90
4.5.2	Assignments	90
5	Data	91
5.1	Data split	92
5.2	Artificially Increasing the Amount of Training Data	93
5.3	Utilization of Data from Related Tasks	94
5.4	Reducing Accuracy Requirements for Data Annotation	94
5.5	Semi-, Self-, and Unsupervised Learning	95
5.6	Exercises	95
5.6.1	Questions	95
5.6.2	Assignments	95

6 Evaluation	97
6.1 Good Practices	97
6.2 Metrics	98
6.2.1 Classification	98
6.2.2 Regression	100
6.2.3 Clustering	100
6.3 Exercises	101
6.3.1 Questions	101
6.3.2 Assignments	101
II Music Transcription	103
7 Tonal Analysis	105
7.1 Human Perception of Pitch	105
7.1.1 Pitch Scales	105
7.1.2 Chroma Perception	107
7.2 Representation of Pitch in Music	107
7.2.1 Pitch Classes and Names	108
7.2.2 Intervals	109
7.2.3 The Frequency of Musical Pitch	109
7.2.3.1 Temperament	110
7.2.3.2 Intonation	111
7.3 Fundamental Frequency Detection	111
7.3.1 Detection Accuracy	112
7.3.1.1 Time Domain	112
7.3.1.2 Frequency Domain	113
7.3.1.3 Potential Solutions	113
7.3.2 Pre-Processing	115
7.3.3 Monophonic Input Signals	115
7.3.3.1 Zero Crossing Rate	115
7.3.3.2 Autocorrelation Function	116
7.3.3.3 Average Magnitude Difference Function	118
7.3.3.4 Harmonic Product Spectrum and Harmonic Sum Spectrum	120
7.3.3.5 Autocorrelation Function of the Magnitude Spectrum	122
7.3.3.6 Cepstral Pitch Detection	122
7.3.3.7 Auditory Motivated Pitch Tracking	124
7.3.4 Polyphonic Input Signals	125
7.3.4.1 Iterative Subtraction	125
7.3.4.2 Non-negative Matrix Factorization	126
7.3.4.3 Other Approaches	129
7.3.5 Evaluation	130
7.3.5.1 Metrics	130
7.3.5.2 Datasets	131
7.3.5.3 Results	131
7.4 Tuning Frequency Estimation	131
7.4.1 Approaches to Tuning Frequency Estimation	133
7.4.2 Evaluation	134
7.5 Key Detection	134
7.5.1 Pitch Chroma	136
7.5.1.1 Pitch Chroma Properties	139
7.5.1.2 Features Derived from the Pitch Chroma	141

7.5.2	Approaches to Key Detection	141
7.5.2.1	Key Profiles	141
7.5.2.2	Similarity Measure between Template and Extracted Vector	142
7.5.3	Evaluation	144
7.5.3.1	Metrics	144
7.5.3.2	Datasets	144
7.5.3.3	Results	144
7.6	Chord Recognition	144
7.6.1	Approaches to Chord Recognition	145
7.6.2	Viterbi Algorithm	147
7.6.3	Evaluation	149
7.6.3.1	Metrics	150
7.6.3.2	Datasets	150
7.6.3.3	Results	150
7.7	Exercises	150
7.7.1	Questions	150
7.7.2	Assignments	152
8	Intensity	155
8.1	Human Perception of Intensity and Loudness	155
8.2	Representation of Dynamics in Music	156
8.3	Features	157
8.3.1	Root Mean Square	157
8.3.2	Weighted Root Mean Square	158
8.3.3	Peak Envelope	159
8.3.4	Psycho-Acoustic Loudness Features	160
8.4	Exercises	161
8.4.1	Questions	161
8.4.2	Assignments	162
9	Temporal Analysis	163
9.1	Human Perception of Temporal Events	163
9.1.1	Onsets	163
9.1.2	Tempo and Meter	165
9.1.3	Rhythm	166
9.1.4	Timing	166
9.2	Representation of Temporal Events in Music	166
9.2.1	Tempo and Time Signature	166
9.2.2	Note Value	167
9.3	Onset Detection	168
9.3.1	Novelty Function	168
9.3.2	Peak Picking	169
9.3.3	Evaluation	171
9.3.3.1	Metrics	171
9.3.3.2	Datasets	172
9.3.3.3	Results	172
9.4	Beat Histogram	172
9.4.1	Beat Histogram Features	173
9.5	Detection of Tempo and Beat Phase	174
9.5.1	Evaluation	176
9.5.1.1	Metrics	176
9.5.1.2	Datasets	177

9.5.1.3 Results	177
9.6 Detection of Meter and Downbeat	177
9.7 Structure Detection	178
9.7.1 Self Similarity Matrix	179
9.7.2 Approaches to Structure Detection	180
9.7.2.1 Novelty analysis	180
9.7.2.2 Homogeneity analysis	182
9.7.2.3 Repetition analysis	182
9.7.3 Evaluation	182
9.7.3.1 Metrics	184
9.7.3.2 Datasets	184
9.7.3.3 Results	184
9.8 Exercises	184
9.8.1 Questions	184
9.8.2 Assignments	185
10 Alignment	187
10.1 Dynamic Time Warping	187
10.1.1 Example	191
10.1.2 Common Variants	192
10.1.3 Optimizations	192
10.2 Audio-to-Audio Alignment	193
10.3 Audio-to-Score Alignment	194
10.3.1 Real-Time Systems	195
10.3.2 Non-Real-Time Systems	195
10.4 Evaluation	196
10.4.1 Metrics	196
10.4.2 Data	197
10.5 Exercises	197
10.5.1 Questions	197
10.5.2 Assignments	197
III Music Identification, Classification, and Assessment	199
11 Audio Fingerprinting	201
11.1 Fingerprint Extraction	202
11.2 Fingerprint Matching	203
11.3 Fingerprinting System: Example	203
11.4 Evaluation	206
11.5 Exercise	206
11.5.1 Questions	206
11.5.2 Assignments	206
12 Musical Genre Classification	207
12.1 Approaches to Musical Genre Classification	209
12.2 Evaluation	210
12.2.1 Metrics	211
12.2.2 Data	211
12.2.3 Results	211
12.3 Exercises	211
12.3.1 Questions	211

12.3.2 Assignments	211
13 Music Similarity Detection	213
13.1 Approaches to Music Similarity Computation	213
13.2 Evaluation	214
13.3 Exercises	214
13.3.1 Questions	214
13.3.2 Assignments	214
14 Mood Recognition	215
14.1 Approaches to Mood Recognition	216
14.2 Evaluation	218
14.3 Exercises	218
14.3.1 Questions	218
14.3.2 Assignments	218
15 Instrument Recognition	219
15.1 Drum Transcription	220
15.2 Exercises	220
15.2.1 Questions	220
15.2.2 Assignments	220
16 Music Performance Assessment	221
16.1 Music Performance	221
16.2 Music Performance Analysis	222
16.3 Approaches to Music Performance Assessment	223
Appendices	225
A Fundamentals	227
A.1 Sampling and Quantization	227
A.1.1 Sampling	227
A.1.2 Quantization	229
A.2 Convolution	231
A.2.1 Identity	231
A.2.2 Commutativity	231
A.2.3 Associativity	232
A.2.4 Distributivity	232
A.2.5 Circularity	233
A.2.6 Simple Filter Examples	233
A.2.6.1 Moving Average Filter	233
A.2.6.2 Single-Pole Low-Pass Filter	235
A.2.7 Zero Phase Filtering with IIRs	235
A.3 Correlation Function	237
A.3.1 Normalization	237
A.3.2 Autocorrelation Function	238
A.3.3 Applications	240
A.3.4 Calculation in the Frequency Domain	240

B Fourier Transform	241
B.1 Properties of the Fourier Transformation	241
B.1.1 Inverse Fourier Transform	241
B.1.2 Superposition	242
B.1.3 Convolution and Multiplication	242
B.1.4 Parseval's Theorem	243
B.1.5 Time and Frequency Shift	243
B.1.6 Symmetry	244
B.1.7 Time and Frequency Scaling	244
B.1.8 Derivatives	245
B.2 Spectrum of Example Time Domain Signals	245
B.2.1 Delta Function	245
B.2.2 Constant	246
B.2.3 Cosine	246
B.2.4 Rectangular Window	246
B.2.5 Delta Pulse	246
B.3 Transformation of Sampled Time Signals	247
B.4 Short Time Fourier Transform of Continuous Signals	247
B.4.1 Window Functions	248
B.4.1.1 Rectangular Window	248
B.4.1.2 Bartlett Window	248
B.4.1.3 Generalized Superposed Cosines	249
B.4.1.4 Generalized Power of Cosine	249
B.5 Discrete Fourier Transform	250
B.5.1 Window Functions	250
B.5.1.1 Discrete Window Properties	251
B.5.2 Fast Fourier Transform	251
B.6 Frequency Reassignment: Instantaneous Frequency	252
C Principal Component Analysis	255
C.1 Computation of the Transformation Matrix	255
C.2 Interpretation of the Transformation Matrix	257
D Linear Regression	259
E Software for Audio Analysis	261
E.1 Software Frameworks and Applications	261
E.1.1 Marsyas	262
E.1.2 CLAM	262
E.1.3 jMIR	262
E.1.4 CoMIRVA	262
E.1.5 Sonic Visualiser	263
E.2 Software Libraries and Toolboxes	263
E.2.1 Feature Extraction	263
E.2.2 Plugin Interfaces	264
E.2.2.1 FEAPI	264
E.2.2.2 VAMP	264
E.2.3 Other Software	264
F Datasets	267
Bibliography	268

List of Figures

2.1	General Audio Content Analysis System	32
3.1	Periodic Audio Signal	35
3.2	Reconstruction of Periodic Signals with Sinusoidals	36
3.3	PDFs of Four Prototype Signals	37
3.4	Laplace Distribution vs. a PDF Estimated from a Music Signal	38
3.5	Arithmetic Mean and Median	39
3.6	Standard Deviation	40
3.7	Visualization of PDF Quantiles	41
3.8	Block Processing: Hop Size and Block Length	42
3.9	Block Processing: Example	43
3.10	to be updated	44
3.11	to be updated	45
3.12	Short-Time Fourier Transform	47
3.13	Spectrogram of a Saxophone Signal	49
3.14	Blocking for the CQT	49
3.15	Log-Mel Spectrogram of a Saxophone Signal	50
3.16	Code example for the Mel spectrogram	51
3.17	Gammatone Filters	51
3.18	Resonance Filterbank	52
3.19	Waveform Shape of Three Signals	53
3.20	Feature Extraction Flow Chart	54
3.21	Spectral Centroid	55
3.22	Implementation of Spectral Centroid	55
3.23	Spectral Spread	57
3.24	Implementation of Spectral Spread	57
3.25	Implementation of Spectral Skewness	58
3.26	Spectral Skewness	59
3.27	Implementation of Spectral Kurtosis	59
3.28	Spectral Kurtosis	60
3.29	Implementation of Spectral Rolloff	60
3.30	Spectral Rolloff	61
3.31	Implementation of Spectral Decrease	61
3.32	Spectral Decrease	62
3.33	Implementation of Spectral Slope	62
3.34	Spectral Slope	62
3.35	MFCC Transformation Basis Functions	64
3.36	MFCCs	65
3.37	Implementation of Spectral MFCCs	65
3.38	MFCC Filterbank	65
3.39	Implementation of Spectral Flux	66

3.40 Spectral Flux	66
3.41 Implementation of Spectral CrestFactor	67
3.42 Spectral CrestFactor	68
3.43 Implementation of Spectral Flatness	68
3.44 Spectral Flatness	69
3.45 Implementation of Spectral TonalPowerRatio	70
3.46 Spectral TonalPowerRatio	70
3.47 Implementation of MaxAcf	72
3.48 Time MaxAcf	73
3.49 Implementation of ZeroCrossingRate.	73
3.50 Time ZeroCrossingRate	74
3.51 Feature Aggregation	77
3.52 to be updated	79
3.53 Feature Selection: Accuracy of different subsets	80
4.1 Overfitting	83
4.2 Classification by Thresholding	84
4.3 Feature Space	85
4.4 K Nearest Neighbor classification	85
4.5 to be updated	85
4.6 Gaussian Mixture Model	86
4.7 Visualization of Linear Regression	88
4.8 Clustering	88
4.9 to be updated	89
4.10 Kmeans clustering	89
5.1 to be updated	93
6.1 ROC and AUC	100
7.1 Harmonics of a Periodic Signal	105
7.2 Non-linear Mapping of Frequency to Pitch	106
7.3 Code example for frequency to Mel conversion	106
7.4 Helix Visualizing Human Pitch Perception	108
7.5 Score representation of Pitches	108
7.6 Piano Keyboard	109
7.7 Musical Intervals in Score Notation	110
7.8 Harmonics in score notation	112
7.9 Pitch Detection Error: Sample Accuracy	113
7.10 Pitch Detection Error: Spectral Bin Accuracy	114
7.11 Interpolation of a time domain signal	114
7.12 Interpolation of a Frequency Domain Signal	115
7.13 Code example for F0 estimation with Zero Crossings	116
7.14 Code example for F0 estimation with ACF	117
7.15 F0 estimation with ACF	117
7.16 Center Clipping	118
7.17 Code example for F0 estimation with AMDF	119
7.18 F0 estimation with AMDF	119
7.19 Spectral compression for HPS	120
7.20 F0 estimation with HPS	121
7.21 Code example for F0 estimation with HPS	121
7.22 F0 estimation with the ACF of the Spectrum	122

7.23	Code example for F0 estimation with ACF of the Spectrum	123
7.24	F0 estimation with the Cepstrum	124
7.25	Half-Wave Rectification in an Auditory Filter Band	125
7.26	Visualization of Eq. (7.37) with example dimensions	127
7.27	Non-negative Matrix factorization example	128
7.28	to be updated	129
7.29	Tuning Frequency Distribution	132
7.30	Tuning Frequency Adaptation	133
7.31	Musical Modes	134
7.32	Major Keys	135
7.33	Circle of Fifths	136
7.34	Pitch Chroma Entry Calculation	137
7.35	Pitch Chroma Example	138
7.36	Code example for Pitch Chroma computation	138
7.37	Pitch Chroma with Harmonics	140
7.38	Key Profiles	142
7.39	Chords	145
7.40	Chord Inversion	145
7.41	Chord Detection Flowchart	145
7.42	Chord Template Matrix	146
7.43	to be updated	147
7.44	to be updated	148
7.45	Chord Detection	149
8.1	Level Measurement Error	156
8.2	Implementation of Rms.	157
8.3	RMS	158
8.4	Flowchart of Weighted RMS Calculation	158
8.5	Frequency Weighting Transfer Functions	159
8.6	Flowchart of a Peak Program Meter	159
8.7	Implementation of PeakEnvelope.	160
8.8	Peak Envelope	161
8.9	Flowchart of the Zwicker Loudness	161
9.1	Attack Time and Onset Time	163
9.2	Hierarchy of Metrical Levels	166
9.3	Time Signatures	167
9.4	Note Values and Rest Values	167
9.5	Onset Detection Flowchart	168
9.6	Implementation of a Flux-based Novelty Function	169
9.7	Implementation of Novelty-Function Smoothing	170
9.8	Peak Picking in the Novelty Function	171
9.9	Beat Histogram	173
9.10	Onsets and Beats	174
9.11	Beat Tracking	175
9.12	Structure Example for a Popular Song	178
9.13	Self-Similarity Matrix	180
9.14	Self-Similarity Matrix based on different Features	181
9.15	Checker Board Filter Kernel	181
9.16	Novelty Function extracted from the SSM	182
9.17	Self Similarity Matrix and Low-pass filtered Diagonal	183
9.18	Rotated Self Similarity Matrix	183

10.1 Alignment of two Sequences	187
10.2 Dynamic Time Warping Path	188
10.3 DTW Distance Matrix and Cost Matrix	190
10.4 Code example for DTW	191
10.5 DTW Performance Optimizations	193
11.1 General Fingerprinting System	202
11.2 Philips Subfingerprint Extraction	204
11.3 Audio Fingerprint Examples	205
11.4 to be updated	205
12.1 Genre Taxonomy Examples	208
12.2 Feature Space	210
14.1 Russel's Mood Model	216
16.1 Musical Communication	221
A.1 Sampling of a Continuous Signal	228
A.2 Sampling Ambiguity	228
A.3 Quantization of a Signal	229
A.4 Characteristic Line of a Quantizer	230
A.5 Low-Pass Filter Magnitude Responses	234
A.6 Visualization of Zero Phase Filtering	236
A.7 Example: ACF of a Sinusoid and White Noise	239
B.1 Sample Theorem and Aliasing	247
B.2 Windows in Time and Frequency Domain	248
B.3 add caption	252
B.4 Instantaneous Frequency Example	253
C.1 PCA Axes Rotation	256
C.2 PCA Example	256

List of Tables

3.1	MFCC Implementations Comparison Table	64
6.1	Confusion Matrix. Ground Truth (GT) refers to the ground truth annotations <i>improve visuals</i>	99
7.1	Pitch Class Names and Indices	108
7.2	Musical Intervals	109
7.3	Deviations of Different Temperaments from Equal Temperament	111
7.4	STFT Frequency Resolution for Different Block Lengths	113
7.5	Tuning Frequency Deviation	132
7.6	Deviation of Harmonics from Equal Temperament	140
7.7	Rearranged Pitch Chroma	141
7.8	Key Profile Templates	143
9.1	Structure Annotations	184
11.1	Comparison of Fingerprinting and Watermarking	201
14.1	Mood Clusters (Schubert)	217
14.2	Mood Clusters (MIREX)	217
B.1	Fourier Window Properties	251

Acronyms

ACA	Audio Content Analysis
ACF	Autocorrelation Function
AMDF	Average Magnitude Difference Function
ANN	Artificial Neural Network
AOT	Acoustic Onset Time
API	Application Programmer's Interface
AUC	Area Under Curve
BPM	Beats per Minute
CAMEL	Content-based Audio and Music Extraction Library
CASA	Computational Auditory Scene Analysis
CCF	Cross Correlation Function
CCIR	Comité Consultatif International des Radiocommunications
CD	Compact Disc
CiCF	Circular Correlation Function
CLAM	C++ Framework for Audio and Music
CNN	Convolutional Neural Network
COG	Center of Gravity
CQT	Constant Q Transform
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DJ	Disk Jockey
DNN	Deep Neural Network
DP	Dynamic Programming
DSP	Digital Signal Processing
DTW	Dynamic Time Warping
EBU	European Broadcasting Union
EM	Expectation Maximization
ERB	Equivalent Rectangular Bandwidth
FEAPI	Feature Extraction Application Programmer's Interface
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
FN	False Negative
FNR	False Negative Rate
FP	False Positive
FPR	False Positive Rate

FT	Fourier Transform
FWR	Full-Wave Rectification
GMM	Gaussian Mixture Model
GT	Ground Truth
HFC	High Frequency Content
HMM	Hidden Markov Model
HPS	Harmonic Product Spectrum
HSS	Harmonic Sum Spectrum
HTK	HMM Toolkit
HWR	Half-Wave Rectification
IBI	Inter-Beat Interval
ICA	Independent Component Analysis
IDFT	Inverse Discrete Fourier Transform
IFT	Inverse Fourier Transform
IIR	Infinite Impulse Response
IO	Input/Output
IOI	Inter-Onset Interval
ITU	International Telecommunication Union
JNDL	Just Noticeable Difference in Level
KNN	K-Nearest Neighbor
LDA	Linear Discriminant Analysis
MA	Moving Average
MAE	Mean Absolute Error
MFCC	Mel Frequency Cepstral Coefficient
MIDI	Musical Instrument Digital Interface
MIR	Music Information Retrieval
MIREX	Music Information Retrieval Evaluation eXchange
ML	Machine Learning
MP3	MPEG-1 Layer 3
MPA	Music Performance Analysis
MPEG	Motion Picture Experts Group
MSE	Mean Squared Error
NMF	Non-negative Matrix Factorization
NN	Nearest Neighbor
NOT	Note Onset Time
PAT	Perceptual Attack Time
PCA	Principal Component Analysis
PDF	Probability Density Function
POT	Perceptual Onset Time
PPM	Peak Program Meter
PSD	Peak Structure Distance

RFD	Relative Frequency Distribution
RLB	Revised Low Frequency B Curve
RMS	Root Mean Square
RNN	Recurrent Neural Network
ROC	Receiver Operating Curve
SIMD	Single Instruction Multiple Data
SNR	Signal-to-Noise Ratio
SOM	Self-Organizing Map
SSM	Self Similarity Matrix
STFT	Short Time Fourier Transform
SVD	Singular Value Decomposition
SVM	Support Vector Machine
SVR	Support Vector Regression
TN	True Negative
TNR	True Negative Rate
TP	True Positive
TPR	True Positive Rate
WEKA	Waikato Environment for Knowledge Analysis
YAAFE	Yet Another Audio Feature Extractor

Chapter 1

Introduction

Audio is an integral and ubiquitous aspect of our daily lives; we intentionally produce sound (e.g., when communicating through speech or play an instrument), we actively listen (e.g., to music or podcasts), and we try to ignore other sound sources (e.g., traffic noise). Its importance in the media world is highlighted by the fact that numerous audio formats exist without video, while video without audio is rare. **Possibly remove first paragraph.**

Audio signals contain a wealth of information: by simply listening to an audio signal, humans are able to infer a variety of content information. A speech signal, for example, obviously transports the textual information, but it also might reveal information about the speaker (gender, age, accent, mood, etc.), the recording environment (e.g., indoors vs. outdoors) and much more. A music signal might allow us to extract melodic and harmonic characteristics, understand the musical structure, identify the instruments playing, perceive the projected emotion, categorize the music genre, and assess characteristics of the performance as well as the proficiency of the performers. An audio signal can contain and transport a wide variety of content beyond these simple examples. This content information is sometimes referred to as *meta data*: data about (audio) data.

The field of *Audio Content Analysis (ACA)* aims at designing and applying algorithms for the automatic extraction of content information from the raw (digital) audio signal. This enables content-driven and content-adaptive services which describe, categorize, sort, retrieve, segment, process, and visualize the signal and its content.

The wide range of possible audio sources and the multi-faceted nature of audio signals results in variety of distinct ACA problems, leading to various areas of research, including:

- *speech analysis*, covering topics such as automatic speech recognition [HBR14, YD15] or recognizing emotion in speech [DPW96, EAKK11],
- *urban sound analysis* with applications in noise pollution monitoring [BMS18] and audio surveillance, i.e., the detection of dangerous events [CCTM16],
- *industrial sound analysis* such as monitoring the state of mechanical devices like engines [GALL19] or monitoring the health of livestock [BHB⁺15], and, last but not least,
- *musical audio analysis*, targeting the understanding and extraction of musical parameters and properties from the audio signal [Ell06].

In this book, we will focus on the analysis of musical audio signals and understand the abbreviation ACA as applied only to music signals. There are many similarities and parallels to the areas above, but there exist also many differences that distinguish musical audio from other signals beyond simple technical properties such as audio bandwidth. Like an urban sound signal, music is a poly-timbral mixture of multiple sound sources, but unlike urban sound, its sound sources are clearly related (e.g., melodically, harmonically, or rhythmically). Like a speech signal, a music signal is a sequence in a constrained language, but unlike speech, the musical language is abstract and has no singular meaning. Like an industrial audio signal, music has both tonal and noise-like components, but unlike the industrial signal, it conveys a (musical) form based on hierarchical grouping of elements, repetition, and variation of rhythmic, dynamic, tonal, and timbral elements.

As we will see throughout the chapters, the analysis of musical audio often requires multi-disciplinary understanding. While we approach this topic mostly from an engineering and Digital Signal Processing (DSP) perspective, the proper formulation of research questions and task definitions often require methods or at least synthesis of knowledge from fields as diverse as music theory, music perception, and psycho-acoustics. Researchers working on ACA thus come from various backgrounds such as computer science, engineering, psychology, and musicology.

The diversity in musical audio analysis is also exemplified by the wide variety of terms referring to it. Overall, ACA is situated in the broader areas of *music informatics* and *Music Information Retrieval (MIR)*. MIR is a broader field that covers not only the analysis of musical audio, but also symbolic non-audio music formats such as musical scores and files or signals compliant to the so-called *Musical Instrument Digital Interface (MIDI)* protocol [MID01]. MIR also covers the analysis and retrieval of information that is music-related but cannot be (easily) extracted from the audio signal such as the song lyrics, user ratings, performance instructions in the score, or bibliographical information such as publisher, publishing date, the work's title, etc. Other areas of research, such as music source separation and automatic music generation are sometimes also considered to belong within MIR. Various overview articles clarify how the understanding of the field of MIR has evolved over time [Dow03, Ori06, CVG⁺08, SGU14, Ler14, BFD15]. Other, related terms are also in use. Audio event detection, nowadays often related to urban sound analysis, is sometimes described as computational analysis of sound scenes [VPE18]. The analysis of sound scenes from a perceptual point of view has been described as *Computational Auditory Scene Analysis (CASA)* [WB06]. In the past, other terms have been used more or less synonymously to the term audio content analysis. Examples of such synonyms are *machine listening* and *computer audition*.

1.1 A Short History of Audio Content Analysis

Historically, the first systems analyzing the content of audio signals appeared shortly after technology provided the means of storing and reproducing recordings on media in the 20th century. One early example is Seashore's Tonoscope, which allowed one to analyze the pitch of an audio signal by visualizing the fundamental frequency of the incoming audio signal on a rotating drum [Sea02]. However, the evolution of digital storage media, digital signal processing methods, and machine learning during the last decades, along with the growing amount of digital audio data available through downloads and streaming services, has significantly increased both the need and the possibilities of automatic systems for analyzing audio content, resulting in a lively and growing research field.

Early systems for audio analysis were frequently so-called expert systems [LG91], designed by experts who implement their task-specific knowledge into a set of rules. Such systems can be very successful if there is a clear and simple relation between the knowledge and the implemented algorithms. A good example for such systems are some of the pitch tracking approaches introduced in Sect. 7.3: as the goal is the detection of periodicity in the signal in a specific range, an approach such as the Autocorrelation Function (ACF), combined with multiple assumptions and constraints, can estimate this periodicity and thus the fundamental frequency.

Later, data-driven systems became increasingly popular and traditional machine learning approaches started to show superior performance on many tasks. These systems extract so-called features from the audio to achieve a task-dependent representation of the signal. Then, training data is used to build a model of the feature space and how it maps to the inferred outcome. The role of the expert becomes less influential in the design of these systems, as they are restricted to select or design a fitting set of features and choose the machine learning approach. An example for these systems is the genre classification approach introduced in Sect. 12.

Modern machine learning approaches include trainable feature extraction approaches (also referred to as feature learning, see Sect. 3.6.14) as they are part of deep neural networks. These approaches have consistently shown superior performance for nearly all ACA tasks. The researcher seldom imparts much domain knowledge beyond choosing input representation and system architecture of these systems. An example of such an end-to-end system could be a music genre classification system based on a convolutional architecture with mel-spectrogram input.

It should be pointed out that, while modern systems tend to have superior performance, they also tend

to be less interpretable and transparent. For example, deducing the reason for a false classification result in a network-based system might be difficult while the reason should be more easily identifiable in a rule-based system.

1.2 Applications and Use Cases

The content extracted from music signals improves or enables various forms of content-based and content-adaptive services, which allow to sort, categorize, find, segment, process, and visualize the audio signal based on its content.

1.2.1 Music Browsing and Music Discovery

One of the most obvious applications is content-based search for music in large databases for which manual annotation by humans is often infeasible. This allows, for example, Disk Jockeys (DJs) to retrieve songs in a specific tempo or key, or listeners find songs in a specific genre or mood. The same information can be used in end consumer applications such as audio-based music recommendation and playlist generation systems using an in-depth understanding of the musical content [KSG19].

A content-based annotation and representation of the audio signal can also be used to design new interfaces to access data. Fingerprinting allows to identify a song currently playing from a large database of songs, while Query-by-humming systems identify songs through a user-hummed melody. Content can also be utilized for new ways of sound visualization and user interaction; a user could, for example, navigate a virtual music similarity space.

1.2.2 Music Consumption

Audio analysis has already started to transform consumer-facing industries such as streaming services as mentioned above. So far, most services focus on recommending music to be played back, but in the near future, we can expect the rise of creative music listening applications that enable the listener to interact not only by choosing content but interact with the content itself. This could include, for example, the gain adjustment for individual voices, replacing instruments or vocalists, or interactively changing the musical arrangement.

1.2.3 Music Production

Knowledge of the audio content can improve music production tools in various dimensions. On the one hand, content information can enable a more “musical” software interface, e.g., by displaying score-like information synchronized with the audio data, and thus enabling an intuitive approach to editing the audio data. On the other hand, production software usage could be enhanced in terms of productivity and efficiency: the better a software understands the details of incoming audio streams or files, the better it can adapt, for instance, by applying default gain and equalization parameters [RB18] or suggest compatible recordings from a library. Systems might support editors by automatic artifact-free splicing of multiple recordings from one session or selecting error-free recordings from a set of recordings.

Modern tools also enhance the creative possibilities in the production process. For example, creating harmonically meaningful background choirs by analyzing the lead vocals and the harmony track is already technically feasible nowadays. Knowing and isolating sound sources in a recording could enable new ways of modifying or morphing different sounds to create new soundscapes, effects, and auditory scenes.

1.2.4 Music Education

The potential of utilizing technology to assist music (instrument) education has been recognized as early as the 1930s, when Seashore pointed out the educational value of scientific observation of music performances [Sea38]. The availability of automatic and fast audio analysis helps the creation of artificially intelligent music tutoring software, which aims at supplementing teachers by providing students with insights and interactive

feedback by analyzing and assessing the audio of practice sessions. An interactive music tutor can highlight problematic parts of the students' performance, provide a concise yet easily understandable analysis, give specific and understandable feedback on how to improve, and individualize the curriculum depending on the students' mistakes and general progress. At the same time, music tutoring software can provide an accessible, objective, and reproducible analysis.

1.2.5 Generative Music

Machine interpretable content information can also feed generative algorithms. The automatic composition and rendition of music is emerging as a challenging yet popular research direction [BHP20], gaining interest from both research institutions and industry. While bigger questions concerning capabilities and restrictions of computational creativity as well as aesthetic evaluation of algorithmically generated music remain largely unanswered, practical applications such as generating background music for user videos and commercial advertisements are currently in the focus of many researchers. The interactive and adaptive generation of sound tracks for video games as well as individualized generation of license-free music content for streaming are additional long-term goals of considerable commercial interest.

1.3 Structural Choices and Information

The goal of this text is to introduce the wide variety of tasks in audio content analysis. For each of these tasks, simple baseline systems are presented that —while not solving most of the tasks— allow insights into the specific challenges and problems. The variety of approaches mirrors the variety of content we can find in audio signals, and code snippets are listed wherever easily possible. Last but not least, evaluation methodologies are summarized for most tasks; while evaluation often seems like a chore and uninteresting part of designing a system, it is a crucial part: without evaluation no-one knows how well a system works. Most chapters conclude with a set of questions and assignments assessing the learning outcomes. As this text intends to provide a starting point and guidance for the design of novel musical audio content analysis systems, every chapter contains many references to relevant work.

This book is structured into three main parts plus an appendix. The first part covers fundamentals shared by many audio analysis systems. It starts with a closer look at musical audio content and a general flow-chart of an audio content analysis system. After a quick coverage of signals and general pre-processing concepts, typical time-frequency representations are introduced, which nowadays serve as the input representations of many analysis systems. A closer look at the most commonly used instantaneous or low-level features is then following by an introduction of feature post-processing methods covering normalization, aggregation, as well as general dimensionality reduction methods.

The second part introduces systems that transcribe individual musical properties of the audio signal. These are mostly properties that are more or less explicitly defined in a musical score, such as pitch, dynamics, and tempo. More specifically, these approaches are grouped into the content dimensions *tonal* (fundamental frequency, tuning frequency, musical key, chords), *intensity* (level, loudness), and *temporal analysis* (onsets, beats, tempo, meter, structure).

Audio analysis approaches related to classification and identification are presented in part three. Many of these systems do not focus on individual musical content dimensions but aim at extracting high-level content from the audio signal. Thus, the classification of musical genre and mood, as well as assessment of student music performances are introduced. In addition audio fingerprinting is discussed as one of the audio analysis technologies arguably most impactful for the consumer in the past.

The appendix provides more in-depth reference to basic concepts such as audio sampling and quantization, the Fourier transform, introduces correlation and convolution, as well as providing quick references to principal component analysis and linear regression. [add dataset chapter?](#)

Part I

Fundamentals of Audio Content Analysis

Chapter 2

The Audio Content Analysis Process

IS THERE A BETTER TITLE?

Taking a closer look at ACA, we first want to discuss the content to be extracted from an audio signal and then introduce the basic processing steps of an ACA system.

2.1 Audio Content

The introduction already provided some examples for content that can be extracted from an audio signal. For a music recording, content originates from three different sources:

- *Composition*: The term composition will be used broadly as a definition of musical ideas.¹ It can refer to any form of notating or remembering music from the *basso continuo* (a historic way of defining the harmonic structure) and the classic Western score notation to the lead sheet and other forms of notation used for contemporary and popular music.

The content related to the composition allows us to recognize different renderings of the same song or symphony as being the same piece. In most genres of western music, this encompasses musical elements such as melody, harmony, instrumentation, structure and form, and rhythm.

- *Performance*: Music generally requires a performer or group of performers to create a unique acoustical rendition of the composition. The performance communicates the explicit information from the composition, but also interprets and modifies the information it.

This happens through the performance-related content, which includes, for example, the tempo and its variation as well as the micro-timing, the realization of musical dynamics, accents, and instantaneous dynamic modulations such as tremolo (see Sect. 8.2), the use usage of expressive intonation and vibrato (see Sect. 7.2.3.2), and specific playing (e.g., bowing) techniques influencing the sound quality.

- *Production*: As the input of an ACA system is an audio recording, the choices made during recording and production impact certain characteristics of the recording.

Production-related content mostly impacts the sound quality of the recording (microphone positioning, equalization, and the application of effects such as reverb to the signal) and the dynamics (by applying manual or automatic gain adjustments). However, changes in timing and pitch may occur as well during the process of editing the recording and applying software for pitch correction.

Note that these content sources can be hard to separate from the final audio recording. For example, the timbre of a recording can be determined by the instrumentation indicated by the score, by the specific choice of instruments (e.g., historical instruments, specific guitar amps, etc.), by specific playing techniques, and by sound processing choices made by the sound engineer or producer.

¹In Sect. 16 we will also refer to it as a *blueprint* for a performance.

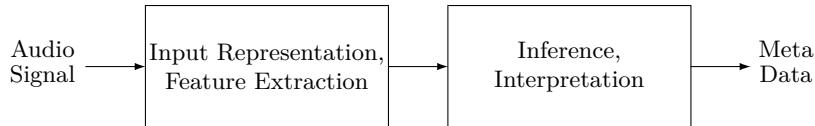


Figure 2.1: General processing stages of a system for audio content analysis

From a practical audio analysis point of view, the categorization by content source has only limited use. A more useful categorization of musical content is driven by musical considerations driven by perception, leading to four main categories:

- *tonal characteristics*: pitch-related information such as melody, harmony, key, intonation, etc. (see Sect. 7),
- *timbre characteristics*: information related to sound quality such as spectral shape, etc. (see Sect. 3.6),
- *intensity-related characteristics*: dynamics-related information such as envelope-, level-, and loudness (see Sect. 8), and
- *temporal characteristics*: information comprising the tempo, timing, meter, etc. of the signal (see Sect. 9).

Obviously, there are additional ways to describe audio signals that do not fall squarely in one of these music-driven categories, for example, statistical or other technical descriptions of the audio data such as an amplitude distribution.

The categories listed above cover the basic musical information we can find in a music signal, and tasks such as tempo detection, key detection, or envelope extraction fall unambiguously into one of the above categories. However, cues from various categories are necessary to deduce, for example, the structure of a musical piece or to extract performance characteristics. Furthermore, there exist many high-level perceptual concepts that humans use when they categorize music, such as the music genre or the projected affective content. While such concepts are hard to define, it is clear that they combine content from all categories mentioned above.

Note that there is also content that is not easily extractable from the signal itself but may still be useful for MIR systems. This information might include the year of the composition or recording, the record label, the song title, information on the artists, etc.

2.2 Audio Content Analysis Flow-Chart

Systems for audio content analysis can often be represented as a two-step process as depicted in Fig. 2.1, where the signal is first converted into a meaningful input representation that is then fed into a data-driven system for inference such as a classifier. If we have a perfect input representation, inference can be as simple as just applying a threshold to the representation. If we have a very powerful classifier, the input representation does not need to be too refined to yield good results. In traditional systems, the design of the input representation used to require considerable effort in feature engineering. Nowadays, the input processing tends to become less sophisticated (e.g., simply computing a spectrogram), which is counterbalanced by the increasing complexity of the inference engine.

For traditional feature-based approaches, the following properties should generally be desirable for the feature input:

- *Compact and non-redundant*: the dimensionality of raw audio data tends to be too high for traditional machine learning approaches. One channel of a digital audio file in Compact Disc (CD) quality (44,100 samples per second, 16 bits per sample²) with a length of 5 minutes contains

$$5 \text{ min} \cdot 60 \text{ s/min} \cdot 44100 \text{ samples/s} \cdot 16 \text{ bits/sample} = 211,680,000 \text{ bits.} \quad (2.1)$$

A feature aims at representing these data with fewer values.

²Note that an audio sample is usually represented with a 32 bit float in music processing and analysis software

- *Task-relevant without noise or other task-irrelevant information:* features necessarily have to represent relevant information. A tempo detection system, for example, might need to focus on the rhythmic parts instead of pitch-related information. Therefore, the features should attempt to capture all task-relevant information and discard all irrelevant information. Note that capturing task-relevant information does not necessarily mean that such features are humanly interpretable; rather, it is sufficient if they can be properly interpreted by the inference system.
- *Easy to analyze:* Although all information is contained in the raw audio data, observing meaningful information directly from the audio data is complicated. For example, a spectrogram contains a similar amount of data as a time domain signal, but it is often easier to parse due to the more sparse representation.

While these qualities themselves continue to be desirable for modern approaches based on neural networks, the features are not explicitly designed but rather learned from data. Some argue that end-to-end systems without input processing can be generally preferable [PNP⁺18], however, this statement is debatable [Gla17]. It should be noted that the terms input representation and feature representation are often used inconsistently. For example, representation learning aims at extracting a compressed feature representation from data that can be potentially used for a multitude of related tasks. This representation might be the activation at one of the deeper layers of a neural network; these activations might then be referred to as features which might in turn be fed into another network.

In a way, this ambiguity may in some way relate to the various abstraction levels between the raw audio data and how humans refer to music. While extracting the tempo might be the output of one system, another system might use this tempo just as one feature amongst many others to, for example, automatically recognize the musical style.

The second processing stage of an ACA system infers the desired information from the feature data. Thus, it aims at mapping the potentially abstract feature information into a domain both usable and comprehensible by humans. This process is often accomplished with a data-driven machine learning system, but might also be a rule-based expert system.

2.3 Exercises

2.3.1 Questions

1. Describe the difference between score-inherent and performance-inherent musical content in the context of Western classical music.
2. List 10 examples of content in a musical audio signal. Assign one (or more) of the four basic content categories to each of your examples.

Chapter 3

Input Representation

This chapter introduces the computation of typical input representations of an ACA system. First, it introduces some fundamental properties of audio signals and their description and outlines typical pre-processing steps such as blocking and down-mixing. Then, time-frequency representations of the audio signal are introduced. Various traditional instantaneous features are then introduced, followed by common ways of post-processing the extracted features.

3.1 Audio Signals

An *audio signal* as humans perceive it can be described as a function of time-variant sound pressure level. A microphone can be used to convert the sound pressure level into voltage. The signal is defined for all times t and is therefore a (time-) continuous signal $x(t)$.

3.1.1 Periodic Signals

A *periodic signal* repeats itself in constant time intervals. Its periodicity can be formulated by

$$x(t) = x(t + T_0) \quad (3.1)$$

with T_0 being the periodicity interval, or, in other words, the *period length* of the first of the harmonics. Figure 3.1 shows a periodic audio signal and highlights its fundamental period length T_0 .

The *fundamental frequency* of this periodic signal is then

$$f_0 = \frac{1}{T_0}. \quad (3.2)$$

The frequency of other tonal components, the remaining harmonics, is always an integer multiple of the frequency of the first harmonic.

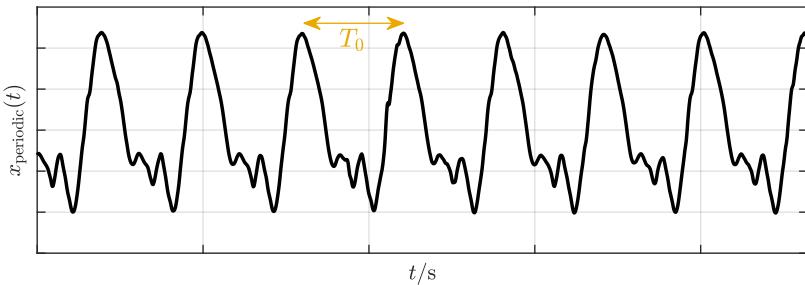


Fig. Gen.: `plotPeriodic.m`

Figure 3.1: Snippet of a periodic audio signal with indication of its fundamental period length.

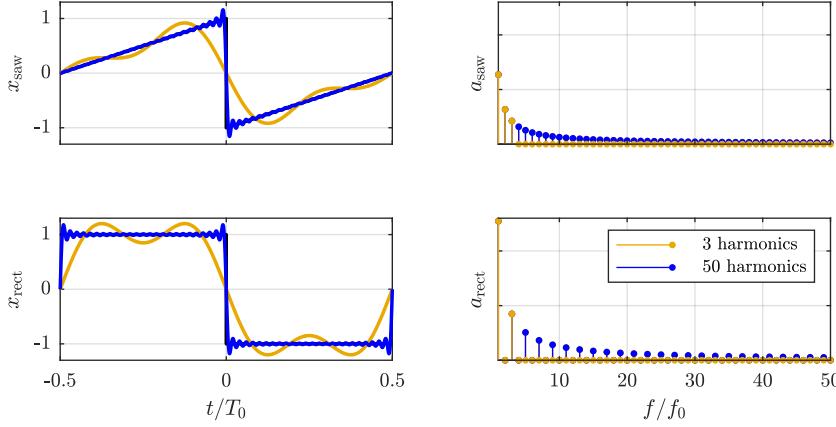


Fig. Gen.: plotFourierSeries.m

Figure 3.2: Approximation of periodic signals: sawtooth (top) and square reconstructed with 3 and 25 sinusoidal harmonics.

Every periodic signal $x(t)$ can be represented by a superposition of weighted sinusoidal signals, the *Fourier series*

$$x(t) = \sum_{k=-\infty}^{\infty} a(k) e^{j\omega_0 k t}. \quad (3.3)$$

The periodicity, or the fundamental frequency, of the signal is represented as angular frequency $\omega_0 = 2\pi f_0$. The real part of $e^{j\omega_0 k t}$ represents the cosine components (even) and its imaginary part the sine components (odd) of the signal: $e^{j\omega t} = \cos(\omega t) + j \sin(\omega t)$. The coefficient $a(k)$ is the weight of the k th harmonic and can be calculated by

$$a(k) = \frac{1}{T_0} \int_{-T_0/2}^{T_0/2} x(t) e^{-j\omega_0 k t} dt. \quad (3.4)$$

If the number of frequencies used to represent the signal is limited

$$\hat{x}(t) = \sum_{k=-\mathcal{O}}^{\mathcal{O}} a(k) e^{j\omega_0 k t}, \quad (3.5)$$

then some periodic signals may be constructed only imperfectly. The larger the order \mathcal{O} , the higher is the highest frequency included and the better is the representation of the signal.

Periodic signals featuring “sudden changes” such as discontinuities of the waveform require higher order Fourier coefficients to model the waveform sufficiently well. The modeling of discontinuities and so-called transients requires high frequencies. More systematically we can say that the higher the order of derivations of the signal that are monotone is the lower the required order will be to represent the signal sufficiently well. The most extreme example is a sinusoidal signal which has an infinite number of monotone derivatives and for which only one coefficient $a(1)$ is required to represent the signal perfectly.

Figure 3.2 shows one period of two periodic signals, a *sawtooth* (top) and a *square wave* (bottom), and their representation with the model orders 3, 25, and unlimited. The coefficients $a(k)$ for these two signals are

$$a_{\text{saw}}(k) = \frac{2}{\pi \cdot k}, \quad (3.6)$$

$$a_{\text{rect}}(k) = \frac{4}{\pi \cdot (2k - 1)}. \quad (3.7)$$

Due to the discontinuities of these two signals there are model errors in the form of overshoots around the point of discontinuity. It can be observed that the frequency of the overshoots increases and the duration of the overshoots decreases with increasing model order. However, the amplitude of the overshoots stays constant unless the order is infinite. This is called *Gibbs’ phenomenon* [Gib98, Gib99].

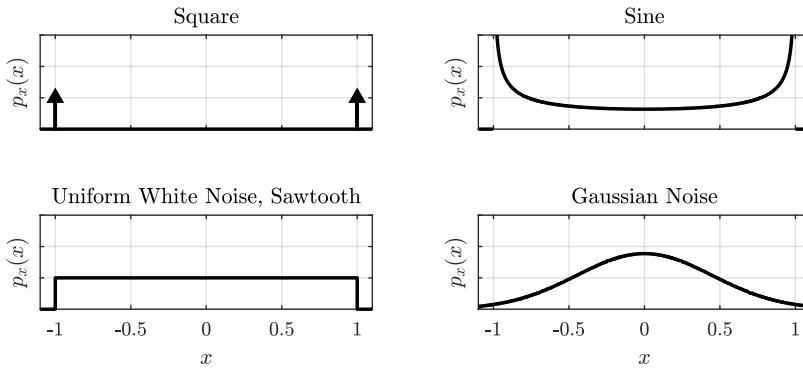
Fig. Gen.: [plotPdfeExamples.m](#)

Figure 3.3: Probability density function of a square wave (top left), a sinusoidal (top right), uniform white noise or a sawtooth wave (bottom left), and Gaussian noise (bottom right).

3.1.2 Random Signals

In contrast to periodic signals, future values of *random signals* cannot be predicted no matter how long the signal has been observed. That means that there exists no fundamental frequency for this type of signal. Every observation of such a signal is therefore only an example of an unlimited number of possible incarnations of the signal. A complete signal description is theoretically only possible with an unlimited number of observations.

An important subcategory of random signals is built of *stationary* signals for which basic properties (such as arithmetic mean and higher central moments, see below) do not change over time. *White noise* is a typical example of a stationary random signal.

3.1.3 Statistical Signal Description

In contrast to sinusoidal signals, noise cannot be described in the time domain with analytical functions. Statistics can help to identify some properties that describe the signal when neglecting its evolution in time. One of the most common representations is the *Probability Density Function (PDF)*, which is a useful tool to describe stationary processes or signals that have the same properties for any point in time t .

The PDF $p_x(x)$ of a signal is the probability distribution of a signal. The x-axis of a PDF plot represents all possible (amplitude) values of the signal x and their probability density is plotted on the y-axis. For example, a rectangular PDF means that all possible signal amplitudes occur with the same probability. The properties of the PDF are

$$p_x(x) \geq 0, \text{ and} \quad (3.8)$$

$$\int_{-\infty}^{\infty} p_x(x) dx = 1. \quad (3.9)$$

A set of prototypical PDFs is shown in Fig. 3.3. For a full-scale square wave, the PDF has only two peaks at maximum and minimum amplitudes while other amplitude values do not exist. A sine wave has a PDF in which values similar to the arithmetic mean (here: 0) are less frequent than values far from the mean. A uniform PDF can be found with noise generated with the `rand` function of different programming languages; a sawtooth has such a uniform PDF as well. A *Gaussian distribution*

$$p_g = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x - \mu_x)^2}{2\sigma^2}\right) \quad (3.10)$$

is assumed to be a typical distribution of “real-world” noise; a typical music signal has in many cases a distribution shaped similar to a *Laplace distribution* with a prominent peak at 0 amplitude (compare Fig. 3.4). The Laplace distribution is given by

$$p_l = \frac{1}{2\beta} \exp\left(-\frac{|x - \mu_x|}{\beta}\right). \quad (3.11)$$

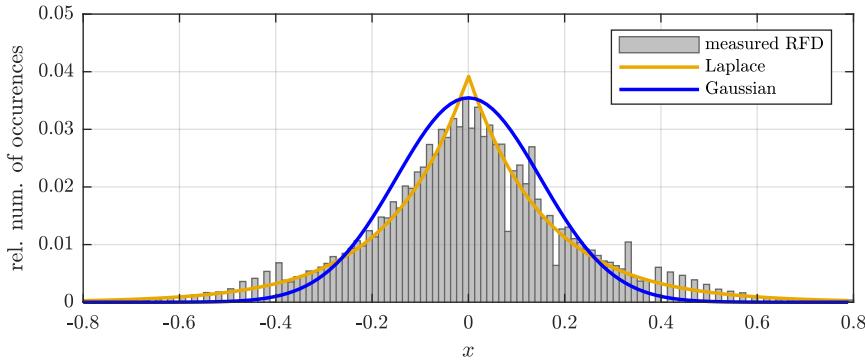
Fig. Gen.: `plotRfd.m`

Figure 3.4: Distribution function estimated from a music signal compared to a Laplace and a Gaussian distribution.

If the input signal is a constant, for example, a DC offset, the PDF consists only of a single peak.

Comparing Figs. 3.3 (top right) and 3.4 makes it obvious that under the assumption that a PDF describes a signal sufficiently well, a sine wave is not a very good approximation of a real-world music or speech signal.

The PDF of quantized input signals has a limited set of amplitude classes as a quantized signal has only a limited set of amplitude values.

The PDF of a signal can be estimated from a sufficiently long block of samples by computing a histogram of signal amplitudes and dividing it by the number of observed samples. It is then sometimes referred to as *Relative Frequency Distribution (RFD)*. For the sake of simplicity the PDF and the RFD will not be differentiated in the following.

Describing a signal statistically does not only make sense for audio signals, but we can learn also important properties of other signals such as a series of features. Often, we attempt to describe signal distribution with a very reduced set of numbers, which will be introduced given an input signal $v(n)$ of length \mathcal{N} with an RFD $p_v(v)$.

3.1.3.1 Arithmetic Mean

The *arithmetic mean* is the average of the input signal (block). It is computed from $v(n)$ or $p_v(v)$ by

$$\mu_v = \frac{1}{\mathcal{N}} \sum_0^{\mathcal{N}-1} v(n) \quad (3.12)$$

$$\mu_v = \sum_{v=-\infty}^{\infty} v \cdot p_v(v) \quad (3.13)$$

The result of the arithmetic mean is a value between the minimum and maximum input signal value. The unit corresponds to the unit of the input signal. For symmetric PDFs, the arithmetic mean will be the (x-axis) position of the axis of symmetry. For example, a time domain audio signal usually has a mean value of approximately 0; if the signal has a DC offset, the mean will indicate the amount of the DC offset. The calculation of the arithmetic mean is only of limited use if the PDF is not symmetric, compare Fig. 3.5. In this case, the computation of the median (see Sect. 3.1.3.5) of the PDF might be more meaningful measures of the average.

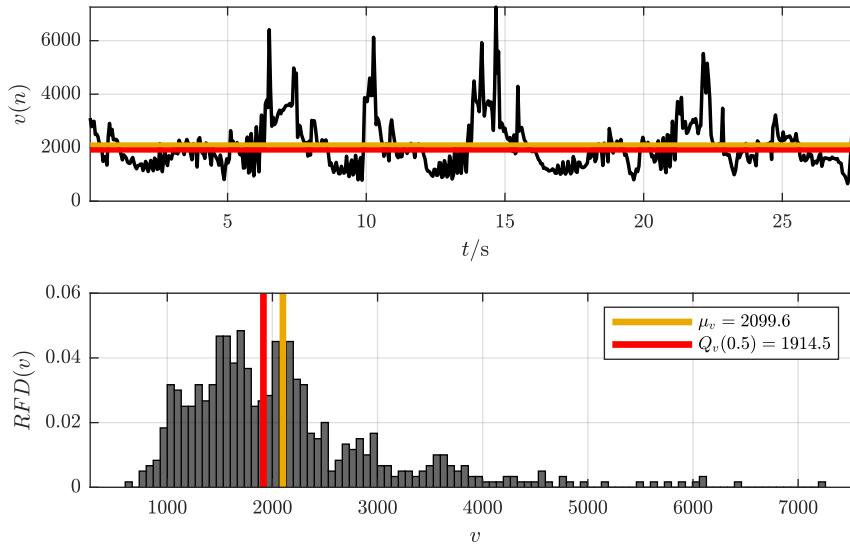
Fig. Gen.: [plotArithmeticMean.m](#)

Figure 3.5: Arithmetic mean and Median of a series of feature values. Top: time series, Bottom: distribution.

3.1.3.2 Geometric Mean

The *geometric mean* is an average measure for sets of positive numbers that are ordered on a logarithmic scale. It can be computed with

$$Mg_v = \sqrt[N]{\prod_0^{N-1} v(n)} \quad (3.14)$$

$$= \exp\left(\frac{1}{N} \sum_0^{N-1} \log(v(n))\right). \quad (3.15)$$

Equation (3.15) is equivalent to Eq. (3.14) but avoids problems with computational accuracy for long blocks of data and large values at the computational cost of applying a logarithm to each signal value. The result of the geometric mean is a value between the minimum and maximum input signal value. The unit corresponds to the unit of the input signal.

3.1.3.3 Harmonic Mean

The *harmonic mean* is a measure appropriate for averaging rates. It is

$$Mh_v = \frac{N}{\sum_0^{N-1} 1/v(n)}. \quad (3.16)$$

3.1.3.4 Variance and Standard Deviation

Both the *variance* and the *standard deviation* measure the spread of the input signal \$v(n)\$ around its arithmetic mean. The variance \$\sigma_v^2\$ is defined by

$$\sigma_v^2 = \frac{1}{N} \sum_0^{N-1} (v(n) - \mu_v)^2 \quad (3.17)$$

$$\sigma_v^2 = \sum_{v=-\infty}^{\infty} (v - \mu_v)^2 \cdot p_v(v) \quad (3.18)$$

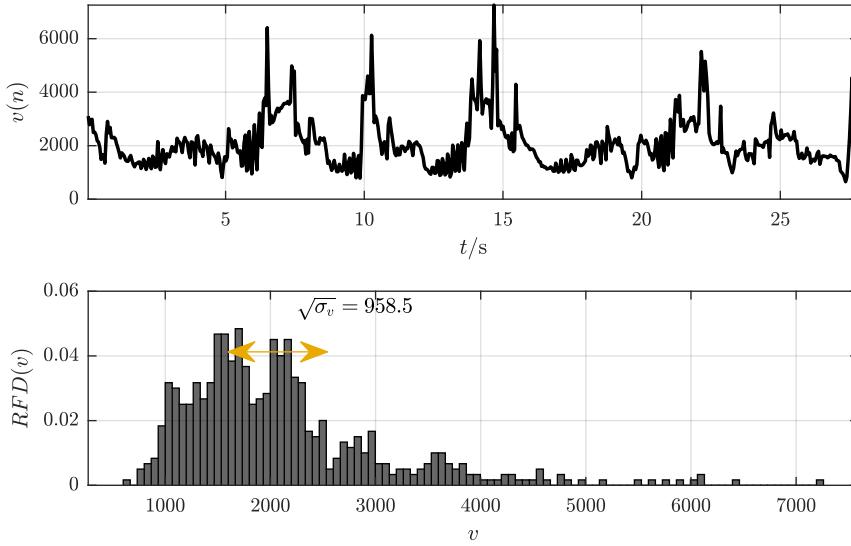
Fig. Gen.: [plotStandardDeviation.m](#)

Figure 3.6: Standard deviation of a series of feature values. Top: time series, Bottom: distribution.

Strictly speaking the former equation is the so-called *biased* estimate of the variance in contrast to the *unbiased* estimate

$$\sigma_{v,b}^2 = \frac{1}{N-1} \sum_{n=0}^{N-1} (v(n) - \mu_v)^2. \quad (3.19)$$

In case of \$\mu_v = 0\$, the variance equals the power of the observed block of samples.

The standard deviation \$\sigma_v\$ can be computed directly from the variance

$$\sigma_v = \sqrt{\sigma_v^2}. \quad (3.20)$$

In case of \$\mu_v = 0\$, the standard deviation equals the Root Mean Square (RMS) of the observed block of samples (see Sect. 8.3.1). The result of the standard deviation is in the range

$$0 \leq \sigma_v \leq \max_{i \in [0; N]} |v(i)|.$$

The standard deviation is 0 for silent or constant input signals. Its unit corresponds to the input signal's unit.

Figure 3.6 shows that similar to the arithmetic mean, the standard deviation is only of limited use for non-symmetric distributions.

3.1.3.5 Quantiles and Quantile Ranges

Quantiles can be computed from the PDF and can be used to divide the PDF into (equal sized) subsets. They are helpful in the description of asymmetric distributions or distributions with so-called outliers, occasional untypical values far from the median (see below).

If the PDF is divided into two quantiles, it is split into two parts each containing 50% of the overall number of observations. The position of the border between those two quantiles will be referred to as the *median* \$Q_v(0.5)\$:

$$Q_v(0.5) = v \left| \int_{-\infty}^v p_v(y) dy = 0.5 \right. \quad (3.21)$$

which equals the arithmetic mean in the special case of symmetric distributions. The median is visualized next to the arithmetic mean in Fig. 3.5.

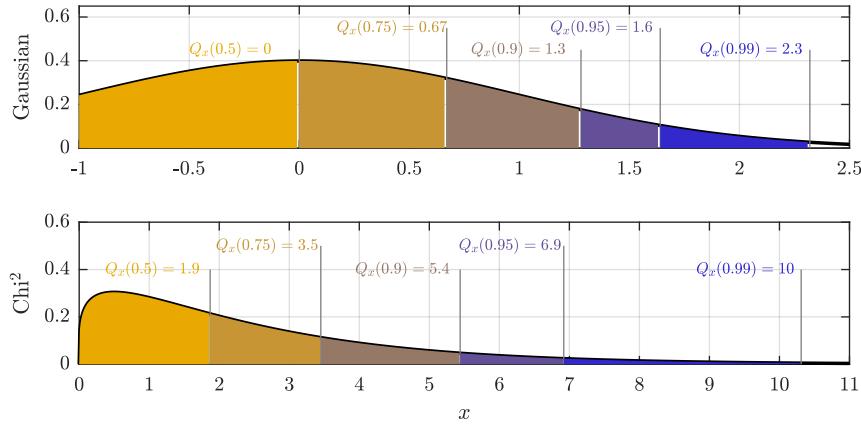


Fig. Gen.: plotPdfrQuantiles.m

Figure 3.7: Two probability distributions with different quantiles marked: Gaussian (top) and Chi-squared (bottom).

Similarly, if the PDF is partitioned in four quantiles (so-called *quartiles*), the quantile borders would be $Q_v(0.25)$, $Q_v(0.5)$, and $Q_v(0.75)$. Figure 3.7 shows selected quantiles for a Gaussian and the skewed Chi-Squared distribution.

In many cases, *quantile ranges* are of specific interest for the simplified description of a distribution's shape. The range spanned by 90% of the samples can be computed by $\Delta Q_v(0.9) = Q_v(0.95) - Q_v(0.05)$. This is often seen as a good measure of the signal's range while discarding infrequent outliers, namely the upper and lower 5%.

The overall range of a signal is

$$\Delta Q_v(1.0) = \min(Q_v(1.0)) - \max(Q_v(0)). \quad (3.22)$$

3.1.4 Digital Audio Signals

Continuous real-world signals cannot be represented digitally. Instead, audio signals are represented as a series of values, so-called *samples*. This means it is required to discretize the input signal $x(t)$ into a series of (quantized) values at equidistant times $x(i)$. A quick introduction into this process is provided in Sect. A.1. Many of the properties of continuous signals also apply to discrete signals; the most important (and over-simplified) differences are that quantized (discretized in amplitude) signals will have a noise floor with a level depending on the quantizer resolution and sampled (discretized in time) signals have a limited bandwidth that is half of the sample rate. In theory and implementation we often ignore that the signal is quantized and pretend that the amplitude is continuous. Typical audio sample rates range from 8 kHz for speech signals to 48 kHz or higher for music signals.

Similar to an image, all signal content of a digital audio signal is encoded through numerical values, however, despite many analogies between audio and image signals, there are some key points that differentiate an audio signal from an image. First, an audio signal is a time series; most content is conveyed over the course of time and can change over time. Second, multiple objects in an audio signal are superposed; although loud sources can completely or partially mask quieter ones, they cannot completely suppress the other signal like a large object can block the view on a smaller object in an image. Third, most music signals have two or more audio channels to provide spatial information; while various forms of stereoscopy exist for images, they are not as common as for music signals. Nevertheless, there are a fair number of audio processing approaches inspired by or similar to image processing methods, some of which will be introduced below.

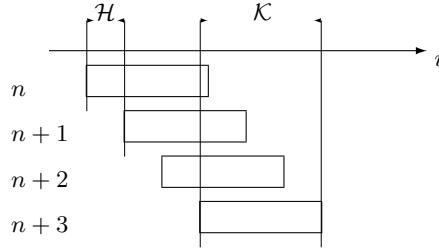


Figure 3.8: Schematic visualization of block-based processing: the input signal with the input sample index i is split into overlapping blocks of length K and block index n ; the hop size \mathcal{H} is the distance from the start of one block and the following block

3.2 Block-Based Processing

Signal processing algorithms usually work block-based, meaning that the input signal is being split into consecutive blocks of frames with length K . These blocks are processed one after another. In extreme cases, the block lengths can be either $K = 1$ or equal the length of the processed audio file, but in most cases K is chosen arbitrarily between $K = 32$ and $K = 16,384$ (in many cases powers of 2 are an advantage) depending on sample rate and required time resolution. Some reasons for this *block-based processing* are

- the internal usage of block-based algorithms such as the *Discrete Fourier Transform (DFT)* (see Sect. 3.4.1 and Appendix B),
- the characteristics of the used audio *Input/Output (IO)* hardware that usually returns blocks of samples,
- reduced memory allocation (instead of the complete audio file length; only the block length has to be allocated),
- enabling real-time capable implementations, and
- increasing computational efficiency compared to individual sample processing by avoiding function call overhead and vectorization with *Single Instruction Multiple Data (SIMD)* operations to optimize workload.

Systems working in *real time* have the constraint that the system has to be able to perform the processing of one block of data during the time frame this block consumes. Another condition of such systems is that they have no access to future input samples but only the present and possibly past samples. For real-time systems, the *latency* of the system, i.e., the delay between an input sample and the corresponding output value, can never be lower than the block length of the algorithm. The block length used to call the algorithm does not necessarily correspond to the algorithm's internal block length. If the input block length is not a multiple of the internal block length, then efficiency problems can arise since the computational workload cannot be distributed uniformly over the input blocks, resulting in occasional workload peaks.

Internally, most audio algorithms use overlapping blocks of data as depicted in Fig. 3.8. That means that the boundaries of subsequently processed blocks overlap, i.e. that the last sample of the first block is after the first sample of the second block. An audio example of this blocking is shown in Fig. 3.9. The sample indices of the block boundaries start sample $i_s(n)$ and stop sample $i_e(n)$ of processing block n can then be formulated as

$$i_s(n) = i_s(n-1) + \mathcal{H}, \quad (3.23)$$

$$i_e(n) = i_s(n) + K - 1, \quad (3.24)$$

with \mathcal{H} being the so-called *hop size* in samples. The hop size should be smaller than or equal the block length. The *block overlap ratio* can be calculated with

$$o_r = \frac{K - \mathcal{H}}{K}. \quad (3.25)$$

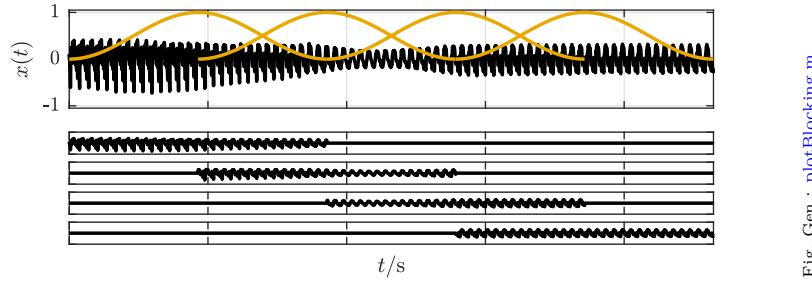
Fig. Gen.: `plotBlocking.m`

Figure 3.9: Example visualization of blocking: the input signal (top) is split into four overlapping blocks.

A typical block overlap ratio is $o_r \geq 1/2$. When only one result $v(n)$ is computed per processing block, the assigned time stamp in samples $t_s(n)$ would usually be either the start frame of the block $t_s(n) = i_s(n)/f_s$ or its middle position:

$$t_s(n) = \frac{i_e(n) - i_s(n) + 1}{2 \cdot f_s} + \frac{i_s(n)}{f_s} = \frac{\mathcal{K}}{2 \cdot f_s} + \frac{i_s(n)}{f_s}. \quad (3.26)$$

If the time stamp $t_s(n)$ is chosen to be in the middle of the block, the practical problem arises that the start and end time stamps always represent a shorter time range than the audio signal itself. For example, if the audio signal starts at sample index 0, the first time stamp will be at sample $(\mathcal{K}-1)/2$. This can be avoided by padding the signal with $\mathcal{K}/2$ zeros at beginning. Note that different blocking implementations might choose to pad a different amount of zeros to begin (typically ranging from 0 to $(\mathcal{K} - \mathcal{H})$) and end (typically ranging from 0 to \mathcal{K}) of the signal. This might lead to differences between implementations in the number of blocks even in the case of identical block lengths and hop sizes. Larger amounts of zeros ensure that no audio samples are discarded regardless of the input audio length, however, the results for the first and the last block may be unreliable due to a potentially large number of zeros.

Each audio block can be interpreted as the signal multiplied with a window function (see also Sect. B.4).

Some algorithms do not only compute results per block of audio data but combine the extracted results per block $v(n)$ again in a *texture window*. Texture windows may in turn overlap depending on texture window length and texture window hop. In this case, the same logic as above is followed with the difference that the window boundaries are now block indices instead of frame indices. Texture windows can be used to compute, e.g., windowed statistical descriptions from the series of results $v(n)$.

3.3 Audio Pre-Processing

The raw audio data is frequently pre-processed before computing instantaneous features (or more generally the input representation) from the data. The motivation for this pre-processing step is either to reduce the amount of data to be analyzed, to remove task-irrelevant information or make the signal invariant to it, or to remove redundant information. This generally increases robustness of the algorithm.

Obviously the options for pre-processing depend on the task; for instance, if we intend to determine the stereo width of a signal we cannot down-mix it to mono. However, applying some pre-processing options even if they do not seem necessary, such as applying normalization before extracting a volume-independent feature such as the spectral centroid (see Sect. 3.6.1), may still be a good choice if the system should be invariant to this input property. This allows for better maintainability and easier extensibility of the system.

3.3.1 Down-Mixing

In many analysis problems the information of interest can be represented by one single audio channel as well as by multiple input channels. For example, the tempo or information on the musical style should be extractable from a mono-recording as well as from stereo or multi-channel recordings.

```

if (size(x,2)> 1)
    x_downmix = mean(x,2);
else
    x_downmix = x;
end

```

(a) Matlab

```

if (size(x,2)> 1)
    x_downmix = mean(x,2);
else
    x_downmix = x;
end

```

(b) Python

Figure 3.10: Code snippet showing the code for downmixing a multi-channel audio signal x .
implement python

Down-mixing is usually done by simply computing the arithmetic mean, also compare Eq. (3.12), over all input channel signals $x_c(i)$ per sample i . The number of input channels is \mathcal{C}

$$x(i) = \frac{1}{\mathcal{C}} \sum_{c=0}^{\mathcal{C}-1} x_c(i). \quad (3.27)$$

It is also possible to apply different weights to different channels; surround channels may, for example, have a lower weight than front channels. An alternative to down-mixing could be to use only one pre-selected audio channel, however, this may result in loss of information if the channels have been mixed to produce a wide spatial image. Some audio applications also apply a phase shift of 90° to one channel before down-mixing a stereo signal to mono. The reason is to avoid a level boost of components present in all audio channels (mono-components) as compared to single channel components.

Figure 3.10 shows example code for downmixing an audio signal.

3.3.2 DC Removal

A DC offset — shown by a signal's arithmetic mean significantly different from zero — usually does not provide any useful information and may have unwanted impact on the following processing steps. Preferably, this should be done at the audio file level, not the block level. Mathematically, the DC offset is simply removed by subtracting the arithmetic mean of the audio data $x_{DC}(i)$ of length \mathcal{I} :

$$x(i) = x_{DC}(i) - \frac{1}{\mathcal{I}} \sum_{i=0}^{\mathcal{I}-1} x_{DC}(i). \quad (3.28)$$

In the case of real-time processing with a steady stream of subsequent samples, this can also be approximated by estimating the DC offset with a long Moving Average (MA) filter (see Sect. A.2.6.1) of a length \mathcal{O}

$$x(i) = x_{DC}(i) - \frac{1}{\mathcal{O}} \sum_{j=i-\mathcal{O}/2}^{i+\mathcal{O}/2-1} x_{DC}(j). \quad (3.29)$$

The number of coefficients has to be high to provide a reliable estimate of the arithmetic mean. Alternatively, any high-pass filter can be used to remove the DC part of the signal. The simplest filter is a differentiator:

$$x(i) = x_{DC}(i) - x_{DC}(i-1). \quad (3.30)$$

This differentiator, however, can have significant impact on higher frequency components as well. This effect can be lessened by low-pass filtering the difference; the resulting DC removal filter would be

$$x(i) = (1 - \alpha) \cdot (x_{DC}(i) - x_{DC}(i-1)) + \alpha \cdot x(i-1) \quad (3.31)$$

with α between 0 and 1 being the coefficient of this single-pole filter.

```

if (length(x)> 1)
    x_norm = x/max(abs(x),[],'all');
end

```

(a) Matlab

```

if (length(x)> 1)
    x_norm = x/max(abs(x),[],'all');
end

```

(b) Python

Figure 3.11: Code snippet showing the code for normalizing a multi-channel audio signal x .
implement python

3.3.3 Normalization

For the majority of analysis tasks, scaling the input amplitude should not affect the system output. In these cases, the signal can be *normalized* to have a pre-defined (maximum) amplitude or power.

A simple and commonly applied method to normalize an audio file is to detect the overall maximum of its absolute sample values and scale the signal so that this maximum's absolute value is mapped to 1:

$$x(i) = \frac{x_s(i)}{\max_{\forall i}(|x_s(i)|)}. \quad (3.32)$$

Figure 3.11 shows example code for this audio signal normalization.

Note that while this results in a normalized magnitude, it does not warrant equal loudness of different input files. Furthermore, the *normalization* may be influenced by signal distortions such as the clicks and crackles of a vinyl recording. In this case, the normalization to the maximum click amplitude will result in an “incorrect” scaling of the audio data.

The alternative is to use some other reference than the maximum for deriving the scaling factor, such as the RMS (see Sect. 8.3.1) or a loudness measurement with long integration time (see Sect. 8.1). In this case, additional processing may be necessary in order to avoid potential clipping of the scaled signal.

Normalizing a signal in a real-time context is difficult. Algorithms for automatic gain control or compressors and limiters provide a possible solution by monitoring the instantaneous input signal characteristics (e.g., the RMS or the peak level) and aiming at adjusting a time-variant gain value accordingly.

3.3.4 Sample Rate Conversion

Sample rate conversion can be a helpful processing step if (i) the sample rate of the input audio may be different for different input files (multiple datasets, real-world use cases), (ii) the system parametrization such as block and hop lengths should be independent of audio sample rate, and/or (iii) reducing the sample rate helps to remove task-irrelevant high frequency data and reduces the amount of audio samples to be processed.

In order to convert the sample rate from an input sample rate f_S to a target sample rate f_T , the number of samples per time unit has to be changed by a factor of l

$$f_T = \frac{f_S}{l}. \quad (3.33)$$

Taking into account the sampling theorem as given in Eq. (A.2) and the time scaling property of the spectrum as given in Eq. (3.45), it becomes clear that simply discarding or adding new samples might lead to aliasing artifacts. For example, if $l > 1$ (*Down-sampling*), picking every l th sample can only work if the input signal $x(i)$ does not contain frequency components higher than $f_T/2$; therefore, proper low-pass filtering is required when changing the sample rate.

Sample rate conversion with non-integer factors l is based on the same principles. The factor can be written as the ratio of two integer factors $l = l_1/l_2$ so that the signal can first be up-sampled by factor l_1 and then down-sampled by factor l_2 . In between the two resampling steps, it is required to apply a low-pass filter which ensures both the reconstruction of the up-sampled signal and the suppression of aliasing artifacts in the down-sampled signal. There exist various combinations of interpolation algorithms and low-pass filters for down-sampling a signal. One example for such a *band-limited* interpolation has been explained by Smith and Gosset and is usually referred to as *sinc* interpolation [SG84].

3.3.5 Other Pre-Processing Options

As with the selection of appropriate input representations, the pre-processing options will always be selected with the application in mind. Every pre-processing which improves the algorithm's accuracy or its robustness, or minimizes its complexity or computational workload is beneficial. In addition to the presented pre-processing options it is, for example, also common to attenuate the level of any unwanted frequency region by applying a filter to the signal.

3.4 Time-Frequency Representations

The vast majority of advanced audio analysis and processing systems do not utilize the time domain series of samples directly; instead, blocks of the signal are commonly transformed to the frequency domain for subsequent processing. The reasons for this are (i) a more intuitively accessible representation of the audio data that directly conveys frequency components, and (ii) the often more sparse representation of the audio data in the spectral domain.

3.4.1 Fourier Transform

The *Discrete Fourier Transform (DFT)* is one of the most important tools for processing and analyzing audio signals. A more detailed explanation — starting with the Fourier Transform (FT) of continuous signals — can be found in Appendix B. The DFT of a block of the signal $x(i)$ is defined by

$$\mathfrak{F}\{x(i)\} = \sum_{i=0}^{\mathcal{K}-1} x(i)e^{-jk_i\Delta\Omega} \quad (3.34)$$

with $\Delta\Omega$ being the distance between two frequency bins as angular frequency difference

$$\Delta\Omega = \frac{2\pi}{\mathcal{K}}. \quad (3.35)$$

The frequency of bin index k can then be computed from the block length \mathcal{K} and sample rate f_S by

$$f(k) = \frac{\Delta\Omega}{2\pi} k f_S = \frac{k}{\mathcal{K}} f_S. \quad (3.36)$$

We will refer to the DFT of the n th block with length $\mathcal{K} = i_e(n) - i_s(n) + 1$ of the audio data as *Short Time Fourier Transform (STFT)*:

$$X(k, n) = \sum_{i=i_s(n)}^{i_e(n)} x(i) \exp\left(-jk \cdot (i - i_s(n)) \frac{2\pi}{\mathcal{K}}\right). \quad (3.37)$$

Figure 3.12 shows a block of a real-world audio signal, its magnitude spectrum, as well as its real and imaginary spectrum. A fast implementation of the DFT is the *Fast Fourier Transform (FFT)*.

The most important properties of the STFT are listed on Pg. 48. See Appendix B for derivations.

A typical visualization of an audio signal that combines time and frequency components is the *spectrogram* as shown in Fig. 3.13. An STFT is calculated for each (overlapping) block of sample data; the resulting STFTs are then plotted in a pseudo-three-dimensional image in which (the magnitude of) each STFT is represented by a column and each bin is darkened according to its level. The input of the majority of ACA systems is either the waveform as discussed above or some kind of spectrogram representation. A spectrogram is a pseudo-3D plot that, compared to the waveform, often gives more insights into the frequency content of the signal by plotting the magnitude of many overlapping Short Time Fourier Transforms (STFT) over time.

Figure 3.13 shows the first 24 bars of a single saxophone playing the jazz standard 'Summertime;' the top visualizes the common waveform representation (x-axis: time, y-axis: amplitude) and the bottom displays the spectrogram (x-axis: time, y-axis: frequency, color: amplitude). Each column of the spectrogram is one

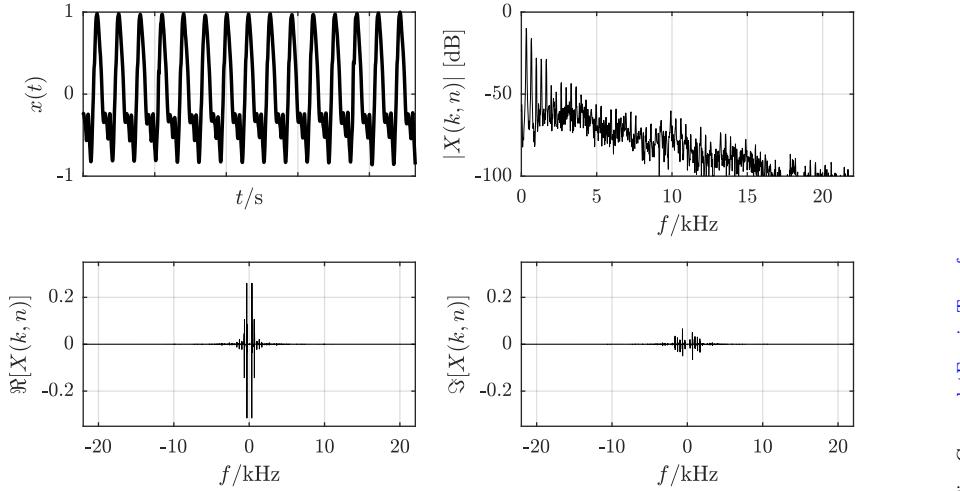


Figure 3.12: Short-Time Fourier Transform: time domain block (top left), magnitude spectrum (top right), real spectrum (bottom left), and imaginary spectrum (bottom right).

magnitude spectrum of a short block of samples with low values colored blue and high values colored yellow. While the phrases are easily identifiable in both representations, the spectrogram also shows that (i) it is a recording of a single monophonic instrument (clear spectral structure of fundamental frequency and harmonics at integer multiples), (ii) it is an instrument that allows vibrato (see seconds 3 and 18), (iii) it is an instrument with a significant number of harmonics (number and strength of “parallel” lines), and that (iv) the melody can be directly derived from the visualization by identifying the lowest frequency of the harmonic series and mapping it to musical pitch.

3.4.2 Constant Q Transform

The linear frequency axis of the STFT does not make musical sense, as neighboring low musical pitches have a smaller frequency distance than high pitches (compare Sect. 7.1). The *Constant Q Transform (CQT)* addresses this issue by adapting the STFT lengths per frequency. The CQT calculates frequency coefficients similar to the DFT but on a logarithmic frequency scale [Bro91]:

$$X(k, n) = \frac{1}{\mathcal{K}(k)} \sum_{i=i_s(n)}^{i_e(n)} w_k(i - i_s) \cdot x(i) \exp \left(j2\pi \frac{\mathcal{Q} \cdot (i - i_s)}{\mathcal{K}(k)} \right). \quad (3.48)$$

\mathcal{Q} adjusts the desired frequency resolution and can be derived from c , the desired number of frequency bins per octave, while $\mathcal{K}(k)$ depends on both the target frequency and the *quality factor* \mathcal{Q} :

$$\mathcal{Q} = \frac{f}{\Delta f} = \frac{1}{2^{1/c} - 1}, \quad (3.49)$$

$$\mathcal{K}(k) = \frac{f_s}{f(k)} \mathcal{Q}. \quad (3.50)$$

The CQT can be interpreted as computing a DFT only for specific, logarithmically spaced, frequency bins. Figure 3.14 schematically visualizes two ways of actually implementing the CQT. On the left, the largest window length is split into smaller lengths so that many STFTs of different length are computed. On the right, all STFTs are of the same size but the input content is zero-padded. The latter approach requires a hop length of the shortest block length, as opposed to the first approach, where the hop length equals the longest block length.

The CQT is not invertible and can in terms of efficiency be compared to a DFT (as opposed to the *FFT*). There are approaches to make the CQT both computationally more efficient and quasi-invertible, see, e.g., [BP92, SK10].

Fourier Transform Properties

- Invertibility

$$x(i_s(n) \dots i_e(n)) = \mathfrak{F}^{-1}\{X(k, n)\} = \frac{1}{K} \sum_{k=0}^{K-1} X(k, n) \cdot \exp\left(jki\frac{2\pi}{K}\right). \quad (3.38)$$

- Linearity and Superposition

$$\begin{aligned} \mathfrak{F}\{c_1 \cdot x(i_s(n) \dots i_e(n)) + c_2 \cdot y(i_s(n) \dots i_e(n))\} \\ = c_1 \cdot X(k, n) + c_2 \cdot Y(k, n). \end{aligned} \quad (3.39)$$

- Periodicity

$$X(k + K, n) = X(k, n). \quad (3.40)$$

- Symmetry [for real $x(i)$]

$$X(K - k, n) = X^*(k, n). \quad (3.41)$$

- Circularity

$$\mathfrak{F}^{-1}\{H(k, n) \cdot X(k, n)\} = h'(i_s(n) \dots i_e(n)) * x(i_s(n) \dots i_e(n)) \quad (3.42)$$

with h' being a *periodically extended* sequence of the signal block with the sample boundaries $i_s(n)$ and $i_e(n)$.

- Time and Frequency Shifting

$$\mathfrak{F}\{x(i - i_0)\} = X(k, n) \cdot \exp\left(-j\frac{2\pi k}{K}i_0\right) \quad (3.43)$$

$$\mathfrak{F}^{-1}\{X(k - k_0, n)\} = x(i_s(n) \dots i_e(n)) \cdot \exp\left(j\frac{2\pi i}{K}k_0\right). \quad (3.44)$$

- Time and Frequency Scaling

$$\mathfrak{F}\{x(c \cdot i)\} = \frac{1}{|c|} X\left(\frac{k}{c}, n\right). \quad (3.45)$$

- Parseval's Theorem

$$\sum_{i=i_s(n)}^{i_e(n)} |x(i)|^2 = \frac{1}{K} \sum_{k=0}^{K-1} |X(k, n)|^2. \quad (3.46)$$

- Duality

$$\mathfrak{F}\{X(i)\} = \frac{1}{K} x(k, n). \quad (3.47)$$

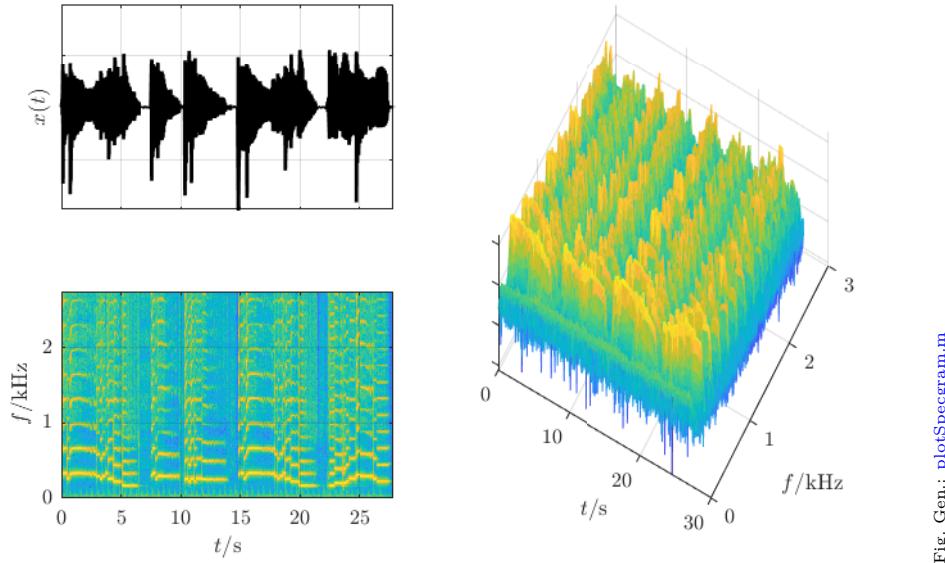


Figure 3.13: Spectrogram of a monophonic saxophone signal: each column is (the magnitude of) an STFT of a block with the time increasing from left to right; the level of each specific point in frequency and time is visualized by the color of the point; on the right is another 3D visualization of the same spectrogram.

Fig. Gen.: `plotSpecgram.m`

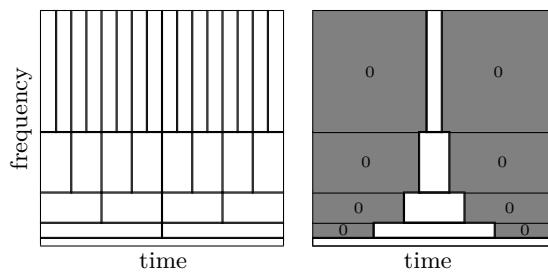


Figure 3.14: Two approaches of implementing blocking for the CQT: compute multiple STFTs with different block lengths (left) and compute multiple zero-padded STFTs of equal block length.

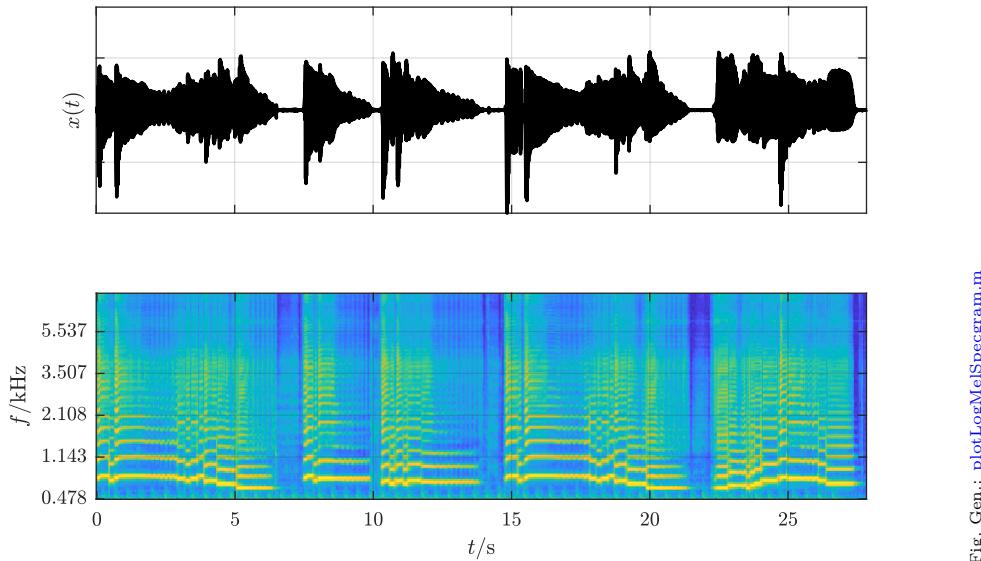
Fig. Gen.: [plotLogMelSpectrogram.m](#)

Figure 3.15: Log-Mel Spectrogram of a monophonic saxophone signal: similarly to normal spectrogram, time increases linearly from left to right with the hop length, but the frequency axis is now non-linear (harmonics are not equidistant).

3.4.3 Log-Mel Spectrogram

The Log-mel spectrogram is nowadays probably the most commonly used input representation for Deep Neural Networks (DNNs) as it seems to be a fitting representation for transporting condensed timbre and pitch information. It is computed from the normal magnitude spectrogram by grouping STFT bins together in overlapping frequency bands that are modeling the pitch perception of the human ear (see the Mel scale introduced in Sect. 7.1.1). Therefore, the frequency resolution will (roughly) logarithmically decrease with increasing frequency. The number of frequency bands is usually significantly lower than the number of frequency bins of the STFT. This Mel spectrogram becomes the Log-mel spectrogram if a logarithm is applied to the spectral magnitudes in order to approximate human loudness perception (compare Sect. 8.1).

Figure 3.15 visualizes the Log-mel spectrogram of the audio sample shown in Fig. 3.13.

Similar to the CQT, the Log-mel spectrogram has a non-linear frequency axis and is not invertible. Its computation, however, tends to be similarly efficient as the STFT as it only applies a few simple operations on the STFT (compare Fig. 3.16).

3.4.4 Auditory Filterbanks

An *auditory filterbank* is frequently used in psycho-acoustical or physiological ear models. It approximates the resolution and selectivity of the human ear. The filterbank's mid-frequencies are usually computed according to the mel scale (see Sect. 7.1).

In signal processing applications the usage of auditory filterbanks is not as widespread as one might assume. The most obvious reasons are (i) the computational workload produced by a filterbank with reasonable frequency resolution and (ii) the restriction to analysis tasks since the computed frequency domain representation cannot be transformed back to the time domain (except for a few carefully designed filterbanks with other limitations). In the case of ACA it has not been consistently shown that approaches using a filterbank give more reliable results than, e.g., an STFT-based analysis. One possible explanation is that in many cases the subject of investigation is not what humans are able to hear in a signal but physical properties of the signal. Still, the argument that ultimately every audio analysis task is about the extraction of *perceivable* features has merit as well, and the frequent use of the Mel spectrogram with its STFT filters emphasizes the usefulness of the logarithmic frequency axis. Auditory modeling is a lively subject of research; Lyon et al. review different

```

end

% in the real world, we would do this block by block...
[X,f,t] = spectrogram( afAudioData, ...
    afWindow, ...
    iBlockLength-iHopLength, ...
    iBlockLength, ...
    f_s);

% magnitude spectrum
X = abs(X)*2/iBlockLength;
X([1 end],:) = X([1 end],:)/sqrt(2); % let's be pedantic about normalization

% compute mel filters
[H,f_c] = locMelFb(iBlockLength, f_s, iNumMelBands, fMax);

M = H*X;

% amplitude to level
if (bLogarithmic)
    M = 20 * np.log10(M + 1e-12)
)
```

(a) Matlab

```

# Compute spectrogram (in the real world, we would do this block by block)
f, t, X = spectrogram(
    afAudioData,
    fs=f_s,
    window=afWindow,
    nperseg=iBlockLength,
    noverlap=iBlockLength - iHopLength,
    nfft=iBlockLength,
    detrend=False,
    return_onesided=True,
    scaling='spectrum' # Returns power spectrum
)

# Convert power spectrum to magnitude spectrum
X = np.sqrt(X / 2)

# Compute Mel filters
H, f_c = ToolMelFb(iBlockLength, f_s, iNumMelBands, fMax)

M = np.matmul(H, X)

if bLogarithmic:
    # Convert amplitude to level (dB)
    M = 20 * np.log10(M + 1e-12)
)
```

(b) Python

Figure 3.16: Computation of the Mel spectrogram by grouping bins of the magnitude spectrum.

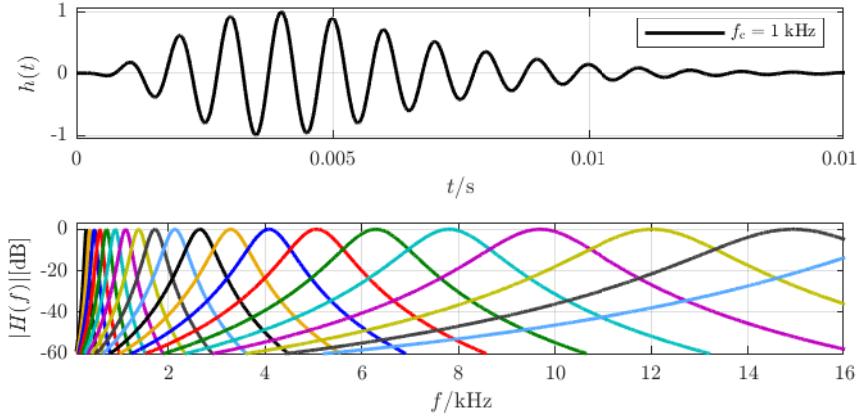


Fig. Gen.: plotGammatone.m

Figure 3.17: Top: Normalized impulse response of a gammatone filter with a center frequency $f_c = 1 \text{ kHz}$, a bandwidth $\Delta f = 125 \text{ Hz}$, and an order $O = 4$; Bottom: Magnitude frequency response of a gammatone filterbank with 20 bands between 100 Hz and 24 kHz.

approaches in [LKD10].

A widely used auditory filterbank is the so-called *gammatone filterbank*. A gammatone filter has an impulse response of the following form [BJ78, AJ80]:

$$h(i) = \frac{a \cdot (i/f_s)^{O-1} \cdot \cos\left(2\pi \cdot f_c \frac{i}{f_s}\right)}{e^{2\pi i \Delta f/f_s}} \quad (3.51)$$

Figure 3.17 shows the impulse response of one gammatone filter as well as the frequency (magnitude) response of a gammatone filterbank computed by the *Auditory Toolbox* [Sla98] for 20 bands between 100 Hz and 24 kHz. Slaney showed that gammatone filters can be efficiently implemented with cascaded second-order filters [Sla93]. A survey of different gammatone filter implementations can be found in [VIP03].

Non-auditory filterbanks have also been proposed for audio analysis. For instance, Lerch used a bank of resonance filters with one filter for each semi-tone as shown in Fig. 3.18 [Ler04].

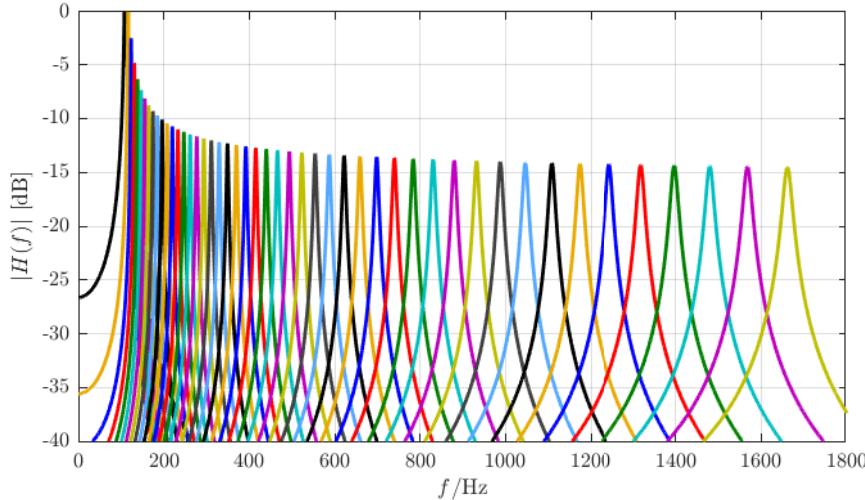


Fig. Gen.: plotResonanceFilterbank.m

Figure 3.18: Frequency response of a resonance filterbank with one filter per semi-tone.

3.5 Other Input Representations

Many other input representations for audio analysis systems have been proposed and used. These include Wavelet Transforms [SJ01], Correlograms [Sch99], and various other representations. It is hard to systematically categorize them; sometimes, the input representation of one system might be the mid-level presentation of another system. Two more task-specific input representations, the beat histogram and the self similarity matrix will be introduced later in Sects. 9.4 and 9.7, respectively.

3.6 Instantaneous Features

Historically, the input representation of audio analysis systems commonly used to be a set of *features*. These features are usually a low-dimensional, task-relevant representation of the audio signal. By selecting and designing task-relevant features, expert knowledge can be applied that allows the system to focus only the meaningful properties of the audio signal. The more powerful the feature representation is, the less powerful the classifier or other inference algorithm needs to be (and vice versa). The potential disadvantage of using this reduced feature representation is that the features might not represent *all* relevant information of the audio signal, however, once stripped away, this information is not available anymore to the classifier.

Although many state-of-the-art systems have moved away from audio feature input and are utilizing input representations such as the log-mel spectrogram, features can still be useful for tasks with limited amounts of annotated training data as neural networks tend to require large amounts of data to be able to generalize. In these cases, a feature-based approach can be preferable as it reduces the required complexity of the machine learning algorithm by utilizing expert knowledge in selecting the features. For this reason, but also because it makes sense didactically, this section introduces a selection of widely used features that have been used extensively for audio content analysis. It is important to note, however, that the choice of the specific features used in an algorithm will ultimately be driven by the task to solve; this can make the modification of well-known features as well as the design of new features advantageous or even mandatory. Therefore, the number of possible features used in audio content analysis is probably limitless and only a more or less representative set can be presented in the following.

The term *instantaneous feature*, *short-term feature*, *low-level feature*, or *descriptor* is generally used for measures that generate one value per (short) block of audio samples. An instantaneous feature is not necessarily musically, musicologically, or perceptually meaningful all by itself, and it is frequently referred to as a *low-level feature*. A low-level feature can serve as a building block for the construction of higher level features describing more meaningful properties of the (music) signal. While there is general consensus about the characteristics of

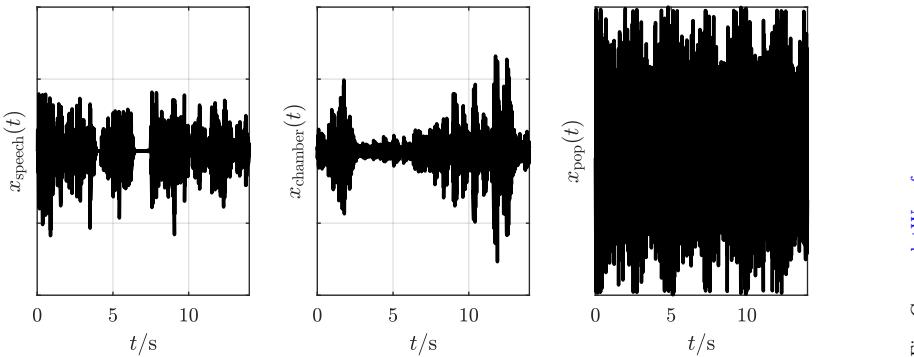


Figure 3.19: Waveform of excerpts from a pop recording (left), a string quartet recording (mid), and a speech recording (right) with a length of 15 s, respectively.

low-level features (usually extracted from short blocks of audio), there is less agreement on what characterizes a mid-level or high level feature beyond the general notion that with increasing level the musical, semantic, and/or perceptual meaning increases. Without trying to formally define something that is probably not meaningful to define, examples for mid-level and high-level features could be tempo and musical genre, respectively.

A good example for an instantaneous feature is the magnitude or power level of the audio signal extracted on a block-per-block basis. It represents a widely known reduced representation of the audio signal, the waveform. Although simple to extract, it may show characteristics usable for the extraction of higher level information. As Fig. 3.19 illustrates with three waveform views of length 15 s, the signal categories speech, chamber music, and pop music. The general shape of the waveform envelope already enables an observer to differentiate between these different signals by deriving descriptive properties related to dynamic range, fluctuations, and pauses. This is, of course, not necessarily true for every possible excerpt from these genres, but it tells us that some kind of envelope description may be useful in the automatic detection of a signal type or musical genre.

Instantaneous features can be categorized using different taxonomies, compare the MPEG-7 standard [15902] or the categories proposed by Peeters [Pee04]. It turns out that it is difficult to find a simple and consistent yet practically useful feature categorization that is task-agnostic, provides a useful level of detail, and ensures that categories are not overlapping. Therefore, we abandon elaborate categorization attempts and simply list a variety of commonly used features in the following. Figure 3.20 visualizes the process of extracting a feature value. Typically, we take one block with index n with the block length

$$\mathcal{K} = i_e(n) - i_s(n) + 1 \quad (3.52)$$

and extract one feature value $v(n)$ from this block of samples. Note that in the spectral domain, the corresponding block length halves as we remove the redundant parts of the audio (magnitude) spectrum.

The following properties are desirable for a feature to have:

- *high “discriminative” or descriptive power* since the feature should be suitable to the task at hand,
- *non-correlation to other features* because each feature should add new information to avoid redundancy,
- *invariance to irrelevancies* to allow the feature to be robust against, e.g., linear transformations of the input audio signal such as scaling and filtering operations (low-pass filtering, reverberation), the addition of signals such as (background) noise, coding artifacts as well as the application of non-linear operations such as distortion and clipping (see Wegener et al. for an example evaluation of feature robustness [WHB⁺08]), and
- *reasonable computational complexity* to ensure that the feature is able to be computed on the target platform (such as a mobile device) and for the required application, respectively.

Note that this list is neither exhaustive nor are all items necessarily required, but it is a good starting point when compiling a feature set.

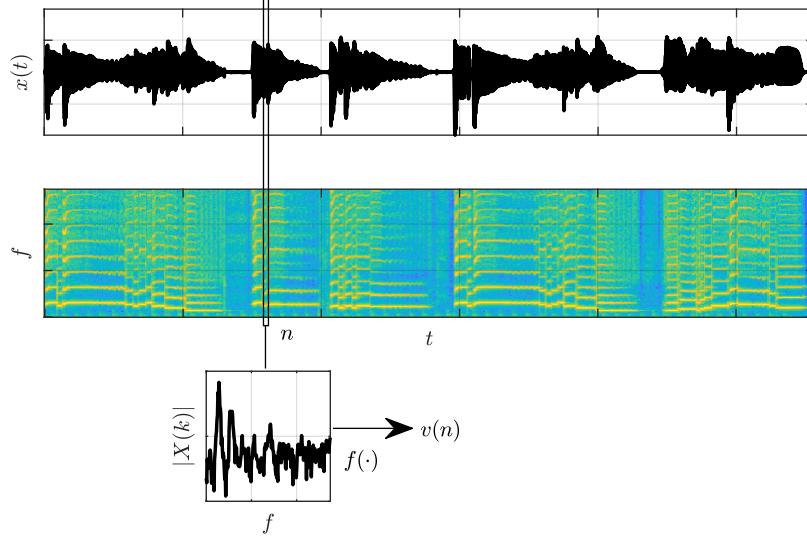
Fig. Gen.: [plotFeatureExtraction.m](#)

Figure 3.20: Visualization of the feature extraction process.

Most features extracted in the frequency domain describe the spectral shape of the audio signal and are closely related to the *timbre* of this signal. The *timbre* of a sound is referred to as its *sound color*, its *quality*, or its *texture*. Besides pitch and loudness, timbre is considered as “the third attribute of the subjective experience of musical tones” [RP82]. Loudness and pitch are uni-dimensional properties, as sounds with different loudness or pitch can be ordered on a single scale from quiet to loud and low to high, respectively. Timbre is a multi-dimensional property [ZF67, Moo97]; this complicates its definition. The most infamous example is probably the definition of the American Standards Association from 1960 that defined timbre as “that attribute of auditory sensation in terms of which a listener can judge that two sounds similarly presented and having the same loudness and pitch are dissimilar” [ASA60]. This definition has been criticized repeatedly by researchers mainly because according to Bregman [Bre94] it

- does not attempt to explain what timbre is, but only what timbre is *not*, i.e., loudness and pitch, and
- implies that timbre only exists for sounds with a pitch, implicating that, for example, percussive instruments do not have a timbre.

Early uses of the term *timbre* can be found in Blumenbach [Blu28]. Helmholtz was probably the first to detect the dependency between the timbre of a sound and the relative amplitudes of the harmonics during the second half of the 19th century [Hel70]. Although he noted that other influences play a role in defining the quality of a tone such as the “beginning” and “ending” of a sound, he restricted his definition of timbre (“Klangfarbe”) to the harmonic amplitude distribution only. Stumpf extended the definition of timbre by two more attributes [Stu90]. He named the relative amplitude of harmonics, the form and length of the attack time and note endings, and additional sounds and noise as the third timbre-determining component. Seashore restricted the term timbre during the first half of the 20th century to the harmonic structure that “is expressed in terms of the number, distribution, and relative intensity of its partials,” but he additionally proposed the term *sonance*, referring to “the successive changes and fusions which take place within a tone from moment to moment” [Sea38]. This distinction, however, did not find broad acceptance in the research community. Nowadays, timbre is understood as the phenomenon that takes into account both spectral patterns and temporal patterns [Reu95, Moo97]. Timbre perception is obviously influenced by numerous parameters of both the onset properties such as rise time, inharmonicities during the onset, etc. and numerous steady-state effects such as vibrato, tremolo, pitch instability, etc. [RP82]. In the following, we will restrict ourselves to measures of spectral shape since most of the features describing “temporal timbre” only work for individual monophonic notes as opposed to complex time-variant mixtures of signals. Many of the presented spectral features definitions are technically motivated;

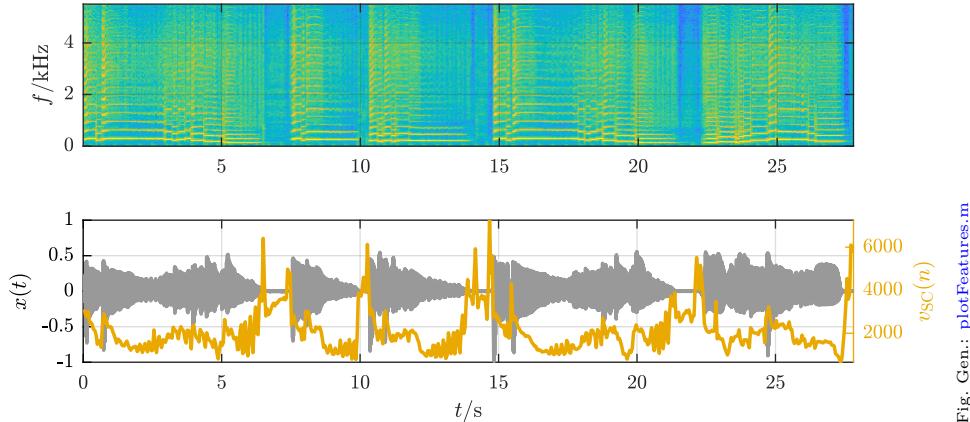


Fig. Gen.: plotFeatures.m

Figure 3.21: Spectrogram (top), waveform (bottom background), and spectral centroid (bottom foreground) of a saxophone signal.

```

vsc      = ([0:size(X,1)-1]*X)./sum(X,1);
# X = X**2 removed for consistency with book
% avoid NaN for silence frames
vsc (sum(X,1) == 0) = 0;
norm = X.sum(axis=0, keepdims=True)
norm[norm == 0] = 1
vsc = np.dot(np.arange(0, X.shape[0]), X) / norm
# convert from index to Hz
vsc      = vsc / (size(X,1)-1) * f_s/2;
    
```

(a) Matlab

```

# X = X**2 removed for consistency with book
norm = X.sum(axis=0, keepdims=True)
norm[norm == 0] = 1
vsc = np.dot(np.arange(0, X.shape[0]), X) / norm
# convert from index to Hz
    
```

(b) Python

Figure 3.22: Implementation of Spectral Centroid from the magnitude spectrogram $X(k, n)$.

therefore, even if they describe the spectral shape of a signal, there might be no direct relation to any dimension of human timbre perception.

3.6.1 Spectral Centroid

The *spectral centroid* represents the Center of Gravity (COG) of spectral magnitudes. In other words, the magnitude spectrum is interpreted as distribution and the spectral centroid is the mean of the distribution (compare Eq. (3.13)). Since the magnitude spectrum is—unlike the typical distribution—not usually normalized to a sum of 1, the spectral centroid is the frequency-weighted sum of the magnitude spectrum normalized by its unweighted sum:

$$v_{SC}(n) = \frac{\sum_{k=0}^{\mathcal{K}/2} k \cdot |X(k, n)|}{\sum_{k=0}^{\mathcal{K}/2} |X(k, n)|}. \quad (3.53)$$

The result of the spectral centroid is a bin index within the range $0 \leq v_{SC}(n) \leq \mathcal{K}/2$. It can be converted either to Hz by using Eq. (3.36) or to a parameter range between zero and one by dividing it by the STFT size $\mathcal{K}/2$. Low results indicate significant low-frequency components and insignificant high frequency components.

Figure 3.21 shows the spectral centroid for the example saxophone signal introduced above. In the case of this monophonic signal one can see how the spectral centroid moves with the fundamental frequency during tonal parts, spikes at initial transients, and is high during pauses because of the noise in the audio signal.

As the spectral centroid is undefined for silence input frames containing only zeros, special consideration is required (see Fig. 3.22).

In the literature, numerous indications can be found that the spectral centroid is highly correlated with the timbre dimension *brightness* or *sharpness* [Bis74, IK93, MWD⁺95, Lak00, MCMW03, SWT04, CMSW05].

It is also common to compute the spectral centroid from the power spectrum instead of the magnitude

spectrum:

$$v_{\text{SC}}(n) = \frac{\sum_{k=0}^{\kappa/2} k \cdot |X(k, n)|^2}{\sum_{k=0}^{\kappa/2} |X(k, n)|^2}. \quad (3.54)$$

Zwicker and Fastl presented a psycho-acoustic model of sharpness that uses the excitation patterns to compute the sharpness [ZF99]. It differs from Eq. (3.53) mainly in two points. First, it is computed on a non-linear bark scale, namely the so-called critical band rate which models the non-linearity of human frequency perception (compare Sect. 7.1.1). Second, it utilizes a psycho-acoustic loudness measure instead of the spectral power; this loudness model takes into account masking and other perceptual effects. While ignoring the difference between loudness and power, the idea of a perceptually motivated non-linear frequency scale has been adapted for the definition of the spectral centroid in the MPEG-7 standard [15902]. The critical band rate is approximated by applying a logarithm to the frequencies with a reference point of $f_{\text{ref}} = 1000$ Hz:

$$v_{\text{SC,log}}(n) = \frac{\sum_{k=k(f_{\min})}^{\kappa/2} \log_2 \left(\frac{f(k)}{f_{\text{ref}}} \right) \cdot |X(k, n)|^2}{\sum_{k=k(f_{\min})}^{\kappa/2} |X(k, n)|^2}. \quad (3.55)$$

In this specific MPEG-definition, all bins corresponding to frequencies below 62.5 Hz are combined to one band with a mid-frequency of 31.25 Hz.

The unnormalized spectral centroid is sometimes referred to as *High Frequency Content (HFC)* [MB96].

3.6.2 Spectral Spread

The *spectral spread*, sometimes also referred to as *instantaneous bandwidth* [Bar93], describes the concentration of the magnitude spectrum around the spectral centroid. If the spectrum is interpreted as a distribution, the spectral spread is the standard deviation of this distribution (compare Eq. (3.20)) around its spectral centroid. Its definition is

$$v_{\text{SS}}(n) = \sqrt{\frac{\sum_{k=0}^{\kappa/2} (k - v_{\text{SC}}(n))^2 \cdot |X(k, n)|}{\sum_{k=0}^{\kappa/2} |X(k, n)|}}. \quad (3.56)$$

The result of the spectral spread is a bin range of $0 \leq v_{\text{SS}}(n) \leq \kappa/4$. It can be converted either to Hz by using Eq. (3.36) or to a parameter range between zero and one by dividing it by $\kappa/4$. Low results indicate the concentration of the spectral energy at a specific frequency region and high values indicate a noise-like spectrum.

Figure 3.23 shows the spectral spread for an example signal. Most prominent are the high feature values during pauses and at transients; the spectral spread is low during steady pitches for this monophonic signal. When the higher harmonics slowly disappear between 12 and 15 s, the spread of the signal decreases accordingly.

As the spectral centroid, the spectral spread is not defined for audio blocks with no spectral energy (see Fig. 3.24).

There are indications that the spectral spread is of some relevance in describing the perceptual dimensions of timbre [MCMW03].

The calculation of the spectral spread has to conform with the definition of the spectral centroid. If the spectral centroid has been calculated from the power spectrum instead of the magnitude spectrum, then the spectral spread should use the power spectrum as well. In the case of the MPEG-7 definition, a logarithmic

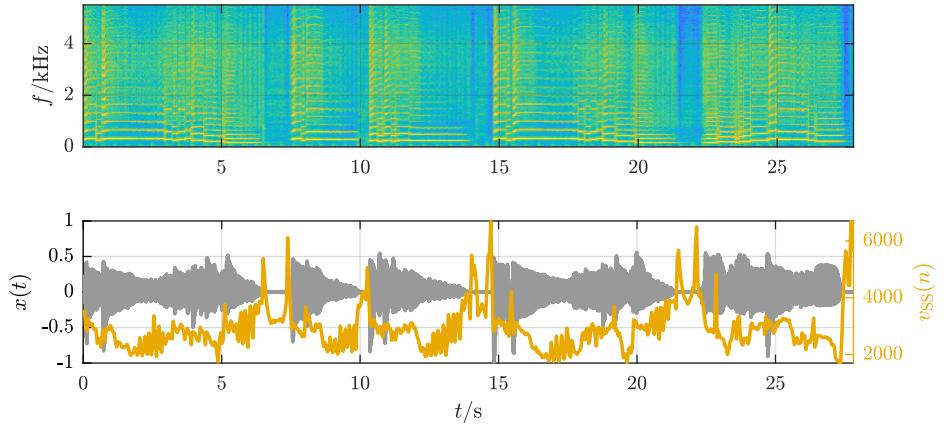


Fig. Gen.: plotFeatures.m

Figure 3.23: Spectrogram (top), waveform (bottom background), and spectral spread (bottom foreground) of a saxophone signal.

```
% get spectral centroid as index
vsc      = FeatureSpectralCentroid(X, f_s)*2/f_s * (size(X,1)-1);

% allocate memory
vss      = zeros(size(vsc));

% compute spread
%X      = X.^2;
for n = 1:size(X,2))
    vss(n) = (([0:size(X,1)-1]-vsc(n)).^2*X(:,n))/sum(X(:,n));
end
vss      = sqrt(vss);

% convert from index to Hz
vss      = vss / (size(X,1)-1) * f_s/2;

% avoid NaN for silence frames
vss (sum(X,1) == 0) = 0;
```

(a) Matlab

```
isSpectrum = X.ndim == 1
if isSpectrum:
    X = np.expand_dims(X, axis=1)

# get spectral centroid as index
vsc = FeatureSpectralCentroid(X, f_s) * 2 / f_s * (X.shape[0] - 1)

# X = X**2 removed for consistency with book
norm = X.sum(axis=0)
norm[norm == 0] = 1

# compute spread
vss = np.zeros(X.shape[1])
indices = np.arange(0, X.shape[0])
for n in range(0, X.shape[1]):
    vss[n] = np.dot((indices - vsc[n])**2, X[:, n]) / norm[n]
```

(b) Python

Figure 3.24: Implementation of Spectral Spread from the magnitude spectrogram $X(k, n)$.

```
% interpret the spectrum as pdf, not as signal
f      = linspace(0, f_s/2, size(X,1));
% compute mean and standard deviation
mu_X   = FeatureSpectralCentroid(X, f_s);
std_X   = FeatureSpectralSpread(X, f_s);
tmp    = repmat(f, size(X,2),1) - repmat(mu_X, size(X,1),1)';
vssk   = sum((tmp.^3).*X)' ./ (std_X.^3 .* sum(X,1)*size(X,1));
(a) Matlab
```

```
# compute kurtosis
vssk = np.sum(X**3, axis=0) / (std_x**3 * X.shape[0])
else:
    f = np.arange(0, X.shape[0]) / (X.shape[0] - 1) * f_s / 2
    # get spectral centroid and spread (mean and std of dist)
    vsc = FeatureSpectralCentroid(X, f_s)
    vss = FeatureSpectralSpread(X, f_s)
    norm = X.sum(axis=0)
    norm[norm == 0] = 1
    vss[vss == 0] = 1
(b) Python
```

Figure 3.25: Implementation of Spectral Skewness from the magnitude spectrogram $X(k, n)$.

frequency scale has to be used for the calculation of the spectral spread as well:

$$v_{SS,\log}(n) = \sqrt{\frac{\sum_{k=k(f_{\min})}^{\kappa/2} \left(\log_2 \left(\frac{f(k)}{f_{\text{ref}}} \right) - v_{SC}(n) \right)^2 \cdot |X(k, n)|^2}{\sum_{k=k(f_{\min})}^{\kappa/2} |X(k, n)|^2}}. \quad (3.57)$$

3.6.3 Spectral Skewness and Spectral Kurtosis

The *skewness* and *kurtosis* are derived from the third and fourth order central moments of a distribution, respectively (compare also Sect. 3.1.3.4). The central moment of order \mathcal{O} is defined by

$$\gamma_{v,\beta} = \sum_0^{N-1} \left(v(n) - \mu_v \right)^\mathcal{O}. \quad (3.58)$$

The skewness is useful as a measure of the asymmetry of the PDF. It will be 0 for symmetric distributions, negative for distributions with their mass centered on the right (left-skewed), and positive for distributions with their mass centered on the left (right-skewed). Note that while every symmetric distribution has zero skewness the converse is not necessarily true. The kurtosis is a measure of “non-Gaussianity” of the PDF; more specifically, it indicates the flatness (and peakiness, respectively) of the input values’ distribution compared to the Gaussian distribution. It equals 0 for a Gaussian distribution (*mesokurtic*), is negative for a flatter distribution with a wider peak (*platykurtic*), and positive for distributions with a more acute peak (*leptokurtic*).

Similar to the spectral centroid and spectral spread, the *spectral skewness* interprets the spectrum as a distribution. It measures the symmetry of the distribution of the spectral magnitude values around their spectral centroid. It is defined by

$$v_{SSk}(n) = \frac{\sum_{k=0}^{\kappa/2} (k - v_{SC}(n))^3 \cdot |X(k, n)|}{v_{SS}^3 \cdot \sum_{k=0}^{\kappa/2} |X(k, n)|}. \quad (3.59)$$

The spectral skewness will result in a value close to zero for noisy signals and higher values for signals with their main energy in the low frequencies.

The spectral skewness is not defined for silence input and relies on the computation of spectral centroid and spectral spread.

Figure 3.26 shows the spectral skewness for an example signal. During signal pauses the spectral skewness drops since the spectral magnitudes are similar, while at positions with high magnitudes at the fundamental frequency the magnitude spectrum is significantly skewed. The spectral skewness increases, for example, in the region between 12s and 14s due to the strong decrease of the higher harmonics compared to the lower harmonics.

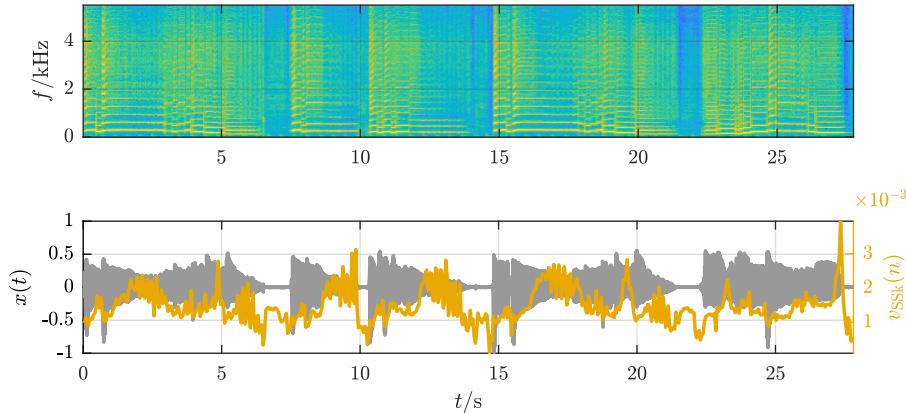


Fig. Gen.: plotFeatures.m

Figure 3.26: Spectrogram (top), waveform (bottom background), and spectral skewness (bottom foreground) of a saxophone signal.

```

# compute kurtosis
vsk = np.sum(X**4, axis=0) / (std_X**4 * X.shape[0])
else:
    f = np.arange(0, X.shape[0]) / (X.shape[0] - 1) * f_s / 2
    # get spectral centroid and spread (mean and std of dist)
    vsc = FeatureSpectralCentroid(X, f_s) # *2/f_s * (X.shape[0]-1)
    vss = FeatureSpectralSpread(X, f_s) # *2/f_s * (X.shape[0]-1)

    norm = X.sum(axis=0)
    norm[norm == 0] = 1
    vss[vss == 0] = 1

# compute kurtosis
vsk = np.zeros(X.shape[1])

```

(a) Matlab

```

# compute kurtosis
vsk = np.sum(X**4, axis=0) / (std_X**4 * X.shape[0])
else:
    f = np.arange(0, X.shape[0]) / (X.shape[0] - 1) * f_s / 2
    # get spectral centroid and spread (mean and std of dist)
    vsc = FeatureSpectralCentroid(X, f_s) # *2/f_s * (X.shape[0]-1)
    vss = FeatureSpectralSpread(X, f_s) # *2/f_s * (X.shape[0]-1)

    norm = X.sum(axis=0)
    norm[norm == 0] = 1
    vss[vss == 0] = 1

# compute kurtosis
vsk = np.zeros(X.shape[1])

```

(b) Python

Figure 3.27: Implementation of Spectral Kurtosis from the magnitude spectrogram $X(k, n)$.

The *spectral kurtosis* measures whether the distribution of the spectral magnitude values is shaped like a Gaussian distribution or not. It is defined by

$$v_{SK}(n) = \frac{\sum_{k=0}^{\mathcal{K}/2} (k - v_{SC}(n))^4 \cdot |X(k, n)|}{v_{SS}^4 \cdot \sum_{k=0}^{\mathcal{K}/2} |X(k, n)|} - 3. \quad (3.60)$$

The spectral kurtosis is not defined for silence input and relies on the computation of spectral centroid and spectral spread.

Figure 3.28 shows the spectral kurtosis for a saxophone signal. While during the notes high values can be observed, indicating a very peaked distribution, the spectral kurtosis drops significantly during pauses.

3.6.4 Spectral Rolloff

The *spectral rolloff* is a measure of the bandwidth of the analyzed block n of audio samples. The spectral rolloff $v_{SR}(n)$ is defined as the frequency bin below which the accumulated magnitudes of the STFT $X(k, n)$ reach a certain percentage κ of the overall sum of magnitudes:

$$v_{SR}(n) = k_r \left| \sum_{k=0}^{k_r} |X(k, n)| = \kappa \cdot \sum_{k=0}^{\mathcal{K}/2} |X(k, n)| \right| \quad (3.61)$$

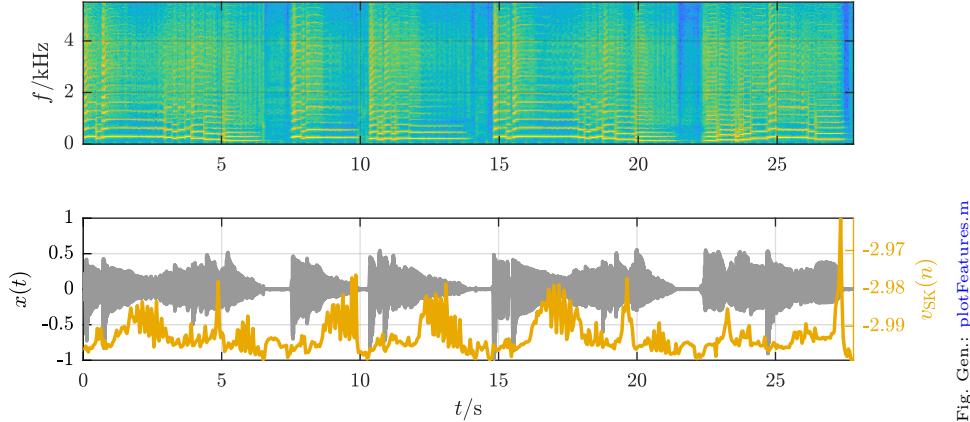


Fig. Gen.: plotFeatures.m

Figure 3.28: Spectrogram (top), waveform (bottom background), and spectral kurtosis (bottom foreground) of a saxophone signal.

```

afSum = sum(X,1);
for n = 1:length(vsr)
    vsr(n) = find(cumsum(X(:,n)) >= kappa*afSum(n), 1)-1;
end

% convert from index to Hz
vsr = vsr / (size(X,1)-1) * f_s/2;

```

(a) Matlab

```

norm = X.sum(axis=0, keepdims=True)
norm[norm == 0] = 1
X = np.cumsum(X, axis=0) / norm
vsr = np.argmax(X >= kappa, axis=0)

```

(b) Python

Figure 3.29: Implementation of Spectral Rolloff from the magnitude spectrogram $X(k, n)$.

with common values for κ being 0.85 (85%) or 0.95 (95%). This can also be interpreted as the Quantile $Q_X(\kappa)$ (compare Sect. 3.1.3.5) of the spectrum as distribution.

The result of the spectral rolloff is a bin index in the range $0 \leq v_{SR}(n) \leq \kappa/2$. It can be converted either to Hz with Eq. (3.36) or to a parameter range between zero and one by dividing it by the STFT size $\kappa/2$. Low results indicate insignificant magnitude components at high frequencies and thus a low audio bandwidth. The spectral rolloff is not defined for silence input frames (see Fig. 3.29).

Figure 3.30 shows the spectral rolloff for an example signal. It is comparably low in the presence of a tone and higher — although somewhat erratic — during the noise-filled pauses.

Spectral bins representing very low or very high frequencies may in many cases be considered to be unnecessary or unwanted for the analysis. Therefore, both sums in Eq. (3.61) may start and stop at pre-defined frequency boundaries f_{\min} , f_{\max} :

$$v_{SR,\Delta f}(n) = k_r \left| \sum_{k=k(f_{\min})}^{k_r} |X(k,n)| = \kappa \cdot \sum_{k=k(f_{\min})}^{k(f_{\max})} |X(k,n)| \right|. \quad (3.62)$$

It is also common to use the power spectrum instead of the magnitude spectrum.

3.6.5 Spectral Decrease

The *spectral decrease* estimates the steepness of the decrease of the spectral envelope over frequency. It is defined by [Pee04]

$$v_{SD}(n) = \frac{\sum_{k=1}^{\kappa/2} \frac{1}{k} \cdot (|X(k,n)| - |X(0,n)|)}{\sum_{k=1}^{\kappa/2} |X(k,n)|}. \quad (3.63)$$

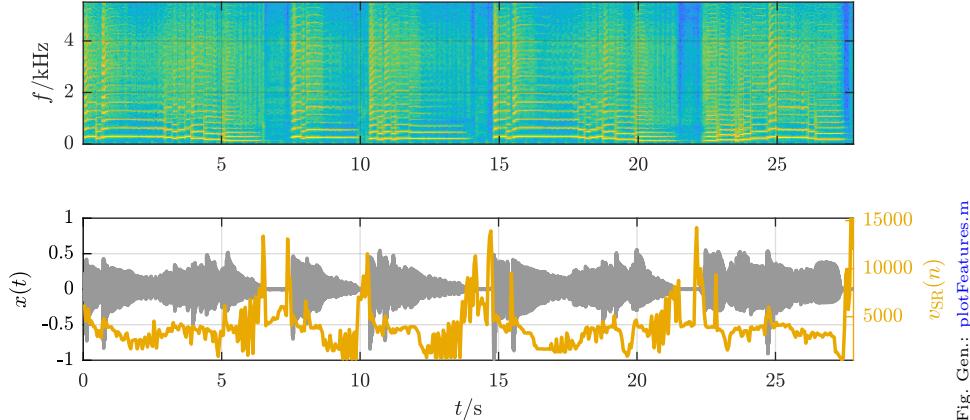


Figure 3.30: Spectrogram (top), waveform (bottom background), and spectral rolloff (bottom foreground) of a saxophone signal.

```
% compute index vector
k       = [0:size(X,1)-1];
k(1)   = 1;
kinv  = 1./k;

% compute slope
vsd    = (kinv*(X-repmat(X(1,:),size(X,1),1)))./sum(X(2:end,:),1);

% avoid NaN for silence frames
vsd (sum(X(2:end,:)) == 0) = 0;
(a) Matlab
```

```
# compute index vector
kinv = np.arange(0, X.shape[0])
kinv[0] = 1
kinv = 1 / kinv

norm = X[1:].sum(axis=0, keepdims=True)
norm[norm == 0] = 1

# compute slope
vsc = np.dot(kinv, X - X[0]) / norm

return np.squeeze(vsc, axis=0)
(b) Python
```

Figure 3.31: Implementation of Spectral Decrease from the magnitude spectrogram $X(k, n)$.

The result of the spectral decrease is a value $v_{SD}(n) \leq 1$. Low results indicate the concentration of the spectral energy at bin 0. The spectral decrease is not defined for silence input frames (see Fig. 3.31).

Figure 3.32 shows the spectral decrease for an example signal. The difficulty to draw any conclusions from the plot except that the feature behaves erratically during the pauses explains why this feature is not frequently used.

Reducing the spectral analysis range can lead to more meaningful results in some cases. This can be done by using a different lower and upper bound for the sum.

3.6.6 Spectral Slope

The *spectral slope* is — similar to the spectral decrease — a measure of the slope of the spectral shape. It is calculated using a linear approximation of the magnitude spectrum; more specifically, a linear regression approach is used (compare Sect. D). The spectral slope is estimated with the equation

$$v_{SSl}(n) = \frac{\sum_{k=0}^{\kappa/2} (k - \mu_k)(|X(k, n)| - \mu_{|X|})}{\sum_{k=0}^{\kappa/2} (k - \mu_k)^2} \quad (3.64)$$

The result of the spectral slope depends on the amplitude range of the spectral magnitudes. Example code is shown in Fig. 3.33.

Figure 3.34 shows the spectral slope for an example signal. It increases with disappearing higher harmonics and is maximal for the noisy rests.

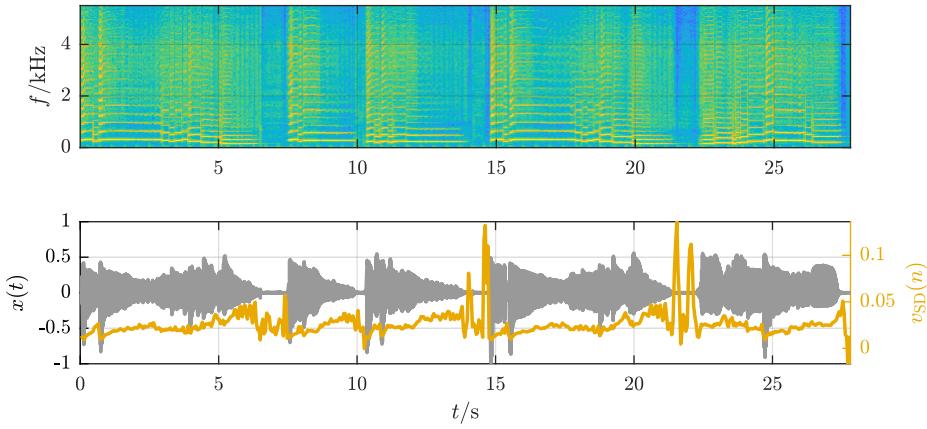


Fig. Gen.: plotFeatures.m

Figure 3.32: Spectrogram (top), waveform (bottom background), and spectral decrease (bottom foreground) of a saxophone signal.

```
% compute mean
mu_x = mean(abs(X), 1);

% compute index vector
kmu = [0:size(X,1)-1] - size(X,1)/2;

% compute slope
X = X - repmat(mu_x, size(X,1), 1);
vssl = (kmu*X)/(kmu*kmu');

(a) Matlab
```

```
# compute mean
mu_x = X.mean(axis=0, keepdims=True)

# compute index vector
kmu = np.arange(0, X.shape[0]) - X.shape[0] / 2

# compute slope
X = X - mu_x
vssl = np.dot(kmu, X) / np.dot(kmu, kmu)

(b) Python
```

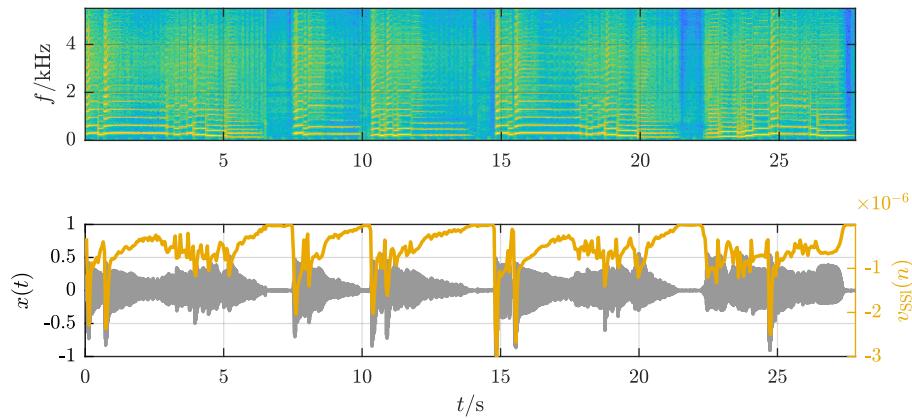
Figure 3.33: Implementation of Spectral Slope from the magnitude spectrogram $X(k, n)$.

Fig. Gen.: plotFeatures.m

Figure 3.34: Spectrogram (top), waveform (bottom background), and spectral slope (bottom foreground) of a saxophone signal.

3.6.7 Mel Frequency Cepstral Coefficients

The *Mel Frequency Cepstral Coefficients (MFCCs)* can be seen as a compact description of the shape of the spectral envelope of an audio signal. The j th coefficient $v_{\text{MFCC}}^j(n)$ can be calculated with

$$v_{\text{MFCC}}^j(n) = \sum_{k'=1}^{K'} \log(|X_{\text{warp}}(k', n)|) \cdot \cos\left(j \cdot \left(k' - \frac{1}{2}\right) \frac{\pi}{K'}\right) \quad (3.65)$$

with $|X_{\text{warp}}(k', n)|$ being the mel-warped magnitude spectrum at the signal block. The MFCC calculation is based on the following steps:

1. computation of the mel-warped (see Sect. 7.1.1) spectrum with a bank of overlapping band-pass filters (also compare Sect. 3.4.3),
2. taking the logarithm of the magnitude of each resulting band, and
3. calculating the *Discrete Cosine Transform (DCT)* on the resulting bands. The DCT equals the real (cosine) part of an FT.

The MFCCs have been widely used in the field of speech signal processing since their introduction in 1980 [DM80] and have been found to be useful in music signal processing applications as well [Foo97, Log00, PDW03, BL03]. In the context of audio signal classification, it has been shown that a small subset of the resulting MFCCs as shown in Fig. 3.36 already contains the principal information [TC02, MB03] — in most cases the number of used MFCCs varies in the range from 4 to 20. Nowadays, the MFCCs are probably the most frequently used audio features for baseline systems due to their proven robustness and usefulness over a wide range of tasks.

The calculation is closely related to the calculation of the cepstrum as introduced in Sect. 7.3.3.6 as it transforms a spectral representation after applying a logarithm to it. The main difference to the standard cepstrum is the use of a warped non-linear frequency scale (the mel scale, see Sect. 7.1.1) to model the non-linear human perception of frequency and the use of the DCT instead of a DFT. The mel-warped basis functions for the DCT are displayed in Fig. 3.35.

The mel warping of the spectrum frequently leads to the claim that the MFCCs are “perceptual” features. This is only partly true as there is no psycho-acoustic evidence to motivate the application of the DCT. Furthermore, there is no direct correlation between the MFCCs and known perceptual dimensions.

The result of the MFCCs depends on the amplitude range of the spectral power. The zeroth MFCC $v_{\text{MFCC}}^0(n)$ is usually ignored as it has no relevance in describing the timbre. It is highly correlated to the energy in decibel. The first four coefficients are shown in Fig. 3.36. Despite their proven usefulness it is difficult to identify non-trivial relationships to the input signal.

The MFCCs are not defined for silence as input signal. An excerpt of the source code to extract the MFCCs is shown in Fig. 3.37

The differences between MFCC implementations can be found mainly in the computation of the mel-warped spectrum, i.e., in number, location, and normalization of the filters. Table 3.1 shows the differences between the three most popular MFCC implementations, the original introduced by Davis and Mermelstein (DM) [DM80], the implementation in the *HMM Toolkit (HTK)* software [YEH⁺02], and the implementation in Slaney’s *Auditory Toolbox (SAT)* [Sla98].

Figure 3.38 shows the triangular filter shapes as used in the Slaney’s *Auditory Toolbox*.

Section 7.1.1 lists the typical mel scale models used for the non-linear frequency warping. It is also possible to use other filter shapes or to compute MFCCs directly from the power spectrum by using warped cosine basis functions as shown in Fig. 3.35 [MPSN01]. The power spectrum might also be approximated by other means such as through linear prediction coefficients.

3.6.8 Spectral Flux

The *spectral flux* is different from the other features here in that it measures the amount of change of the spectral shape instead of describing the spectral shape itself. It is defined as the average difference between

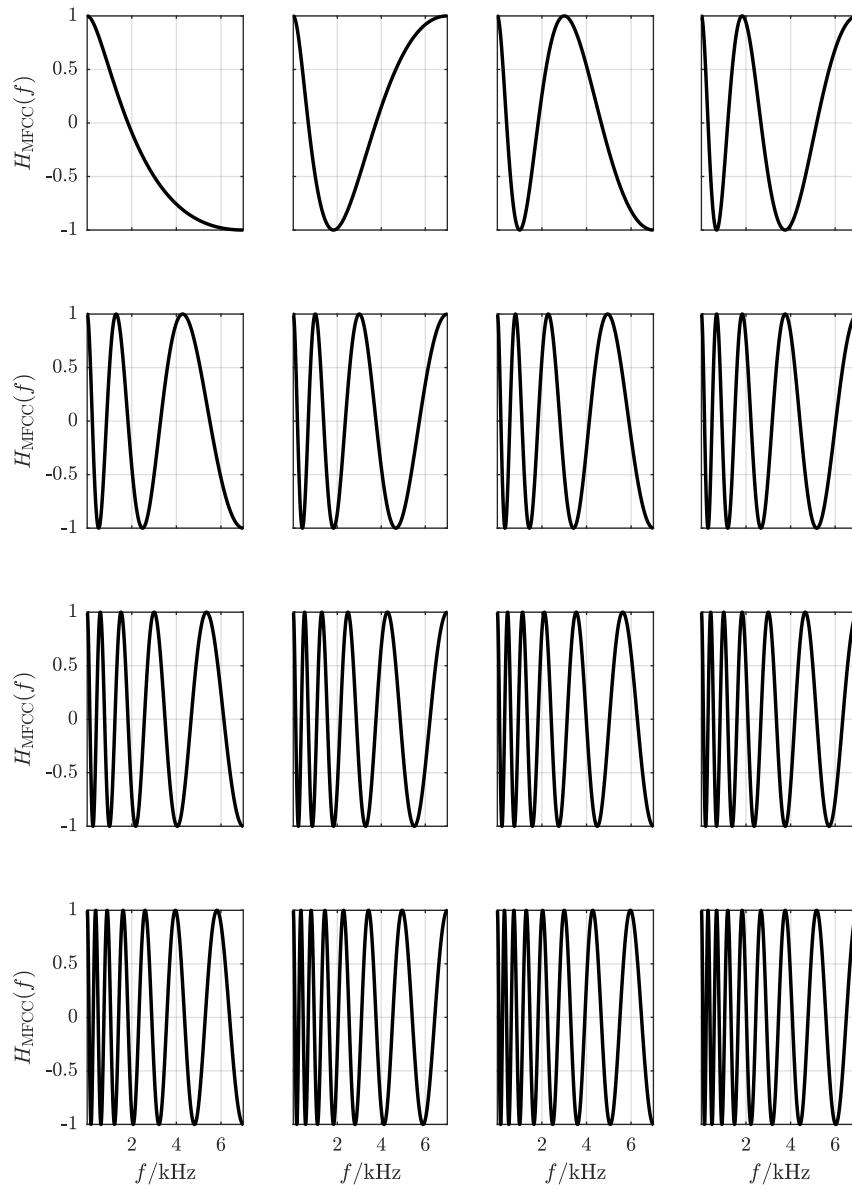
Fig. Gen.: [plotMfccMelDct.m](#)

Figure 3.35: Warped cosine-shaped transformation basis functions for the computation of MFCC coefficients (order increases from left to right and from top to bottom).

Table 3.1: Properties of three popular MFCC implementations

Property	DM	HTK	SAT
Num. filters	20	24	40
Mel scale	lin/log	log	lin/log
Freq. range	[100; 4000]	[100; 4000]	[200; 6400]
Normalization	Equal height	Equal height	Equal area

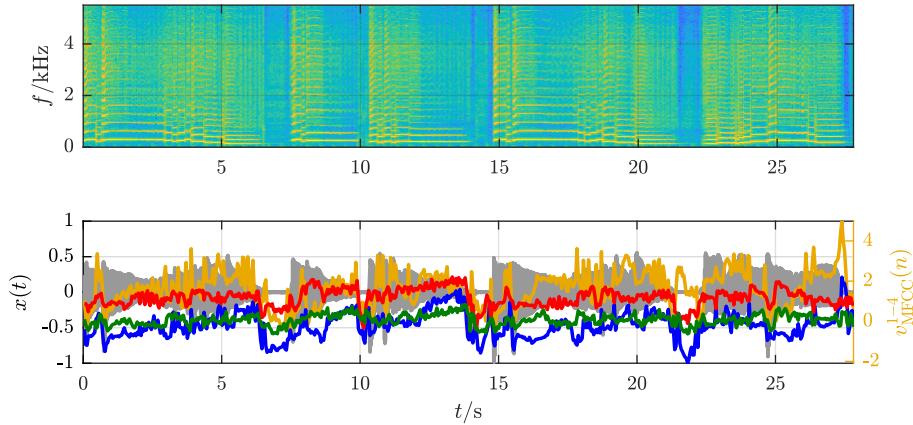


Fig. Gen.: plotFeatures.m

Figure 3.36: Spectrogram (top) and mel frequency cepstral coefficients 1–4 (bottom) of a saxophone signal

```
iNumCoeffs = 13;
% allocate memory
vmfcc = zeros(iNumCoeffs, size(X,2));
H = ToolMfccFb(size(X,1), f_s);
T = GenerateDctMatrix (size(H,1), iNumCoeffs);

for (n = 1:size(X,2))
    % compute the mel spectrum
    X_Mel = log10(H * X(:,n)+1e-20);

    % calculate the mfccs
    vmfcc(:,n) = T * X_Mel;
end
```

(a) Matlab

```
isSpectrum = X.ndim == 1
if isSpectrum:
    X = np.expand_dims(X, axis=1)

# allocate memory
v_mfcc = np.zeros([iNumCoeffs, X.shape[1]])

# generate filter matrix
H = ToolMfccFb(X.shape[0], f_s)
T = generateDctMatrix(H.shape[0], iNumCoeffs)

for n in range(0, X.shape[1]):
    # compute the mel spectrum
```

(b) Python

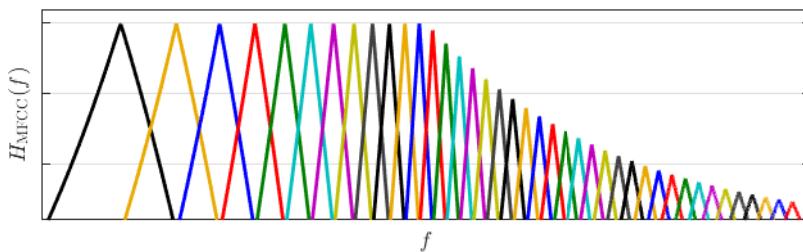
Figure 3.37: Implementation of Spectral MFCCs from the magnitude spectrogram $X(k, n)$.

Fig. Gen.: plotMfccFilterbank.m

Figure 3.38: Magnitude transfer function of the filterbank for MFCC computation as used in Slaney's *Auditory Toolbox* [Sla98].

```
% difference spectrum (set first diff to zero)
afDeltaX = diff([X(:,1), X],1,2);

% flux
vsf = sqrt(sum(afDeltaX.^2))/size(X,1);
(a) Matlab
```

```
isSpectrum = X.ndim == 1
if isSpectrum:
    X = np.expand_dims(X, axis=1)

# difference spectrum (set first diff to zero)
X = np.c_[X[:, 0], X]
(b) Python
```

Figure 3.39: Implementation of Spectral Flux from the magnitude spectrogram $X(k, n)$.

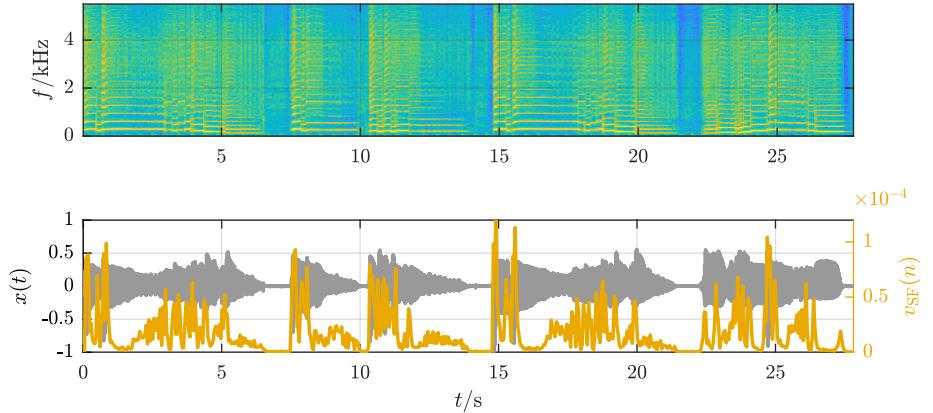


Fig. Gen.: `plotFeatures.m`

Figure 3.40: Spectrogram (top), waveform (bottom background), and spectral flux (bottom foreground) of a saxophone signal.

consecutive STFT frames:

$$v_{SF}(n) = \frac{\sqrt{\sum_{k=0}^{\mathcal{K}/2} (|X(k, n)| - |X(k, n-1)|)^2}}{\mathcal{K}/2 + 1}. \quad (3.66)$$

In some ways, the spectral flux can be seen as related to the sensation of *roughness* which, according to Zwicker and Fastl, is driven by a quasi-periodic change or a modulation in the excitation pattern levels [ZF99].

The result of the spectral flux is a value within the range $0 \leq v_{SF}(n) \leq A$ with A representing the maximum possible spectral magnitude. Thus, its output range depends on the normalization of the audio signal and the frequency transform. Low results indicate steady-state input signals or low input levels. The source code for the computation of the spectral flux can be found in Fig. 3.39.

Figure 3.40 shows the spectral flux for an example signal. It is low during the stationary parts of the signal, such as during a note or a pause, and spikes at pitch changes and at the beginning of a new note.

The definition of the spectral flux above is the Euclidean distance of the two spectra as given in Eq. (4.5). The distance norm can also be generalized:

$$v_{SF}(n, \beta) = \frac{\sqrt{\sum_{k=0}^{\mathcal{K}/2} (|X(k, n)| - |X(k, n-1)|)^\beta}}{\mathcal{K}/2 + 1}. \quad (3.67)$$

Typical values for β range between $[0.25; 3]$, with $\beta = 1$ [Manhattan distance, Eq. (4.6)] and $\beta = 2$ [Euclidean distance, Eq. (4.5)] being the most common.

In some applications such as note onset detection, only an increase in spectral energy is of interest; in these cases, the difference magnitude spectrum is computed¹

$$\Delta X(k, n) = |X(k, n)| - |X(k, n-1)| \quad (3.68)$$

¹Another distance measure can be used as well.

```

vtsc = max(X,[],1) ./ sum(X,1);

% avoid NaN for silence frames
vtsc (sum(X,1) == 0) = 0;

```

(a) Matlab

```

norm = X.sum(axis=0, keepdims=True)
norm[norm == 0] = 1

```

(b) Python

Figure 3.41: Implementation of Spectral CrestFactor from the magnitude spectrogram $X(k, n)$.

and all negative differences $\Delta X(k, n) < 0$ will be set to zero before summation while positive differences will be left unaltered. This is called *Half-Wave Rectification (HWR)*. Mathematically the HWR of a signal x is

$$\text{HWR}(x) = \frac{x + |x|}{2}. \quad (3.69)$$

Alternative approaches can be used to derive a measure of spectral change. An example is the computation of the standard deviation of the difference magnitude spectrum $\Delta X(k, n)$:

$$v_{\text{SF},\sigma}(n) = \sqrt{\frac{2}{\mathcal{K}+2} \sum_{k=0}^{\mathcal{K}/2} (\Delta X(k, n) - \mu_{\Delta X})^2}. \quad (3.70)$$

Another variant is to compute the logarithmic difference. This has the advantage of making the resulting feature independent of magnitude scaling but the disadvantage of zero-mean frames having to be handled individually:

$$v_{\text{SF},\log}(n) = \frac{2}{\mathcal{K}+2} \sum_{k=0}^{\mathcal{K}/2} \log_2 \left(\frac{|X(k, n)|}{|X(k, n-1)|} \right). \quad (3.71)$$

3.6.9 Spectral Crest Factor

The *spectral crest factor* gives an estimate on “how sinusoidal” a spectrum is. It is a simple measure of *tonalness* in the sense that it crudely estimates the amount of *tonal* components in the signal as opposed to noisy components.²

The spectral crest factor is the ratio of maximum of the magnitude spectrum and the sum of this magnitude spectrum. It is, therefore, defined by

$$v_{\text{Tsc}}(n) = \frac{\max_{0 \leq k \leq \mathcal{K}/2} |X(k, n)|}{\sum_{k=0}^{\mathcal{K}/2} |X(k, n)|}. \quad (3.72)$$

The result of the spectral crest factor is a value between $2/\mathcal{K}+2 \leq v_{\text{Tsc}}(n) \leq 1$. Low results indicate a flat magnitude spectrum and high results indicate a sinusoidal. The spectral crest factor is not defined for audio blocks with no spectral energy (compare Fig. 3.41).

Figure 3.42 shows the spectral crest factor for an example signal. It is low for noisy parts during the pauses and higher during the tonal passages. With decreasing amplitude of higher harmonics, the spectral crest factor increases as the spectral energy is more and more concentrated at a single spectral bin.

In some implementations, the denominator of Eq. (3.72) can also be the arithmetic mean of the magnitude spectrum. This scales the range of the spectral crest factor to $1 \leq v_{\text{Tsc}}(n) \leq \frac{2}{\mathcal{K}+2}$.

²The somewhat unusual term *tonalness* is used here to distinguish this measure from the musical term *tonality* which describes a specific harmonic or key context.

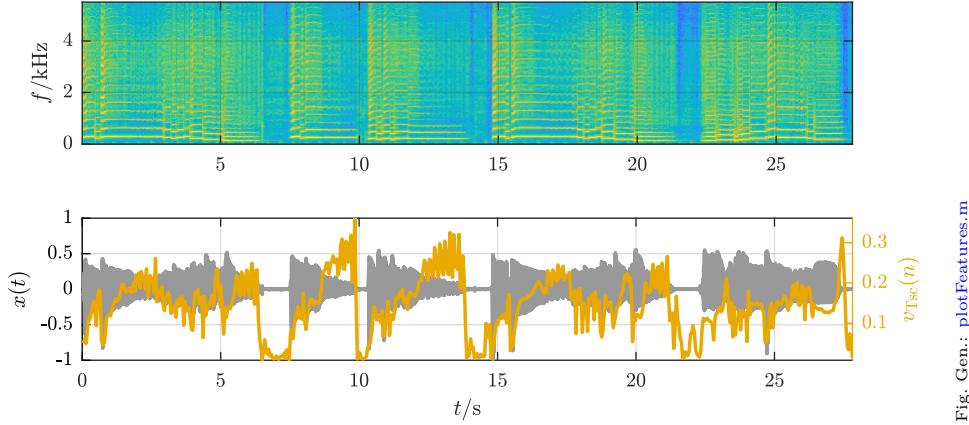


Figure 3.42: Spectrogram (top), waveform (bottom background), and spectral crestfactor (bottom foreground) of a saxophone signal.

```
XLog      = log(X+1e-20);
vtf      = exp(mean(XLog,1)) ./ (mean(X,1));

% avoid NaN for silence frames
vtf (sum(X,1) == 0) = 0;
(a) Matlab
```

```
norm = X.mean(axis=0, keepdims=True)
norm[norm == 0] = 1
X = np.log(X + 1e-20)
vtf = np.exp(X.mean(axis=0, keepdims=True)) / norm
(b) Python
```

Figure 3.43: Implementation of Spectral Flatness from the magnitude spectrogram $X(k, n)$.

3.6.10 Spectral Flatness

The *spectral flatness* is the ratio of geometric mean and arithmetic mean of the magnitude spectrum. It is defined by [JN84]

$$v_{\text{Tf}}(n) = \frac{\sqrt[\kappa/2]{\prod_{k=0}^{\kappa/2-1} |X(k, n)|}}{\frac{2/\kappa}{\sum_{k=0}^{\kappa/2-1} |X(k, n)|}} = \frac{\exp\left(\frac{2/\kappa}{\sum_{k=0}^{\kappa/2-1} \log(|X(k, n)|)}\right)}{\frac{2/\kappa}{\sum_{k=0}^{\kappa/2-1} |X(k, n)|}}. \quad (3.73)$$

The latter formulation uses the arithmetic mean of the logarithmic magnitude spectrum in the numerator in order to avoid problems with computing accuracy.

The result of the spectral flatness is a value larger than 0. The upper limit depends on the maximum spectral magnitude. Low results hint toward a non-flat — possibly a tonal — spectrum, while high results indicate a flat (or noisy) spectrum. The spectral flatness is thus a measure of noisiness as opposed to tonalness. However, as soon as only the magnitude at one individual bin equals 0, v_{Tf} will be zero as well. The implementation of the spectral flatness is shown in Fig. 3.43.

The behavior of the spectral flatness at pauses in the input signal requires special consideration as it is not defined for silence and will be comparably large for (low-level) noise.

Figure 3.44 shows the spectral flatness for an example signal. It is low during tonal passages, high in noisy pauses, and produces spikes at transients.

It is common to use the power spectrum instead of the magnitude spectrum in order to emphasize peaks. To avoid problems with individual zero magnitudes having too large an impact on the overall result, the magnitude spectrum can be smoothed. One typical approach is to compute the arithmetic mean of a group of neighboring spectral coefficients, which is basically the same as applying an MA filter to the magnitude spectrum. However, the length of the filter might also increase with frequency to take into account the lower frequency resolution of the human ear at higher frequencies. In many cases, more useful information can be gathered if the spectral flatness calculation takes only magnitudes within a pre-defined frequency range into account, as opposed to

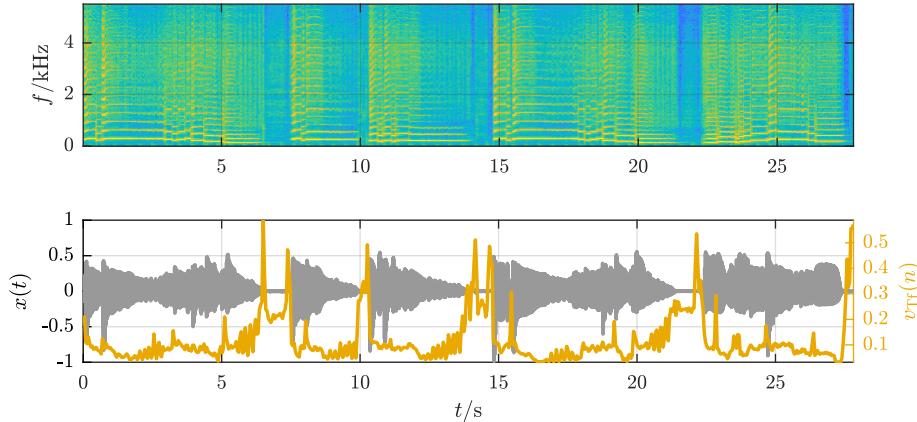


Fig. Gen.: plotFeatures.m

Figure 3.44: Spectrogram (top), waveform (bottom background), and spectral flatness (bottom foreground) of a saxophone signal.

computing it from the whole spectrum. The MPEG-7 standard recommends a frequency range of from 250 Hz to 16 kHz, divided into 24 slightly overlapping frequency bands with quarter-octave bandwidth [15902]. Since the spectral flatness is then computed for each individual frequency band, the result per STFT is a vector of spectral flatness results.

3.6.11 Tonal Power Ratio

Another way to compute the tonalness of a spectrum is to compute the ratio of the tonal power $E_T(n)$ to the overall power:

$$v_{\text{Tpr}} = \frac{E_T(n)}{\sum_{i=0}^{\kappa/2} |X(k, n)|^2}. \quad (3.74)$$

This reduces the problem to the estimation of the power of the tonal components. A simple approximation to estimating the tonal energy is summing all bins k which

- are a local maximum: $|X(k-1, n)|^2 \leq |X(k, n)|^2 \geq |X(k+1, n)|^2$ and
- lie above a threshold G_T .

The result of the tonal power ratio is a value between $0 \leq v_{\text{Tpr}} \leq 1$. Low results hint toward a flat (noisy) spectrum or a block with low input level while high results indicate a tonal spectrum. The tonal power ratio is not defined for audio blocks with no spectral energy (compare Fig. 3.45).

Figure 3.46 shows the tonal power ratio for an example signal. It is zero in noisy pauses, high for tonal passages and, in the case of this simple saxophone signal, drops distinctively at the initial transients at note beginnings.

There are many other proposed approaches for identifying tonal components in the spectral domain, as the range of applications which can benefit from a reliable tonalness detector spans psycho-acoustic models in perceptual audio encoders that can be optimized with a more accurate estimation of the signal-to-mask ratio, source separation algorithms and analysis/synthesis systems such as phase vocoders which may be improved by treating tonal and noisy components separately, and pitch-based analysis systems such as key detection and chord recognition and ultimately music transcription systems for which non-tonal information usually should be removed as irrelevant.

Similar to the simple approach mentioned above, Parsons identified peaks by finding local maxima in the magnitude spectrum — a first processing step that can be found in practically every publication dealing with the detection of tonal bins — and used the peak’s symmetry, its proximity to the next peak as well as the continuity of the frequency bin’s phase for detecting “peak overlaps,” i.e., bins with supposedly two or more

```
% initialize
if (nargin < 3)
    G_T = 5e-4;
end

% allocate memory
vptr = zeros(1,size(X,2));

X = X.^2;
fSum = sum(X,1);

for (n = 1:size(X,2))
    if (fSum(n) == 0)
        % do nothing for 0-blocks
        continue;
    end
    % find local maxima
    [afPeaks] = findpeaks(X(:,n));

    % find peaks above the threshold
    k_peak = find(afPeaks > G_T);

    % calculate the ratio
    vptr(n) = sum(afPeaks(k_peak))/fSum(n);
end

(a) Matlab
```

```
isSpectrum = X.ndim == 1
if isSpectrum:
    X = np.expand_dims(X, axis=1)

X = X**2

fSum = X.sum(axis=0)
vptr = np.zeros(fSum.shape)

for n in range(0, X.shape[1]):
    if fSum[n] < G_T:
        continue

    # find local maxima above the threshold
    afPeaks = find_peaks(X[:, n], height=G_T)

    if not afPeaks[0].size:
        continue

    if isSpectrum:
        X = np.expand_dims(X, axis=1)

    vptr[n] = afPeaks[0].sum() / fSum[n]

(b) Python
```

Figure 3.45: Implementation of Spectral TonalPowerRatio from the magnitude spectrogram $X(k, n)$.

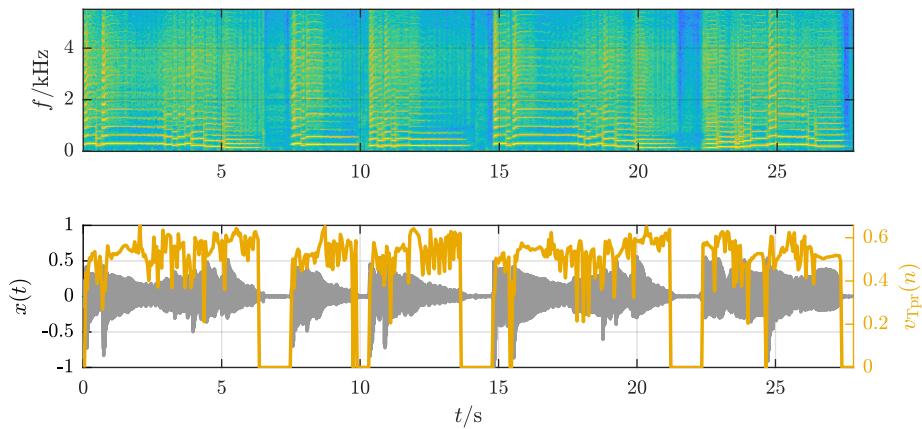


Fig. Gen.: plotFeatures.m

Figure 3.46: Spectrogram (top), waveform (bottom background), and spectral tonalpowerratio (bottom foreground) of a saxophone signal.

influencing sinusoidals [Par76]. Terhardt extended the concept of detecting local maxima by expecting more distant bins (specifically the bins with a distance of 2 and 3) to be a certain level lower than the maximum itself [TSS82]. Serra proposed a measure of “peakiness” of local maxima by comparing the bin magnitude with the surrounding local minima; he also discarded peaks outside of a pre-defined frequency and magnitude range [Ser89]. An amplitude-based measure computing the correlation function between the magnitude spectrum and the shifted spectrum of the used window function has been presented by Peeters and Rodet [PR98] as well as Lagrange [LMR02]. They also utilized a phase-derived measure comparing the bin frequency of a peak with its reassigned (instantaneous) frequency (see Sec. B.6). In addition to a local maximum feature similar to Terhardt’s, Every proposed to discard peaks below an adaptive threshold computed from the low-pass filtered magnitude spectrum [Eve06]. Röbel et al. presented a set of features to classify spectral peaks into being sinusoidal or non-sinusoidal [RZR04]. These features included the deviation of the bin frequency and its reassigned frequency, the peak’s energy location according to its group delay, as well as the bandwidth of a spectral peak.

The publications presented above make a binary decision for a spectral bin being tonal or not; Kulesza and Czyzewski proposed an algorithm which aims at estimating the likelihood of a bin’s tonalness [KC10]. They refer to this as a scoring classifier and use a so-called peakiness feature similar to Serra’s, a frequency stability criterion for detected peaks, and a phase-based frequency coherence over subsequent blocks of the STFT. Their approach combines several features and uses a combination of heuristics and both binary and non-binary features to compute the resulting likelihood. Similarly, Kraft and Lerch extract multiple features per frequency local maximum, including its “peakiness,” its saliency (computed by adaptive thresholding), and a “frequency coherence” measure based on the distance of the bin frequency and its instantaneous frequency [KLZ13]. The individual features weighted with a Gaussian function can then also be aggregated for an overall tonalness measure.

3.6.12 Maximum of Autocorrelation Function

The ACF of a time signal yields local maxima where the ACF lag matches the wavelengths of the signal-inherent periodicities (see Sect. A.3.2). The less periodic and therefore less tonal the signal is, the lower is the value of such maxima. The absolute value of the overall *ACF maximum* is therefore a simple estimate of the signal’s tonalness:

$$v_{\text{Ta}}(n) = \max_{0 \leq \eta \leq \mathcal{K}-1} |r_{xx}(\eta, n)|. \quad (3.75)$$

Values in the main lobe of the ACF around lag $\eta = 0$ have to be discarded to ensure useful results. Different approaches can be used to ignore the main lobe:

- *Minimum lag*: Assuming that a maximum of interest will not be found at high frequencies (small lags and period lengths, respectively), the search for the maximum can be started at a pre-defined lag, ignoring values at smaller lags. The lower the expected maximum frequency is, the larger the minimum lag can be. Depending on the task at hand and the sample rate, the maximum frequency might be too high to correspond to a reasonably large minimum lag. For example, at a sample rate of 48 kHz a frequency of 9.6 kHz corresponds to a lag of 5 samples, a frequency of 4.8 kHz to a lag of 10 samples, and a frequency of 1920 Hz to a lag of 25 samples.
- *Minimum magnitude threshold*: Maxima are only detected at lags larger than the lag η_r . This lag is the smallest lag at which r_{xx} crosses a pre-defined threshold G_r

$$\eta_r = \operatorname{argmin}_{0 \leq \eta \leq \mathcal{K}-1} (r_{xx}(\eta) < G_r). \quad (3.76)$$

Theoretically, however, the threshold might never be crossed; this case has to be considered in the implementation.

- *Search range from the first local minimum*: Only consider maxima at lags larger than the lag of the “first” local minimum. The idea is to avoid the detection of “insignificant” local maxima in the main lobe around lag $\eta = 0$, but depending on the signal, a local minimum might be detected at a very low lag.

Practically, a combination of these approaches plus additional, problem-specific constraints are used to ensure meaningful results.

The result is a value between $-1 \leq v_{\text{Ta}}(n) \leq 1$. This ACF-based feature will work best for monophonic signals or signals with a limited number of fundamental frequencies. Low results indicate a non-periodic signal and high result values indicate a periodic signal (compare Fig. 3.47).

```
% initialization
% these values are arbitrary - adapt to your use case
if ( nargin < 6)
    f_max      = 2000;
end
if ( nargin < 5)
    fMinThresh = 0.35;
end

% number of results
iNumOfBlocks = floor ((length(x)-iBlockLength)/iHopLength + 1);

% compute time stamps
t = ((0:iNumOfBlocks-1) * iHopLength + (iBlockLength/2))/f_s;

% allocate memory
vta = zeros(1,iNumOfBlocks);

for (n = 1:iNumOfBlocks)
    eta_min = ceil(f_s/f_max);

    i_start = (n-1)*iHopLength + 1;
    i_stop  = min(length(x),i_start + iBlockLength - 1);

    % calculate the acf
    afCorr = xcorr(x(i_start:i_stop), 'coeff');
    afCorr = afCorr((ceil((length(x)/2))+1):end);

    % ignore values until threshold was crossed
    eta_tmp = find (afCorr < fMinThresh, 1);
    if (~isempty(eta_tmp))
        eta_min = max(eta_min, eta_tmp);
    end

    % only take into account values after the first minimum
    afDeltaCorr = diff(afCorr);
    eta_tmp = find(afDeltaCorr > 0, 1);
    if (~isempty(eta_tmp))
        eta_min = max(eta_min, eta_tmp);
    end
end

# create blocks
xBlocks = pyACA.ToolBlockAudio(x, iBlockLength, iHopLength)

# number of results
iNumOfBlocks = xBlocks.shape[0]

# compute time stamps
t = (np.arange(0, iNumOfBlocks) * iHopLength + (iBlockLength / 2)) / f_s

# allocate memory
vacf = np.zeros(iNumOfBlocks)

for n, block in enumerate(xBlocks):
    eta_min = np.floor(f_s / f_max).astype(int)

    # calculate the acf
    if not block.sum():
        continue
    else:
        afCorr = np.correlate(block, block, "full") / np.dot(block, block)

        afCorr = afCorr[np.arange(iBlockLength, afCorr.size)]

        # update eta_min to avoid main lobe
        eta_tmp = np.argmax(afCorr < fMinThresh)
        eta_min = np.max([eta_min, eta_tmp])

        afDeltaCorr = np.diff(afCorr)
        eta_tmp = np.argmax(afDeltaCorr > 0)
        eta_min = np.max([eta_min, eta_tmp])

        # find the coefficients specified in eta
        vacf[n] = np.max(afCorr[np.arange(eta_min + 1, afCorr.size)])
return vacf,
```

(a) Matlab

(b) Python

Figure 3.47: Implementation of MaxAcf.

Figure 3.48 shows the ACF maximum for an example signal. As expected, there is the tendency of giving low values at noisy pause segments and higher values for tonal segments.

3.6.13 Zero Crossing Rate

The number of changes of sign in consecutive blocks of audio samples — the *zero crossing rate* — is a low-level feature that has been used for decades in speech and audio analysis due to its simple calculation:

$$v_{\text{ZC}}(n) = \frac{1}{2 \cdot \mathcal{K}} \sum_{i=i_s(n)}^{i_e(n)} |\text{sign}[x(i)] - \text{sign}[x(i-1)]| \quad (3.77)$$

with the sign function being defined by

$$\text{sign}[x(k)] = \begin{cases} 1, & \text{if } x(i) > 0 \\ 0, & \text{if } x(i) = 0 \\ -1, & \text{if } x(i) < 0 \end{cases} \quad (3.78)$$

and $x(i-1) = 0$ used as initialization if $x(i-1)$ does not exist.

The output is a value in the range of $0 \leq v_{\text{ZC}}(n) \leq 1$. The more often the signal changes its sign, the more noisy or high-frequency content can be assumed to be in the signal. Furthermore, the more the zero crossing rate varies over blocks, the less periodic the signal can be assumed to be. The concept of the zero

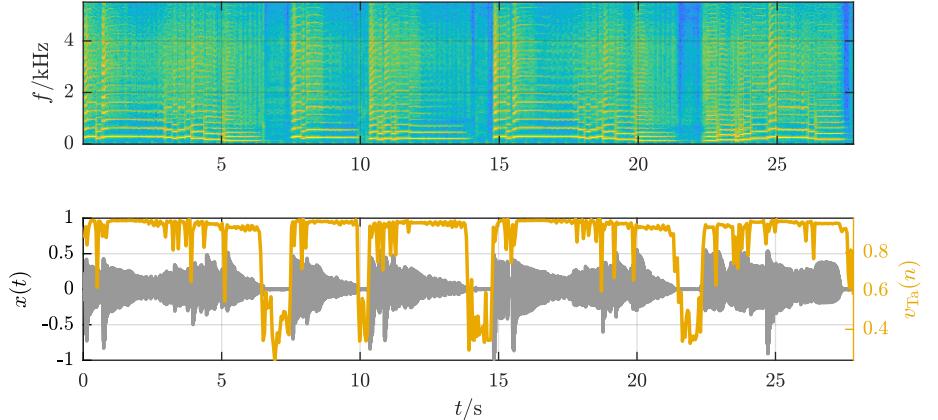


Fig. Gen.: plotFeatures.m

Figure 3.48: Spectrogram (top), waveform (bottom background), and feature maxacf (bottom foreground) of a saxophone signal.

```
% number of results
iNumOfBlocks = floor ((length(x)-iBlockLength)/iHopLength + 1);

% compute time stamps
t = ((0:iNumOfBlocks-1) * iHopLength + (iBlockLength/2))/f_s;

% allocate memory
vzc = zeros(1,iNumOfBlocks);

for n = 1:iNumOfBlocks
    i_start = (n-1)*iHopLength + 1;
    i_stop = min(length(x),i_start + iBlockLength - 1);

    % compute the zero crossing rate
    vzc(n) = 0.5*mean(abs(diff(sign(x(i_start:i_stop)))));
end
```

(a) Matlab

```
# create blocks
xBlocks = pyACA.ToolBlockAudio(x, iBlockLength, iHopLength)

# number of results
iNumOfBlocks = xBlocks.shape[0]

# compute time stamps
t = (np.arange(0, iNumOfBlocks) * iHopLength + (iBlockLength / 2)) * f_s

# allocate memory
vzc = np.zeros(iNumOfBlocks)

for n, block in enumerate(xBlocks):
    # calculate the zero crossing rate
    vzc[n] = 0.5 * np.mean(np.abs(np.diff(np.sign(block))))
```

(b) Python

Figure 3.49: Implementation of ZeroCrossingRate.

crossing rate is based on the assumption that the input signal has an arithmetic mean of approximately 0. An implementation of the zero crossing rate is shown in Fig. 3.49.

Figure 3.50 shows the zero crossing rate for an example signal. As mentioned above, it is high for noisy parts and low for tonal parts. Its usability for fundamental frequency detection (see Sect. 7.3.3.1) is indicated by the long constant values for constant pitches.

The zero crossing rate has been used for both measuring the noisiness of a signal and estimating its fundamental frequency by assuming a sinusoidal input signal and then relating the number of zero crossings directly to the fundamental frequency. To improve the robustness of such attempts, the input signal can be low-pass filtered to suppress high-frequency content. The cut-off frequency of the low-pass filter then should be chosen as low as the highest expected fundamental frequency to ensure maximum suppression of high-frequency content.

3.6.14 Feature Learning

The low-level features described above are only a subset of the vast set of hand-crafted and custom-designed features that have been successfully used for many years. As classifiers grew more powerful over time, researchers started to identify problems with this expert-driven approach. Humphrey et al. argued, for example, that “hand-crafted feature design is neither scalable nor sustainable” and that “shallow processing architectures struggle to describe the latent complexity of real-world phenomena” and called for data-driven feature design, known as *Feature Learning* [HBL13].

The advantage of using features learned from the data is that an algorithm might be able to identify and extract hidden relevant attributes in the data which might not be represented in low-level features. This might

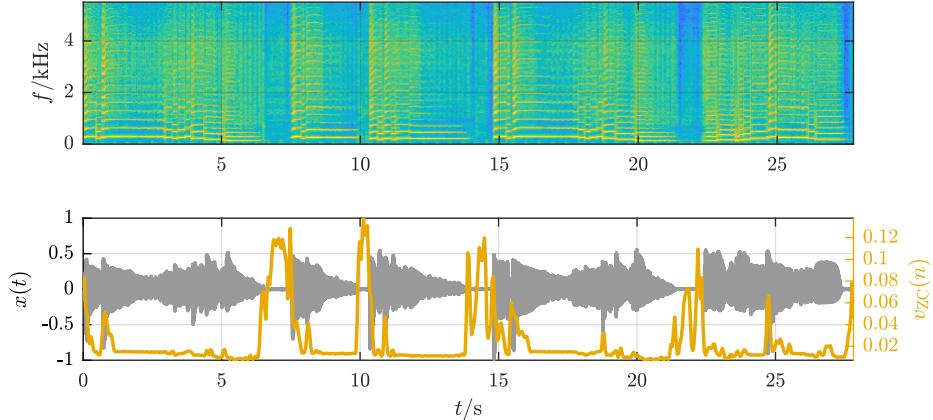


Fig. Gen.: plotFeatures.m

Figure 3.50: Spectrogram (top), waveform (bottom background), and feature zero-crossingrate (bottom foreground) of a saxophone signal.

be even true for randomly projected features: a randomly initialized untrained neural network may have the power to project the input data into a feature space that enables successful inference [RWK⁺20]. A random projection will often be outperformed, however, by a projection learned from relevant data. Feature learning methods can be categorized into unsupervised and supervised approaches, depending on whether unlabeled data or labeled data is used for training.

Unsupervised approaches to feature learning have been extensively explored in the late 2000s and early 2010s. Sparse coding, which attempts to learn a high-dimensional sparse representation of audio data via a dictionary learning approach, gained some popularity by learning features from spectrogram-like representations of the data [GRKN07, HJKL11, MM12]. Other researchers proposed to learn features from spectrograms with a k-means algorithm [SWK08, WR12]. These and other methods for feature learning have also been evaluated in comparison [NHSS12, PBD16], often with the result that deep neural architectures [SD14] generally seem to exhibit superior performance. Typical architectures include Deep Belief Networks [LPLN09, HE10, DBS11] or auto-encoders [WL18a].

The current trend goes towards supervised approaches for feature learning, a category that covers essentially all end-to-end networks. Due to the integration of feature extraction and inference in modern networks, the term feature learning is not frequently used anymore, although the concept of extracting a compressed, meaningful representation from the audio input remains unchanged. One aspect related to the concept of feature learning is *transfer learning* [Ben12]. In this case, a network is trained on one task and subsequently the learned representation is used for a different albeit related task. Choi et al., for example, trained a network for an auto-tagging task and then successfully used the embedded representation of the trained network as a feature input to a classifier for music genre classification [CFSC17]. Other widely used audio features (also “embeddings” or “representations”), all extracting by very deep architectures trained on large amounts of data, are VGGish [HCE⁺17], L3 [CWSB19], and musicnn [PS19]. Such audio features have been successfully transferred to tasks as different from the training data as music instrument classification [GSL19].

Despite the generally superior performance of features learned by deep architectures, there exist potential drawbacks and arguments against them. First of all, feature learning is usually not applicable if only small amounts of data are available, as most algorithms are data hungry. Second, learned features are often entangled and hard to interpret representations and thus not useful for an analysis of important factors. Last but not least, Krig points out that a learned feature is not necessarily as impartial as the data-driven approach might imply, as “DNN architectures and training methods are very handcrafted, and rely on several ad hoc design assumptions” [Kri16].

3.7 Feature Post-Processing

The result of the feature extraction process is a series of feature values that can — dependent on the use case — be processed, transformed, and selected.

It is quite common to compute a large number of features. Formally, they can be represented in a feature matrix:

$$\begin{aligned} \mathbf{V} &= [\mathbf{v}(0) \ \mathbf{v}(1) \ \cdots \ \mathbf{v}(\mathcal{N}-1)] \\ &= \begin{bmatrix} v_0(0) & v_0(1) & \cdots & v_0(\mathcal{N}-1) \\ v_1(0) & v_1(1) & \cdots & v_1(\mathcal{N}-1) \\ \vdots & \vdots & \ddots & \vdots \\ v_{\mathcal{F}-1}(0) & v_{\mathcal{F}-1}(1) & \cdots & v_{\mathcal{F}-1}(\mathcal{N}-1) \end{bmatrix} \end{aligned} \quad (3.79)$$

with the number of rows being the number of features \mathcal{F} and the number of columns being the number of blocks \mathcal{N} . Each vector $\mathbf{v}(n)$ consists of \mathcal{F} feature values at block n and will be referred to as an *observation*.

3.7.1 Derived Features

In some case it can be beneficial to process the extracted features. The resulting new “derived” features do not have to replace the original features; they can be added as additional features. The most commonly applied processing options are filters, especially both high-pass filters and low-pass filters. Filtering the feature series might help to remove irrelevant information. For example, in some cases the detection of (sudden) changes of feature values is of special interest as it may mark the start or end of important segments such as note onsets and structural boundaries. A simple way of analyzing these changes is to compute the difference between consecutive feature results (which would be called *derivative* if it were a continuous function):

$$v_{j,\Delta}(n) = v_j(n) - v_j(n-1). \quad (3.80)$$

The resulting series $v_{j,\Delta}(n)$ is either one value shorter than the corresponding series $v_j(n)$ or an appropriate initialization for $v_j(-1)$ has to be defined. The time stamp $t_{s,\Delta}(n)$ of $v_{j,\Delta}(n)$ is

$$t_{s,\Delta}(n) = \frac{t_s(n) + t_s(n-1)}{2} \quad (3.81)$$

with $t_s(n)$ being the time stamp of $v_j(n)$.

Computing the derivative has the character of a high-pass filter; it is also common to do the opposite, namely to smooth out $v_j(n)$ with a low-pass filter. This allows to identify long-term variations of the feature more easily. It is generally beneficial if the used filter has a zero phase or linear phase response in order to retain correct timing properties, therefore either an MA filter (see Sect. A.2.6.1) can be used or any Infinite Impulse Response (IIR) filter applied twice forward and backward on the series to produce the low-pass filtered series $v_{j,LP}(n)$ (see Sect. A.2.7).

Other derived features might involve computing linear or non-linear combinations of the existing features. The usage of such derived features is frequently met with only limited success as most of the information of interest can already be found in the original features. Furthermore, increasing the number of features and thus the dimensionality of the observations can potentially have negative impact on the classifier performance.

3.7.2 Normalization and Mapping

The normalization of features is a trivial but crucial step for many inference systems. As different features different ranges and distributions are combined into observation vectors, proper feature scaling ensures that one feature does not have outsized impact on the training or the inference result. Consider two identical features with identical distribution except for an amplitude scale factor λ . Each observation contains the two features. When computing the squared Euclidean distance, the second dimension's distance will have the weight λ^2 while

the first dimension will be weighted with 1. Large λ will thus let the second feature dominate the distance while small λ will render the second dimension superfluous.

A common approach to normalize features if they all have *symmetric* distributions with *identical shape* is to remove their mean value and scale them to a standard deviation of 1 (see, e.g., [AH01]):

$$v_{j,N}(n) = \frac{v_j(n) - \mu_{v_j}}{\sigma_{v_j}}. \quad (3.82)$$

This normalization is referred to as *z-score normalization*. Note that the resulting normalized features are not bounded. Z-score normalization ensures similar impact of individual features as long as the features are distributed symmetrically and similar to a Gaussian distribution. However, if the distributions of different features are different and not Gaussian, then this normalization has to be applied with care.

A second common feature normalization is the so-called *min-max normalization* which normalizes each feature to a range $[0; 1]$:

$$v_{j,N}(n) = \frac{v_j(n) - \min(v_j)}{\max(v_j) - \min(v_j)}. \quad (3.83)$$

This normalization results in bounded and positive features, but infrequent Outliers might impact the range.

Note that some literature will refer to these two standard approaches to feature scaling with different terminology: what is referred to here as z-score normalization is sometimes called standardization, and what is referred to here as min-max normalization might just be called normalization.

The normalization constants $\mu_{v_j}, \sigma_{v_j}, \max(v_j), \min(v_j)$ have to be estimated from the *training set*. The same (training) constants are then applied during inference. Extracting constants from the test set is meaningless as the system has to infer with exactly the same parameters as during training.

If the features have dramatically different and skewed distributions, the above normalizations might not be applicable. The normalization of slightly skewed features might improve when the median $Q_{v_j}(0.5)$ replaces μ_{v_j} (also in the computation of the standard deviation σ_{v_j}).

For more skewed features, the transformation of the feature to a target distribution (e.g., a Gaussian distribution) might help. Several approaches exist to transform a given distribution into a Gaussian distribution; widely used is the *Box-Cox transform* [BC64]. One example of this transform is

$$v^{(\lambda)} = \begin{cases} \frac{v^\lambda - 1}{\lambda}, & \lambda \neq 0 \\ \log(v), & \lambda = 0 \end{cases} \quad (3.84)$$

with the parameter λ to be estimated. The Box-Cox transform only considers a limited class of transformation functions and thus does not guarantee that any arbitrary distribution can be mapped to a Gaussian distribution.

There are also numerical methods of finding appropriate feature transformations. One example is the work of Albada and Robinson who transform arbitrary distributions to the normal distribution [AR07]. The transformation function for every feature has then to be stored numerically.

In many practical applications it is sufficient to transform selected features in a way that results only in roughly approximated Gaussian distributions. Only features failing a test for Gaussianity have to be subjected to a transformation. Statistical procedures to test a distribution for Gaussianity are, for example, the Kolmogorov-Smirnov test, the Lilliefors test, the Shapiro-Wilk test [SW65], and the Anderson-Darling test [AD52]. Thode gives a good introductory overview [Tho02]. Unfortunately, the result of these statistical tests is only of limited use in ACA because these tests nearly always tend to fail for large numbers of observations. In these cases it is more practical to compute both the skewness and possibly the kurtosis (see Sect. 3.6.3) of the features to determine how *Gaussian* their distribution is. As a rule of thumb, distributions with a skewness between -2 and 2 are not significantly skewed and can thus be assumed to be symmetric [MS01].

The normalization of multi-dimensional features requires special consideration and depends on both the feature characteristics as well as the specific use case. The following examples show the variety of approaches:

$$\begin{array}{c}
 \left[\begin{array}{ccccc} v_0(0) & v_0(1) & v_0(2) & v_0(3) & v_0(4) \\ v_1(0) & v_1(1) & v_1(2) & v_1(3) & v_1(4) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ v_{\mathcal{F}-1}(0) & v_{\mathcal{F}-1}(1) & v_{\mathcal{F}-1}(2) & v_{\mathcal{F}-1}(3) & v_{\mathcal{F}-1}(4) \end{array} \right] \\
 \underbrace{\qquad\qquad\qquad}_{\left[\begin{array}{c} \mu_0(0) \\ \sigma_0(0) \\ \mu_1(0) \\ \sigma_1(0) \\ \vdots \\ \mu_{\mathcal{F}-1}(0) \\ \sigma_{\mathcal{F}-1}(0) \end{array} \right]}
 \end{array}$$

Figure 3.51: Example aggregation with a texture window length of 4 with the arithmetic mean μ and the standard deviation σ

- MFCCs: Each coefficient is usually considered independent and therefore normalized independently as if they were different features.
- Pitch Chroma: Treating each pitch class independently would distort the pitch class relationships in the feature. The common choice is to normalize each observation (i.e., vector per block) independently and thus preserve pitch class relationships while sacrificing volume information over time. Thus, each 12-dimensional pitch chroma observation is usually either normalized to a sum of 1 (like a pitch distribution) or to a vector length of 1 (like a point in Cartesian pitch space) [EM11].
- (Mel) Spectrogram: a magnitude spectrogram is most commonly normalized to a range between 0 ad 1 by dividing it by its maximum value. There have been, however, also proposals to apply a localized normalization strategy [LSC⁺19].

3.7.3 Feature Aggregation

The feature matrix \mathbf{V} contains the feature value per block and we might treat each column as an observation vector to feed to the inference system. Practically, however, the comparably high time resolution leads to large variations of the feature values and make any inference such as classification unnecessarily hard. Therefore, it is common to combine multiple neighboring blocks into a windows (sometimes referred to as *texture window*), which might overlap in time. Each window covers a submatrix; in order to convert it to a single vector, statistical descriptors can be computed for each feature $v_j(n)$ per window. The most common statistical aggregations are via the arithmetic mean and the standard deviation, although any measure presented in Sect. 3.1.3 could be used. In case we have \mathcal{F} features and \mathcal{M} observations per window, for example, and compute both the arithmetic mean and the standard deviation of each feature, the resulting length of the new observation vector fed to the inference system will be $2\mathcal{F}$. Figure 3.51 shows the aggregation of one texture window of length 4 using the arithmetic mean and the standard deviation.

The resulting observations are sometimes referred to as *subfeature*. A relatively large number of possible meaningful subfeatures in the context of musical genre classification was evaluated by Mörchen et al. [MUTL06]. Note that the process can also be applied multiple times, leading to a hierarchical feature aggregation with decreasing time resolution as the hierarchy level increases. This hierarchical feature aggregation is somewhat similar to the repeated use of max-pooling layers often found in Convolutional Neural Networks (CNNs) with the difference that in this case the maximum is used as aggregation method (after convolution).

The typical length of the texture window depends on the task; it can vary from values below 0.5 s for quasi-real time inference over 15–30 s for various audio classification tasks up to the song length.

3.7.4 Feature Dimensionality Reduction

Although large numbers of features can easily be extracted from the audio data, it is unclear a priori which features will help the final inference (in the following, we will only refer to classifiers as a placeholder for other inference stages) of the ACA system.

While it might seem intuitive to assume that any classifier will benefit from more information at the input (i.e., from a higher input dimensionality of each observation), this is not necessarily true. In fact, increasing the input dimensionality can require a higher model complexity, which in turn increases the likelihood of *overfitting* to occur during the training. Overfitting means that the classifier starts to learn training set-specific characteristics which model the training set well but do not generalize anymore to the task [Haw04]. The classifier will then perform poorly on unknown input data. A higher input dimensionality might also increase the sparseness of the feature space which can make it harder to build a machine learning model; this is sometimes referred to as the *curse of dimensionality* [Bel72].

Depending on the classifier, the task complexity, and the amount of training data, it can be beneficial to reduce the feature dimensionality. As some classifiers are more robust against high-dimensional input dimensions than others, no clear recommendations can be made regarding the optimal number of features; often, the number of features is picked that maximizes validation accuracy. A typical feature vector dimensionality for traditional, non-DNN classifiers ranges from 20 to around 100, although there are many exceptions.

The goal when reducing the dimensionality of the feature set is removing both redundant and irrelevant information while retaining the (task-) relevant information. There are two different approaches to reduce the dimensionality of the feature space:

- *feature subset selection* to discard specific features, and
- *feature space transformation* to transform the features to a lower dimensional space.

In the first case the most promising feature subset is chosen; in the latter case only those dimensions in the transformed feature space are discarded that contribute the least information for the target application.

Note that the requirement of feature dimensionality reduction strongly depends on the classification algorithm used; furthermore, it is still under discussion whether complex methods of dimensionality reduction really outperform simple ones [FF06a].

3.7.4.1 Feature Subset Selection

The aim of *feature subset selection* is to reduce the number of used features by discarding the least powerful features. Formally, the available feature set

$$\mathcal{V} = v_j |_{j=1, \dots, \mathcal{F}} \quad (3.85)$$

should be reduced to the feature subset

$$\mathcal{V}_s = v_j |_{j=1, \dots, \mathcal{F}_s} \quad (3.86)$$

with $\mathcal{F}_s < \mathcal{F}$; the subset is chosen to optimize a given objective function $J(\mathcal{V}_s)$.

Feature selection algorithms are called *wrapper methods* if the objective function is the classifier itself and *filter methods* if the objective function $J(\mathcal{V}_s)$ is independent of the classification system used. Filter methods select features based on properties a good feature set is presumed to have and are usually computationally less expensive than wrapper methods.

This section will only provide a short introduction to the most common approaches to feature subset selection. More in-depth surveys of this topic have been published by Guyon and Elisseeff [GE03] and Cantú-Paz et al. [CPNK04].

Examples for wrapper methods are

- *Brute Force Subset Selection*

The most obvious way of finding the optimal feature subset is to compute the classification accuracy for all possible combinations of features and to select the subset that performed best. The disadvantage of

```
% initialize
selectedFeatureIdx = [];
unselectedFeatures = ones(1,iNumFeatures);
AccPerSubset = zeros(1,iNumFeatures);

% iterate until target number of features is reached
for (i = 1:iNumFeatures2Select)
    acc = zeros(1,iNumFeatures);

    % iterate over all features not yet selected
    for f = 1:iNumFeatures
        if (unselectedFeatures(f) > 0)
            % accuracy of selected features plus current feature f
            acc(f) = ToolLooCrossVal(FeatureMatrix([selectedFeatureIdx f],:),ClassIndices);
        else
            acc(f) = -1;
            continue;
        end
    end

    % identify feature maximizing the accuracy
    % move feature from unselected to selected
    [maxacc,maxidx] = max(acc);
    selectedFeatureIdx = [selectedFeatureIdx, maxidx];
    unselectedFeatures(maxidx) = 0;
    AccPerSubset(i) = maxacc;

```

(a) Matlab

```
% initialize
selectedFeatureIdx = [];
unselectedFeatures = ones(1,iNumFeatures);
AccPerSubset = zeros(1,iNumFeatures);

% iterate until target number of features is reached
for (i = 1:iNumFeatures2Select)
    acc = zeros(1,iNumFeatures);

    % iterate over all features not yet selected
    for f = 1:iNumFeatures
        if (unselectedFeatures(f) > 0)
            % accuracy of selected features plus current feature f
            acc(f) = ToolLooCrossVal(FeatureMatrix([selectedFeatureIdx f],:),ClassIndices);
        else
            acc(f) = -1;
            continue;
        end
    end

    % identify feature maximizing the accuracy
    % move feature from unselected to selected
    [maxacc,maxidx] = max(acc);
    selectedFeatureIdx = [selectedFeatureIdx, maxidx];
    unselectedFeatures(maxidx) = 0;
    AccPerSubset(i) = maxacc;

```

(b) Python

Figure 3.52: Example code for Sequential Forward Selection.
implement python

this approach is that the number of subsets to test, i.e., to train and evaluate, will be $2^{\mathcal{F}}$. This renders this method impractical for large numbers of features \mathcal{F} .

- *Single Variable Classification*

A simple feature ranking can be obtained by calculating the classification accuracy for each individual feature. This enables the identification of features performing very poorly individually. While discarding the features that perform worst seems to be an intuitive solution, there are two problems with selecting or discarding features this way:

1. The feature ranking contains no information on the *correlation* of two or more features. Consider the case of two features with identical values. They will both have the same ranking which is possibly high, but leaving both in the selected feature subset cannot improve classifier performance (and might even harm it in the case of simple classification algorithms).
2. The feature ranking contains no information on the *combined usefulness* of features. A feature that adds no information individually might be able to add information in combination with other features.

- *Sequential Forward Selection*

Sequential forward selection starts with an empty subset of features. In the first iteration, it considers all feature subsets with only one feature (compare the *single variable classification* approach). The subset with the highest classification accuracy is used as the basis for the next iteration. The iterative algorithm can be structured into the following processing steps:

1. Start with an empty feature subset $\mathcal{V}_s = \emptyset$.
2. Find the one feature v_j not yet included in the feature subset that maximizes the objective function

$$v_j = \underset{\forall j | v_j \notin \mathcal{V}_s}{\operatorname{argmax}} J(\mathcal{V}_s \bigcup v_j). \quad (3.87)$$

3. Add feature v_j to \mathcal{V}_s .
4. Go to step 2 and repeat the procedure until the required number of features has been selected or the required classification accuracy has been reached.

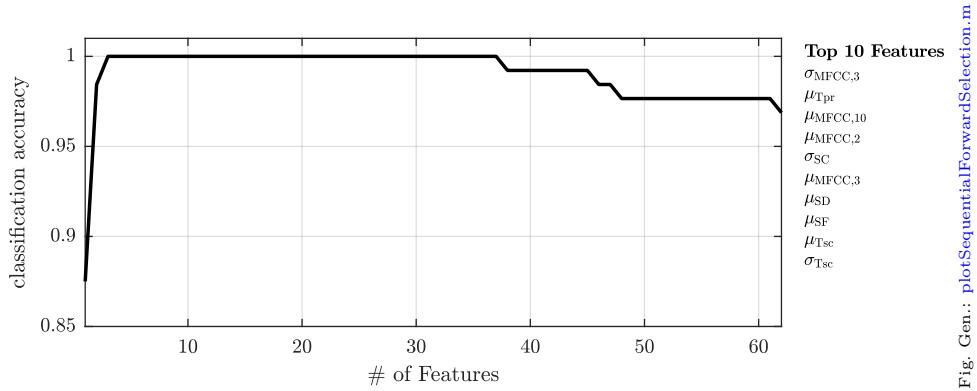


Figure 3.53: Accuracy over number of features selected by Sequential Forward Selection.

Figure 3.52 shows example code for sequential forward selection. In this case, the accuracy per feature is estimated with a Nearest neighbor classifier through leave-one-out crossvalidation (see below in Sect. 4). The accuracy over the number of selected features for a simple 2-class speech/music classification problem is shown in Fig. 3.53.

- *Sequential Backward Elimination*

Sequential backward elimination works in an analogous way to sequential forward selection but starts with a full subset of features and iteratively removes features from the subset. It is computationally less efficient than sequential forward selection. This is particularly true for large feature sets. Sequential backward elimination can be argued to give better results since sequential forward selection does not assess the importance of features in combination with other not yet included features. **remove backward elimination as not that important?**

There exist many filter methods for finding variable rankings. The methods include, for example, chi-square statistics or any arbitrary separability measure. One common example of a filter yielding an implicit feature ranking is *Principal Component Analysis (PCA)*. The concept of PCA is summarized in Appendix C. It can be used for feature selection by examination of the transformation matrix \mathbf{T} . The idea is to keep the features with major influence on the principal components and to eliminate features with major influence on the components with low variance.

A simple rule for feature elimination is to start with the component with the smallest eigenvector, discard the feature which contributes most to this component, proceed to the next-smallest eigenvector, and repeat the procedure until all features have been ranked. Then, an arbitrary number of features can be discarded.

3.7.4.2 Feature Space Transformation

The objective of *feature space transformation* is to reduce the number of used features by transforming them into a lower dimensional space.

A frequently used feature space transformation is the *Principal Component Analysis (PCA)*. The PCA, as explained in Appendix C in detail, does not reduce the dimensionality of the data by itself. However, the new dimensions are sorted according to the variance they contribute to the data which is often interpreted as a measure of importance. Discarding the components that account for low variance is therefore a frequently used approach. A widely used systematic criterion to decide how many components can be discarded is based on the eigenvalue. This approach is based on the assumption that every component with an eigenvalue lower than 1 can be discarded. This criterion is equivalent of a threshold of $1/\mathcal{F}$ for the relative variance for which a component accounts. In many cases, this criterion leaves more components in the data set than useful. A slightly more “hands-down” approach is to identify either the index after which the eigenvalues are significantly lower or the index after which eigenvalues tend to be very similar to each other.

Other transformation methods can be used for feature space transformation but will not be explained in detail in this book. Typical approaches that can be found in the literature are *Independent Component Analysis*

(ICA) and *Singular Value Decomposition (SVD)*. *Linear Discriminant Analysis (LDA)* also transforms the feature space, however, the number of output components cannot be chosen freely in this case. The main distinction of LDA and PCA is that PCA maximizes the variance and LDA maximizes class separability.

The disadvantage of using transformations for dimensionality reduction is that the transformed features cannot be interpreted as easily as the original features since the transformed features are combinations of the original features. As all input features have to be computed regardless of their importance, feature space transformation methods cannot be used to reduce the workload.

3.8 Exercises

3.8.1 Questions

1. How are (fundamental) frequency f_0 and period length T_0 of a periodic signal related?
2. What are the frequencies of the first 5 harmonics of a periodic signal with a fundamental frequency $f_0 = 220 \text{ Hz}$?
3. Is the signal $x(t) = A \cos(2\pi t w(t))$ (with A constant, $w(t)$ a Gaussian random variable, and t the time) a deterministic or a random signal?
4. Is it possible to compute the Fourier Series coefficients of a random white noise input signal?
5. Consider a periodic signal with period length 4. What is the 0th Fourier coefficient for this signal given that the area under the signal curve for one period is 8.
6. A continuous random variable $X \in [0, 2]$ has a uniform distribution. What is the mean of the random variable? What is the variance of this variable?
7. Sketch the Fourier Transform of two delta impulses located symmetrically around 0?
8. What happens to the Fourier Transform of a signal if it is multiplied (in the time-domain) with a cosine with non-zero frequency?
9. Given that the Fourier transform of $x(t)$ is $X(j\omega)$, what is the Fourier transform of $x(t - T_1)$?
10. Discuss the differences between a Mel-spectrogram and a 'normal' spectrogram.
11. Given a signal $x(t) = A \cos(2\beta ft)$ ($A = 0.5, f = 440$ for $0 \leq t < 1$ and $A = 0.9, f = 440$ for $1 \leq t < 2$). Which of the following features would be the best choice to detect the change in the amplitude?
 - Spectral Centroid
 - Spectral Flux
 - Spectral Spread
 - Spectral Skewness
12. Choose the correct sentence completion. The Spectral Centroid has been shown to correlate to some degree to the human perception of
 - (i) roughness,
 - (ii) brightness,
 - (iii) happiness,
 - (iv) loudness.
13. What is the Spectral Centroid of a zero input signal?

14. What is the Spectral Flux of an infinite sinusoidal? Assume the sine frequency to match a bin frequency of the spectrum.
15. One row of a feature matrix with mean $\mu = 5$ and standard deviation $\sigma = 2$ is passed through a z-score normalization block. What is the mean and standard deviation of the resulting row?
16. A feature matrix \mathbf{F} of size $m \times n$ ($m = 5$ is the number of features per block and $n = 2000$ is the number of blocks of the input audio) are aggregated over the entire feature matrix. If we use mean, standard deviation, and range as the three aggregating metrics, what will be the dimension of the resulting aggregated feature matrix?
17. Summarize typical problems with high-dimensional data in machine learning.
18. We apply a PCA to a feature matrix \mathbf{F} of size $m \times n$ ($m = 5$ is the number of features per block and $n = 2000$ is the number of blocks of the input audio) to get a transformed feature matrix \mathbf{F}_{PCA} . Given the equation of PCA transformation is given as $\mathbf{F}_{\text{PCA}} = \mathbf{T}^T \mathbf{F}$, what is the dimension of the transformation matrix \mathbf{T} ?
19. We apply a PCA to a feature matrix \mathbf{F} of size $m \times n$ ($m = 5$ is the number of features per block and $n = 2000$ is the number of blocks of the input audio) to get a transformed feature matrix \mathbf{F}_{PCA} . Given the equation of PCA transformation is given as $\mathbf{F}_{\text{PCA}} = \mathbf{T}^T \mathbf{F}$, what is the dimension of the transformation matrix \mathbf{F}_{PCA} ?

3.8.2 Assignments

Chapter 4

Inference

The second stage of the ACA system, the *inference*, takes the extracted features and maps them into a domain both usable and comprehensible by humans. In other words, it interprets the feature data and maps it to meaningful meta-data, such as a class label for the example in Fig. 3.19 (speech, chamber music, pop) or the pitch of a melody. This inference system can be an expert-defined algorithm, a classifier, or a regression algorithm. The more condensed and the more meaningful the features are, the less powerful the inference algorithm has to be and vice versa: a raw feature representation close to the audio sample requires a sophisticated inference approach.

Historically, rule-based expert systems used to be widely used. Especially for simple tasks, it is possible to use knowledge to implement a set of algorithmic rules solving a specific problem. Part of the algorithms presented in this book will follow such an approach. For the majority of non-trivial (analysis) tasks, however, data-driven Machine Learning (ML) systems have shown superior performance. Such systems are based on more or less generic algorithms using data to adapt to specific use cases by parametrizing its algorithm adaptively. In order to do this properly, sufficient training data is required (see Sect. 5).

This chapter is not intended as a formal introduction into machine learning and classifiers. Rather, it focuses on giving a high-level overview of the general principles with the occasional detail to address potential practical issues. There exists a lot of introductory and advanced literature on the topic; see, for example, Bishop [Bis06], Shalev-Shwartz and Ben-David [SSKS04], or Duda et al. [DHS00]. Goodfellow is a book that focuses on neural networks [GBC16] and Ng's book provides helpful practical tips for deep learning [Ng18].

add section on overfitting? with Fig. 4.1

4.1 Classification

The simplest, most intuitive classifier is just a threshold: if a feature value is higher than a threshold ϵ choose class 1, otherwise class 0. Figure 4.2 gives an example for this: the peak envelope is extracted from a drum loop as the feature, which is the thresholded at G . If the feature is higher than the threshold, we classify this block

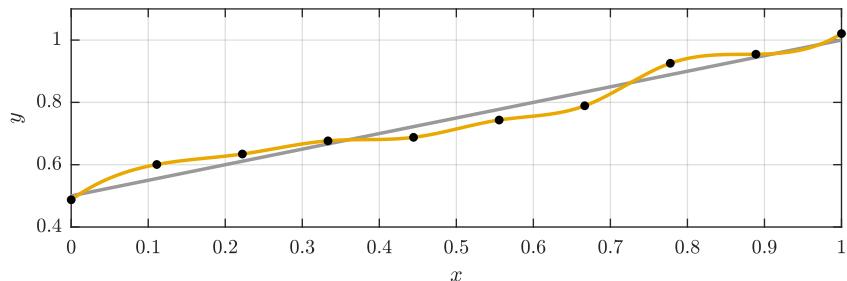


Fig. Gen.: `plotOverfitting.m`

Figure 4.1: Example visualization of an overfitted model exactly matching the noisy observations vs. a correct model.

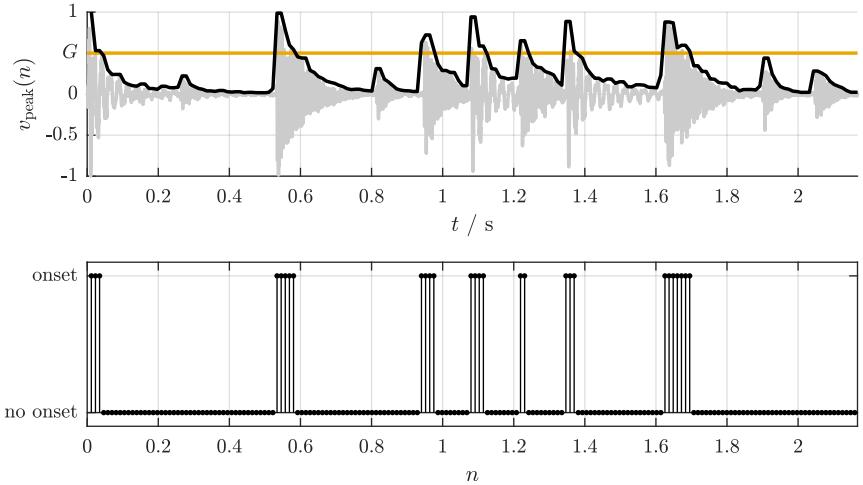
Fig. Gen.: `plotThresholdClassification.m`

Figure 4.2: Example for classifying the blocks of a drum loop into the classes onset and no onset based on the peak envelope of the signal and a simple threshold.

as class 1 (onset), otherwise as class 2 (no onset). The value of the threshold determines the ratio between class 1 and 2 detections. Note that this approach is far from perfect for onset detection, as several onsets have been not detected in this snippet. For more details on approaches to onset detection please refer to Sect. 9.3.

A data-driven system derives this threshold G from the data itself. If the system is more sophisticated, it might generalize to a multi-dimensional space with non-linear thresholds. In other words, it learns what combination of feature values are common for each class and how to differentiate between classes given these feature values. Figure 4.3 shows a so-called scatter plot for two classes (speech and music) represented by two features (two-dimensional feature space). It can be seen that for this data, neither feature is able to separate the two classes well, although the RMS feature seems to work slightly better as the distribution of feature values has less overlap between classes. The better the available features separate the classes, the less sophisticated the classifier model has to be. This visualization also emphasizes the importance of the so-called training data set; if the class distributions used to parametrize the classifier do not match the data sent into the classifier when testing, the classifier will not perform well.

A common basic classifier is the Nearest Neighbor classifier [FH51]. While training, it stores the location of each data point in the feature space with its corresponding class label. When queried with a new and unseen feature vector, the distance to every single previously stored training vector is computed; the final result is the class label of the closest vector, the nearest neighbor. An often-used variant is the Knn (k Nearest Neighbor) classifier, which uses not only the nearest neighbor as reference but the nearest k training data points. As the estimated class label is then the majority of the labels of the k nearest vectors, k is usually an odd number (e.g., 3 or 7). Figure 4.4 visualizes how the classification result might change for different k . In the case of more than two classes, it might occur that there is no clear class majority. This case can be handled by implementing fallback solutions such as taking into account the average distance per class or reducing k iteratively until a majority is found.

As the kNN classifier is based on distance calculations, most commonly the Euclidean distance, feature normalization can have a major impact on the classifier performance. Its simple implementation with few tunable parameters is an advantage. The dimensionality of the feature vector, however, must not be too large to avoid the effects of the curse of dimensionality, often requiring feature dimensionality reduction methods (see Sect. 3.7.4). As Nearest Neighbor classifiers learn by memorization, the amount of data that needs to be stored for the model can be large (all training data points). The distance computation to each stored data point can also make it expensive to execute. This makes kNN a less practical approach for large datasets. Figure 4.5 shows a simple implementation of a kNN classifier.

The vast majority of classifiers do not store all individual training data points but build a model from the

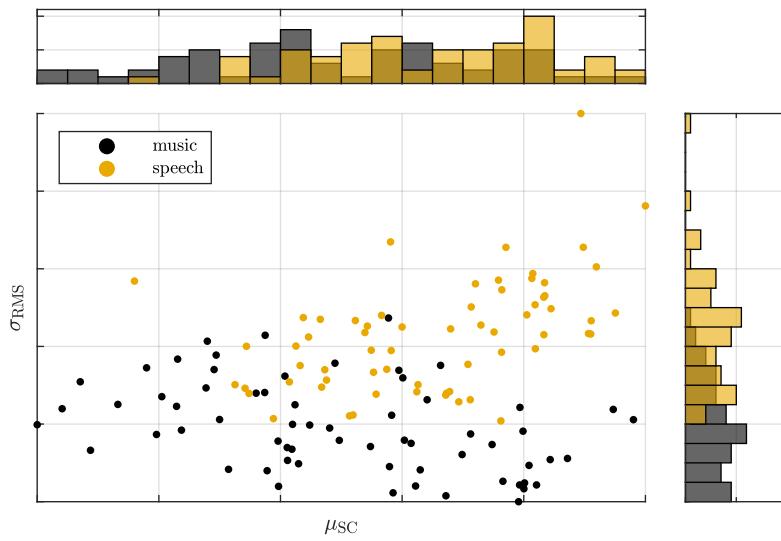
Fig. Gen.: `plotFeaturespace.m`

Figure 4.3: A music/speech dataset visualized in a two-dimensional feature space (x-axis: average spectral centroid, y-axis: standard deviation of RMS).

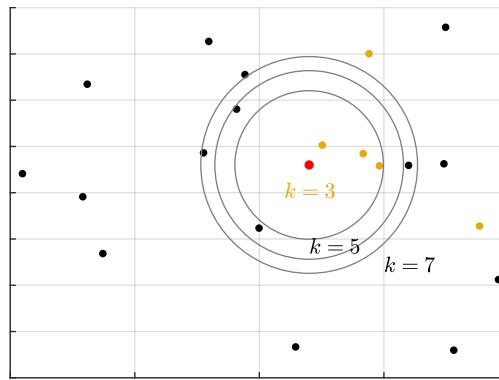
Fig. Gen.: `plotKnn.m`

Figure 4.4: Nearest neighbor classification of a test data point (red) with different k . The result is the same for $k = 3$ and $k = 5$, but changes for $k = 7$.

```
% compute distances to all training observations
d = computeEucDist(TestFeatureVector, TrainFeatureMatrix); % compute distances to all training observations
d = computeEucDist(TestFeatureVector, TrainFeatureMatrix);

% sort the distances to find closest
[dummy,idx] = sort(d); % sort the distances to find closest
[dummy,idx] = sort(d);

% pick the majority of the k closest training observations
% note that for multi-class problems and even k, this needs to be
% refined
class = mode(TrainClassIndices(idx(1:k))); % pick the majority of the k closest training observations
% note that for multi-class problems and even k, this needs to be
% refined
class = mode(TrainClassIndices(idx(1:k)));
```

(a) Matlab

(b) Python

Figure 4.5: Implementation of a kNN classifier.
implement python

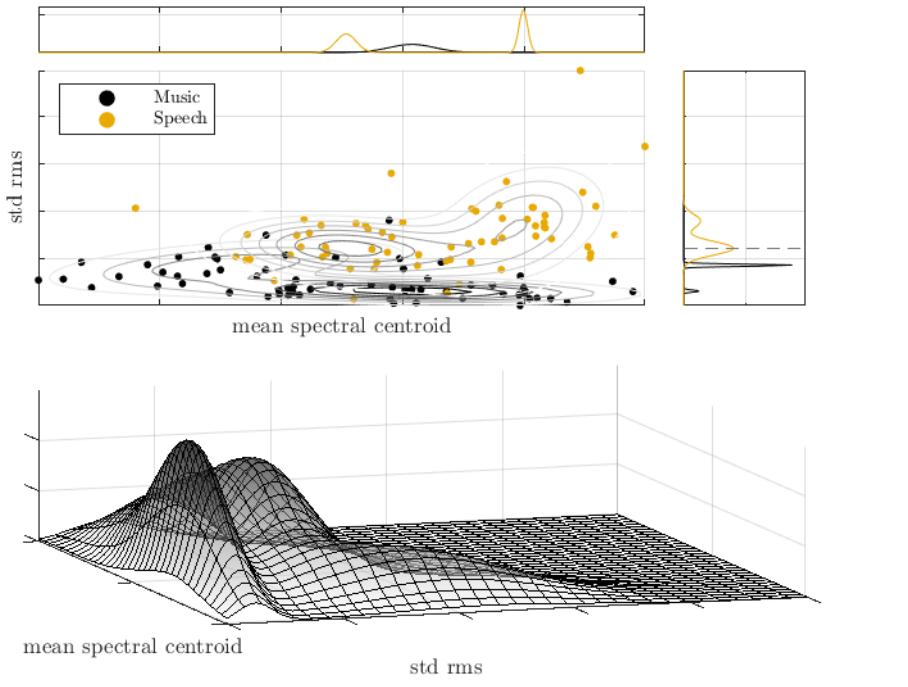
Fig. Gen.: `plotGmm.m`

Figure 4.6: The feature space from Fig. 4.3 and a corresponding Gaussian Mixture Model with two Gaussians per class **regenerate this with two 3D colors and remove code dependency to statistics toolbox.**

data. Classification with Gaussian Mixture Models (GMMs), for example, assumes that the data per class can be sufficiently modeled with multiple superposed Gaussian distributions. Figure 4.6 shows the two mixtures of two Gaussian distributions per class for the data example presented above. The distributions have been fitted with the Expectation Maximization (EM) algorithm [DHS00]. In this classification scenario, each test data point is assigned the label of the mixture model with the higher value at this location. The value can also be interpreted as the likelihood.

The Support Vector Machine (SVM) is a classifier that has shown good performance in a large variety of tasks and is often the classifier of choice if deep learning is not suitable. It projects the data points into a higher-dimensional space and estimates a separating hyperplane that balances the margin between the classes (as large as possible) with the classification accuracy on the training data (as high as possible) [BGV92]. SVMs also allow for more complex classification scenarios by applying a non-linear kernel to the data before estimating the separating hyperplane. This allows to go beyond linear classification even in the high-dimensional projected space.

The Random Forest classifier utilizes concepts of ensemble learning [Bre01]. It is referred to as 'forest' because it utilizes a (large) number of Decision Tree classifiers, and it is called 'random' as it is trained with randomly selected subsets of both data and features. A Decision Tree is a comparably simple classifier doing nested binary classifications to form nodes containing a large number of data points. Different trees will be differently shaped as the derived decision thresholds depend on the provided training data and on the provided subset of features. Thus, they might or might not lead to the same classification result. The Random Forest aggregates all individual tree results and returns the majority vote. Generally, such an ensemble methods tend to be less sensitive to the training data and generally very robust.

The Hidden Markov Model (HMM) commonly utilizes a GMM, but deserves a special mention since it does not only classify single instances individually, but models the probability of a sequence of observations. This is most frequently accomplished with the Viterbi algorithm (see Sect. 7.6) [Rab89].

Deep Neural Networks (DNNs) have shown superior performance in nearly all tasks they have been applied to during the past decade. Nearly all state-of-the-art systems in the field of audio analysis are nowadays deep

networks. Despite this general superiority, however, they are not always the best choice for all tasks. Many networks have a high complexity with a large number of hyper-parameters; not only does this complexity require large amounts of training data that might not be easily available, it also increases the likelihood of overfitting and many other training issues. An important distinction between DNNs and “traditional” classifiers is that the feature extraction and classification are not easily separated — while the classifiers above mostly benefit from a powerful feature representation at their input, a DNN is supposed to learn the relevant features from a raw representation of the data. This removes the feature design and engineering from the list of tasks; while this removes the expert knowledge utilized in the feature design, it allows to learn features from the data that can capture information not included in the custom-designed features. Neural networks come in a large variety of different architectures [GBC16]. As of writing this text, a very common classification setup for audio classification consists of a CNN with a logmel-spectrogram input. The input representation preserves the majority of the information of the audio signal, while the convolutional architecture allows for efficient internal data aggregation with a degree of translation invariance. Other architectures (e.g., various forms of recursive neural networks) have also been used successfully in audio analysis.

4.2 Regression

In regression analysis the (statistical) relationship between a dependent variable (output) and one or more independent variables (features) is modeled. Similar to classifiers, this model maps the input to the output. The difference to classifiers is that a value is predicted as opposed to a class label. Thus, regression models are commonly used for tasks that need the estimation of a continuous value given a set of features. That makes regression analysis a good fit for, for example, a standard mood classification task which tries to estimate the values of arousal and valence on two scales. Another example could be the grading of student music performances.

Linear regression is a simple approach to regression that assumes a linear relationship between the input features and output. In its simplest form, both the input and the output values are one-dimensional (not vectors). Given a set of \mathcal{R} feature observations (also: independent variables) v and corresponding target values (also: dependent variables) y , the goal is to estimate the slope m and offset (also: bias, intercept) b of a straight line

$$\hat{y}(r) = m \cdot v(r) + b \quad (4.1)$$

that minimizes the error between the model and the target given the features. The most common objective function to be minimized is the Mean Squared Error (MSE) (see Eq. (6.7)). Minimizing the MSE between the model and the targets leads to the following solution:

$$b = \mu_y - m \cdot \mu_v \quad (4.2)$$

$$m = \frac{\sum_{r=0}^{\mathcal{R}-1} (y(r) - \mu_y) \cdot (v(r) - \mu_v)}{\sum_{r=0}^{\mathcal{R}-1} (v(r) - \mu_v)^2} \quad (4.3)$$

The detailed derivation of these results can be found in the Appendix D.

Figure 4.7 shows two example results; a linear model with a good fit between feature (RMS) and target (peak envelope) on the left and a linear model with a bad fit between feature (RMS) and target (spectral centroid) on the right.

In the case of multiple independent variables v , the model takes the form

$$\hat{y}(r) = m_1 v_1(r) + m_2 v_2(r) + \dots + m_f v_f(r) + b. \quad (4.4)$$

There exist many other regression approaches, and most of them are more sophisticated and powerful than the simple linear regression introduced above. Two examples for nonlinear regression are Support Vector Regression (SVR) (compare SVM) and a DNN with a sigmoid output layer.

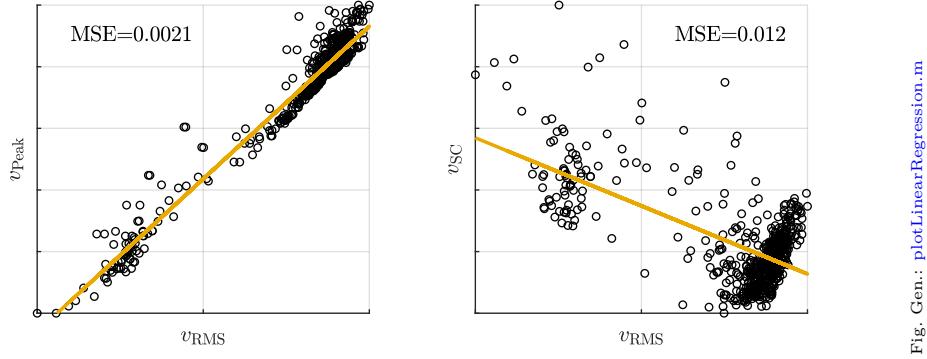


Figure 4.7: Linear Regression of two feature/target pairs, left: RMS and peak envelope, right: RMS and spectral centroid.

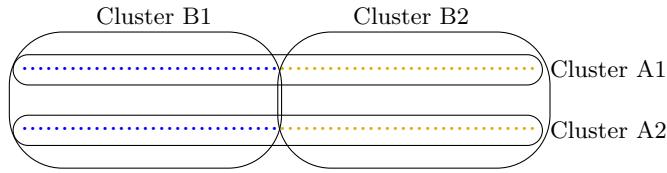


Figure 4.8: Example illustrating the importance of the similarity definition for clustering: the observations can be intuitively grouped according to spatial separation (clusters A) or according to color (clusters B)

4.3 Clustering

Algorithms for *clustering* are usually unsupervised and are especially useful for exploratory analysis. The goal of clustering is to group observations in a way that similar observations end up in the same group while dissimilar observations end up in other groups [SSKS04]. Obviously, the definition of similarity plays a major role in what is being clustered together: Shalev-Shwartz and Ben-David illustrate this with the example in Fig. 4.8 [SSKS04]: given the observations represented as dots, we might be inclined to either group them in spatially separated clusters by lines (Clusters A1 and A2) or by color (Clusters B1 and B2). This emphasizes the importance of the distance or similarity metric used for clustering.

K-means clustering is a simple and popular method to identify clusters in data. There exist multiple variants; here, we will focus on a an objective function that minimizes the distances of the data points to the mean of each cluster. Note that, like many clustering methods, we require the target number of clusters K for initialization. The iterative clustering has the following algorithmic steps:

1. *Initialization*: randomly select K observations from the data set as initialization.
2. *Update*: compute the mean for each cluster.
3. *Assignment*: assign each observation to the cluster with the mean of the closest cluster.
4. *Iteration*: go to step 2 until the clusters converge.

Figure 4.9 shows an implementation of this algorithm.

Figure 4.10 shows the results of this implementation applied to the observations in the feature space shown in Fig. 4.3 for three different stages. The algorithm converges after 8 iterations. Note that in this case, the clustering is unsuccessful if the goal is to match the two ground truth classes music and speech.

4.4 Distance and Similarity

Distance measures are integral parts of many algorithms introduced in this book. Whether we want to classify a new observation with the K-Nearest Neighbor (KNN) classifier (see Sect. 4.1), find clusters in the feature

```

for i=1:numMaxIter
    prevState = state;

    % update means
    state = computeClusterMeans(FeatureMatrix,clusterIdx,k);

    % reinitialize empty clusters
    state = reinitState(state, clusterIdx, k, range);

    % assign observations to clusters
    clusterIdx = assignClusterLabels(FeatureMatrix,state);

    % if we have converged, break
    if (max(sum(abs(state.m-prevState.m)))==0)
        break;
    end
end

```

(a) Matlab

```

for i=1:numMaxIter
    prevState = state;

    % update means
    state = computeClusterMeans(FeatureMatrix,clusterIdx,k);

    % reinitialize empty clusters
    state = reinitState(state, clusterIdx, k, range);

    % assign observations to clusters
    clusterIdx = assignClusterLabels(FeatureMatrix,state);

    % if we have converged, break
    if (max(sum(abs(state.m-prevState.m)))==0)
        break;
    end

```

(b) Python

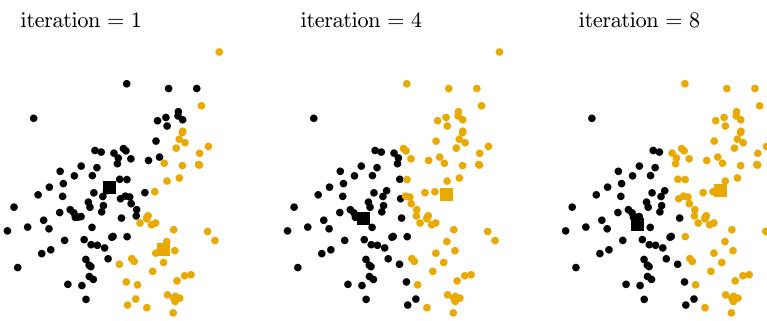
Figure 4.9: Implementation of a kNN classifier.[implement python](#)Fig. Gen.: [plotKmeans.m](#)

Figure 4.10: Three iterations of Kmeans clustering. The colors indicate the cluster assignments and the square indicates the current cluster mean.

space (see Sect. 4.3), find musically “similar” pieces (see Sect. 13), or compute the most likely key in a piece of music (see Sect. 7.5): some form of distance between observations may need to be computed.

A large variety of distance measures exists, and different distance measures are suited to different tasks, data representations, and spaces. At this point, we only want to shortly and informally introduce the most widely-used measures. Typical distance measures between the vectors \mathbf{v}_a and \mathbf{v}_b of dimensionality \mathcal{J} are:

- *Euclidean distance*:

$$d_{\text{EU}}(\mathbf{v}_a, \mathbf{v}_b) = \|\mathbf{v}_a - \mathbf{v}_b\|_2 = \sqrt{\sum_{j=0}^{\mathcal{J}-1} (v_a(j) - v_b(j))^2}. \quad (4.5)$$

It is also referred to as L2 distance.

- *Manhattan distance*:

$$d_M(\mathbf{v}_a, \mathbf{v}_b) = \|\mathbf{v}_a - \mathbf{v}_b\|_1 = \sum_{j=0}^{\mathcal{J}-1} |v_a(j) - v_b(j)|. \quad (4.6)$$

This distance is also known as L1 distance or Rectilinear distance.

- *Cosine similarity*:

$$s_C(\mathbf{v}_a, \mathbf{v}_b) = \frac{\sum_{j=0}^{\mathcal{J}-1} v_a(j) \cdot v_b(j)}{\sqrt{\sum_{j=0}^{\mathcal{J}-1} v_a(j)^2} \cdot \sqrt{\sum_{j=0}^{\mathcal{J}-1} v_b(j)^2}}. \quad (4.7)$$

The Cosine similarity does not easily fit in this list of distances because (i) its range is from $[-1; 1]$ in case of potentially negative inputs, otherwise from $[0; 1]$, (ii) it is not a distance but a similarity measure, but in case of non-negative inputs the *Cosine distance* can be computed as

$$d_C(\mathbf{v}_a, \mathbf{v}_b) = 1 - s_C(\mathbf{v}_a, \mathbf{v}_b), \text{ and} \quad (4.8)$$

(iii) it does not take into account the length of the input vectors, only the angle between them. In case the inputs $\mathbf{v}_a, \mathbf{v}_b$ have zero mean, the Cosine similarity equals the *Pearson Correlation Coefficient* (see Sect. A.3).

- *Kullback-Leibler Divergence*:

$$d_{\text{KL}}(\mathbf{v}_a, \mathbf{v}_b) = \sum_{j=0}^{\mathcal{J}-1} v_a(j) \cdot \log \left(\frac{v_a(j)}{v_b(j)} \right). \quad (4.9)$$

The Kullback-Leibler Divergence is not a great fit in this list either, as it is not symmetric, i.e., $d_{\text{KL}}(\mathbf{v}_a, \mathbf{v}_b) \neq d_{\text{KL}}(\mathbf{v}_b, \mathbf{v}_a)$ and thus not really a distance measure. It is designed to measure how different one probability distribution is to another.

4.5 Exercises

4.5.1 Questions

4.5.2 Assignments

Chapter 5

Data

Machine learning systems are data-driven, meaning that they learn the most likely mapping between input (features) and output (classes) from a set of data. For most supervised machine learning approaches, these data have thus to be annotated with task-dependent *ground truth* labels. For example, if we want to train a music genre classifier, the data annotation have to give us each song's genre. In order to train and test a machine learning system well, there exist important requirements on data.

First, the data have to be representative. That means that, on the one hand, the expected variability of the input data should be well-covered to enable the system to learn. A music genre can, for instance, not be properly represented by only one band as the system might learn to distinguish the band but not the genre. Diversity is also necessary to avoid the system learning non-relevant confounding factors: if, for example, one class of a genre classifier is mostly represented by live recordings then the system might train to identify live recordings instead of genre characteristics. Diversity is, however, not restricted to musical genre: other potential factors might —depending on the task— include audio and production quality and environment, recording years, tempo, musical complexity, playing techniques proficiency of the musicians, instrumentation, and tuning. This list gives only examples and is far from being exhaustive. Careful consideration of possible impacting factors and variables can help minimizing the risk of dataset and algorithm bias.

The class balance (or the label distribution in case of regression tasks) is another representativeness issue that should be carefully checked. It is common that real-world data is imbalanced, e.g., baroque music may not be streamed or sold as often as popular music. Whether this imbalance should be reflected by the dataset or not is a design question that ties in with the choice of system architecture and evaluation metrics — in some cases a uniform class distribution might suit the requirements better. Note that both, an unchanged “real-world” class distribution and a modified class distribution can potentially lead to bias of your system and potentially unwanted results.

Second, the labels or annotation data should not be noisy, meaning the labels should be consistent and unambiguous. This poses a problem for many tasks in ACA as user annotations are frequently subjective so that no absolute ground truth exists.

Third, the training data have to be sufficient. As a rule of thumb: the more complex a task is, the more complex a system needs to be, and the more training data is required. Without a sufficient amount of data, a complex system will not be able to generalize a model from the training data and thus will “overfit,” meaning that the system works very well on the data it has been trained on but poorly on unseen data [DHS00]. The amount of training data becomes a crucial issue for machine learning approaches and systems based on Deep Neural Networks which have shown superior performance at nearly all tasks in audio content analysis but require large amounts of data for training.

These requirements emphasize the importance of careful dataset curation, but also indicate that creating a dataset usually needs considerable effort. This is especially true with respect to annotation data. Although there is a vast amount of music data easily accessible, not all of these data can be directly used for training a machine learning system. This is partly due to copyright restricting the distribution and sharing of music data in the research community, but more importantly due to a lack of annotation data. A system for transcribing drum events from popular music, for instance, needs expert annotations precisely marking each drum hit in

terms of instrument and timing (and possibly the playing technique). Marking these individual hits is, however, a very time-consuming and tedious task so that the increasing requirement for annotation data due to the increasing system complexity usually outpaces the annotation of new data by human annotators. Given the multitude of annotations needed for various content analysis tasks, it is likely that the gap between available annotated data and required amount of training data will widen in the future. This results in a growing need of systems and approaches addressing this challenge, as is reflected by an increasing research interest in this problem from various angles. Current approaches include artificially improving the amount of data, utilizing data from related tasks, reducing the requirements on data annotations, and using semi-, self-, or unsupervised systems. As the field of machine learning evolves rapidly, it is difficult to predict if any of the methods above will become standard approaches. It is clear, however, that the lack of large-scale training data will continue to impact the progress and methods applied to audio content analysis and therefore warrant a closer look at current approaches.

5.1 Data split

It is common to split one big dataset into smaller subsets, the *training data*, *testing data*, and often also the *validation data*. The training data comprises the data that the machine learning model utilizes to learn; this data is known to the system in every stage of the process. The validation subset, particularly used for DNNs with a large number of (hyper-)parameters to adjust, is used for parameter tuning; the parameters of the network trained with the training data are evaluated and adjusted by computing the results for the validation set. Since this is often an empirical process, experiments are often run many times on the validation set. The testing data is “unseen” by the trained system to result in a better estimate of the real-world performance of the system. The familiarity of the system with the different subsets is also reflected in the expected outcome: we expect high system performance when testing with training data, lower system performance for the validation data, and low (normal) performance for testing data.

The three subsets have to be strictly separated as, for example, it could be considered cheating if inputs that have been used for training and are thus known to the system are reused for testing. Otherwise, the evaluation results most likely look better than they are in a real-world scenario. This requirement should also be applied to very similar inputs. For example, even if an audio file is split into multiple snippets, the data split should be applied on the file level, not the snippet level to ensure all snippets originating from the same file end up in either the training or the validation or the testing subset. Similarly, genre classifiers have reported much better results if songs from the same artist or album are in both training and testing subsets [Fle07, FS10].

For each of the subset, the data requirements remain the same, and —although different— the subset statistics and relative properties should remain similar to the overall dataset.

A typical data split between training, validation, and testing is often 70-80 %, 10-15 %, and 10-15 %, respectively.

In case of small datasets, a fixed split into training and testing data might not be feasible as the splits might not be representative for the overall data distribution. A standard way of addressing this challenge, albeit not solving the issue of insufficient dataset size, is *N-fold cross validation*. It is a method to ensure that training set and test set are not identical while maximizing training and testing data variability given in the data. First, the data are split into N equal-sized splits. Then, one split is picked as the temporary testing set and the remaining $N - 1$ splits are combined into the training set. The inference system is trained and evaluated accordingly. The process is repeated N times until each of the N segments has been used as the testing set. The N evaluation results can then be averaged to receive a final result. A typical value for N would be 10, meaning that the data is partitioned into 10 splits. *Leave-one-out cross validation* is an extreme case where N equals the number of data observations. Figure 5.1 shows an example code snippet of Leave-one-out cross validation. Note that high values of N might lead to considerably increased workload or processing time especially in the case of hard to train systems. [add visualization?](#)

```
% loop over observations
for o = 1:size(FeatureMatrix,2)
    % remove current observation from 'training set'
    v_train = [FeatureMatrix(:,1:o-1) FeatureMatrix(:,o+1:end)]';
    C_train = [ClassIndices(1:o-1) ClassIndices(:,o+1:end)]';

    % compute result of Nearest Neighbor Classifier given the traindata
    res = ToolSimpleKnn(FeatureMatrix(:,o)', v_train,C_train,1);

    % if result is correct increment number of true positives
    if (res == ClassIndices(o))
        TP = TP+1;
    end
end

% compute overall (macro) accuracy
Acc = TP/length(ClassIndices);
```

(a) Matlab

```
% loop over observations
for o = 1:size(FeatureMatrix,2)
    % remove current observation from 'training set'
    v_train = [FeatureMatrix(:,1:o-1) FeatureMatrix(:,o+1:end)]';
    C_train = [ClassIndices(1:o-1) ClassIndices(:,o+1:end)]';

    % compute result of Nearest Neighbor Classifier given the traindata
    res = ToolSimpleKnn(FeatureMatrix(:,o)', v_train,C_train,1);

    % if result is correct increment number of true positives
    if (res == ClassIndices(o))
        TP = TP+1;
    end
end

% compute overall (macro) accuracy
Acc = TP/length(ClassIndices);
```

(b) Python

Figure 5.1: Implementation of Leave-One-Out Cross Validation.[implement python](#)

5.2 Artificially Increasing the Amount of Training Data

Given an insufficient amount of annotated data, the machine learning engineer can “cheat” by artificially increasing the amount of training data. This can be done either with synthesized data or by *data augmentation*, generating new training samples through processing existing audio data with task-irrelevant transformations.

The most common way of synthesizing data is rendering MIDI files into audio. A system for musical instrument recognition, for example, might be trained with audio rendered from MIDI data with instrument information. While it is possible to essentially generate any amount of audio data using this method, the resulting audio files are usually clearly different from “real-world” recordings — these differences between training and test data may lead to an underperforming system. In practice, rendered MIDI data is most successfully used either for a pre-training step followed by training with original data or as a small extension of the original training set.

Data augmentation increases the amount of available training data by applying irrelevant transformations to the audio data. A secondary goal of data augmentation is increasing the robustness of the system against irrelevant variables [MP19]. The determination of what data transformations are irrelevant depend on the task. The following forms of data augmentation are most common for ACA systems:

- *Data segmentation*: Segmenting the available data is probably the most established form of data augmentation. A longer audio sample with one label might be split into shorter, possibly overlapping, segments with identical labels [Pee07]. This can work, for example, for music genre classification as it can be assumed that the genre does not change over the time of a song. Obviously, the new segments must have a minimum length that still allows the recognition of a genre. An example where this might not work well is instrument recognition; although it is known that a specific instrument is present in the longer segment, it cannot be guaranteed to be present in all shorter segments.
- *Audio quality degradation*: Many tasks are supposed to be largely independent on the audio quality. The music genre does not change whether you listen to a song over high-quality headphones or your phone speakers. A typical way to augment existing data in terms of audio quality is to add noise to the signal. This can be a simple white or low-pass filtered generated noise [SG15] or recorded real-world noise signals such as street noise or the noise of a concert audience [MHB15]. Other ways of impacting the audio signal quality could be clipping the audio signal or even aliasing [ME13]. Potential audio coding artifacts can be simulated by encoding and decoding at different target bit rates [ME13].
- *Application of audio effects*: A variety of audio effects can be applied without impacting the task too much if used moderately. These include equalization (e.g., low-pass and high-pass filters) and adding reverberation. Modifying the input gain can helpful as long as the input representation is not normalized, otherwise a gain modification has to be time variant through a dynamics compressor. Non-linear effects such as wave-shaping and distortion might be used as well although they usually have considerable impact on the sound quality. Note that some of these effects can also be used to simulate audio degradation.

- *Changing pitch and/or tempo:* Time-stretching and Pitch-shifting can be considered audio effects, however, they are listed here separately because they can be complex algorithms and most openly available algorithms have severe quality issues. If implemented carefully, however, they provide a musically relevant way of transforming the audio signal for augmentation, as subtle pitch or tempo modifications should not impact most tasks. Pitch shifting by small amounts has also been shown to increase robustness of classifiers against adversarial attacks [QL19]. Pitch shifting and time stretching can alternatively be implemented by scaling and interpolating a spectrogram representation.
- *Data combination:* Some datasets for music transcription, instrument classification, or audio source separation include not only the mix of a song but also the individual instrument tracks. In this case, it can be helpful to generate different data by (i) mixing tracks from different songs [SG15] and/or (ii) applying different volume levels before mixing [UPG⁺17] and/or (iii) shifting tracks slightly in time before mixing [UPG⁺17]. Although these data modifications do not make any musical sense, they might provide additional training data that keeps the network from overfitting and thus ultimately improve classification performance.
- *Other forms of data augmentation:* Not all augmentation methods need to be musically relevant to be successful. In the case of an input spectrogram it is possible, for example, to randomly mask specific time-frequency regions so that even the same input might look slightly different each time [PCZ⁺19].

5.3 Utilization of Data from Related Tasks

While there might be a lack of data for a specific target task, they might be easily available for a related base task. *Transfer learning* is a method that tries to transfer the knowledge learned from the related task to the target task. This can be achieved by taking a network trained on the related large dataset, replacing the last few layers (that should be task specific), and retraining it with the target data. Since the network had already been trained for a similar task, fewer data are needed to update it to the target task. If the target dataset is small, retraining the whole network might not be practical because of overfitting. In this case, only the weights of the newly added layers are updated while all the other network weights are frozen.

Two examples given in previous sections can fall in the category transfer learning. Using synthesized data to pre-train a network, for example, clearly transfers the knowledge from the generated audio data to the real-world data. The only difference is that we deal with the same task but different input data, instead of the same data but different tasks. Feature Learning (see Sect. 3.6.14) can be seen as a special case of transfer learning. One example in the music world was presented by Choi et al., who trained a network for an auto-tagging task and then used the embedded representation of the trained network as a feature input to a simpler classifier for music genre classification [CFSC17].

Multi-task Learning can also be seen as a way of transferring knowledge between tasks, however, it is not usually suitable to address small datasets. Multi-task learning attempts to solve two or more tasks at the same time, thus exploiting commonalities between the tasks. This might improve the system's accuracy compared to being trained on a single task alone.

5.4 Reducing Accuracy Requirements for Data Annotation

Another possibility to solve the problem of insufficient data is the investigate and simplify the data annotation. One way of doing so for tasks requiring time annotations is to weakly label them. For example, instead of marking the exact time positions where a specific musical instrument is present in a recording it is possible to simply annotate that *somewhere* within a certain time span (e.g., 10 s), the instrument was active or not. This is referred to as a *weak label* (as opposed to detailed strongly labeled data). This allows the data collection to be simpler and to proceed more rapidly, in turn allowing larger datasets. Traditional machine learning approaches, however, may have to be modified to work successfully with weakly labeled data; more specifically, multiple instance learning approaches are most common for this type of annotation. While in traditional

(“single instance”) supervised systems each instance is assigned a label (e.g., every second), multi-instance learning approaches need only one label for a so-called “bag of instances” (e.g., one label for ten 1 s segments). Multi-instance learning has been successfully used for musical instrument detection [GSL19] and audio event detection [KR16].

5.5 Semi-, Self-, and Unsupervised Learning

If there is a labeled training set available but the dataset size is insufficient, *Semi-Supervised Learning* can be useful by utilizing both the labeled and unlabeled data. The unlabeled data can be used to learn a more complete representation of the data and thus possibly improve the classification results. Semi-supervised learning can take various forms. Self-training iteratively trains the model on the labeled data, predicts the labels of the unseen data, and adds the predicted labeled data to the new training data [XZY05]. We can also choose to add only the most confident predictions to the training data or weigh the added data by confidence. A potential problem with self-training is that early mistakes might reinforce themselves. Generative models benefit from unlabeled data by providing more data points to build the model (such as a Gaussian mixture per class).

The target of self- or unsupervised systems is to train the system without previously annotated training data by automatically estimating the labels or using some form of distance measure based on the data itself as training target. This can, for example, be achieved with an auto-encoder-like architecture which learns a compact internal representation of data by trying to reconstruct it. This internal representation might then be used as input feature for other tasks (see also Feature learning, Sect. 3.6.14). Another possibility is to utilize the outputs of multiple (simple) pre-existing systems as training targets; these pre-existing systems are called the teachers which train the new student system [WL18b, MBCHP18]. Other approaches utilize sound synthesis iteratively to learn from unlabeled data [CC19].

5.6 Exercises

5.6.1 Questions

5.6.2 Assignments

Chapter 6

Evaluation

The evaluation of a designed analysis system is often-neglected in text books, however, is an important part of building a system. After all, what good is a system without knowledge about whether it performs as expected? Typical evaluation-related mistakes obfuscating the real-world performance of a system are (i) using a non-representative test set (too small, not covering all potential input characteristics), (ii) tuning system settings and hyperparameters on the test set, and (iii) using misleading evaluation procedures and metrics.

If the amount of test data is too small or not having the breadth needed to represent the task, it means that the experimental results will most likely neither valid nor be replicable, i.e., results from different data will be inconsistent. If the system parameters have been tuned on the test set, the system most likely does not generalize well and is possibly overfitted, meaning that the system will perform worse on unseen test data. Using misleading evaluation procedures and metrics can have multiple outcomes; it may impact the reproducibility of your evaluation, reduce the comparability of your results, or even lead to meaningless results.

Therefore, a proper system evaluation procedure aims at maximizing validity, reproducibility, and replicability of the results. In order to do so, the evaluation procedure should follow established guidelines, use often-used and publicly available data, allow for comparison with similar studies and systems, and utilize established, fitting metrics.

6.1 Good Practices

First, the evaluation methodology should be unrelated to the implemented algorithm. The evaluation procedure has to be generic enough that it can properly evaluate different algorithms for the same task, the selection of evaluation metrics has to be task-driven, not algorithm-driven, and the evaluation metrics should, if possible, be unrelated to the training metric. There might also be multiple different evaluation metrics that are relevant for the evaluation of a system. For example, in the evaluation of onset detection systems it might not only be of interest how many onsets are predicted correctly (and incorrectly) but also how close they are to the ground truth onset times.

Second, the expected outcome has to be clear: what is the worst case performance and the best case performance? The definition of the *worst case performance* can take different forms. On the one hand, it can be based on simple dataset analysis; for example, the worst case performance of a non-trivial classifier could be modeled by a trivial classifier: simply returning the majority class for each output. This so-called Zero-r-classifier would be correct half the time in the case of two equally distributed classes and more often in the case of imbalanced class distributions. On the other hand, we can construct what is referred to a baseline system (most of the analysis systems detailed in this book would nowadays be considered as baseline systems). Such a baseline system is based on often-used, well established, and mostly simple to implement standard practices. For many audio classification tasks, for example, we might want to use a well-established feature set (e.g., MFCCs) and often-used classifiers such as a k-Nearest Neighbor (NN), GMM, SVM, or Random forest. The *best case performance* is often just the maximum of the evaluation metric, however, that is not always the case. To give an example, let's assume the target is to improve a system for audio classification by providing side information, namely chord labels estimated from the audio signal. In this scenario, the best case would be

if the chord detection would work flawless and we can evaluate the impact on the audio classification by feeding the ground truth labels (this can also be referred to as oracle). Our final system can never improve upon the system based on the oracle.

Third, comparing to the state-of-the-art is a necessity. Practice has shown, however, that simply reporting the numbers from another study as comparison has multiple potential issues. The most obvious problems are the usage of a different dataset or different metrics. In this case, all direct comparisons are basically useless. But there are other potential issues, for example, the dataset might be split into different training and testing splits or the metric might be computed slightly differently. The best approach to comparing to the state-of-the-art is running the published system on your data with your evaluation setup and metrics; unfortunately, this is only possible if the authors publish source code (or a pre-compiled system) that is easy to use. The *Music Information Retrieval Evaluation eXchange (MIREX)*¹ has made laudable progress in providing a default evaluation setting for various MIR tasks.

Fourth, allowing your evaluation to be easily reproducible allows quick verification of results, unproblematic evaluation of different algorithmic iterations, and easy adaptation to different data. It also might enable other researchers to verify and compare results. Reproducibility is not a simple binary property (yes/no) but should be made as easy as reasonably possible. Tasks increasing the reproducibility include (i) publishing the source code of your algorithm and evaluation pipeline, (ii) describing evaluation procedure, data split, and metrics in detail, (iii) ensuring public accessibility of data.

Fifth, tests for the significance of results can help make a stronger argument for an algorithmic improvement. Especially when comparing results that are close together, reporting measures of statistical significance such as the *p*-value (see, e.g., [Ber42]) can increase the confidence in the results.

6.2 Metrics

The use of appropriate metrics is crucial for estimating the performance of a system. Some of the presented metrics can also be used as loss functions for ML systems (e.g., see MSE for computing the best fit for linear regression and for evaluation of regression predictions); in that case, it should be carefully considered whether other evaluation metrics might lead to more meaningful evaluation results (while out of context, Goodhart's law is still a fitting warning: "When a measure becomes a target, it ceases to be a good measure" [Str97]).

6.2.1 Classification

In the case of a binary classifier, there exist two classes (for the remainder of the section, they will be referred to as *Positive* and *Negative*). Possible outcomes are the correct classification of the positive or negative class, which we will refer to as True Positive (TP) or True Negative (TN), respectively, or the incorrect classification, referred to as False Positive (FP) and False Negative (FN), respectively. To summarize, there are

- TPs: Positives correctly identified as Positives,
- TNs: Negatives correctly identified Negatives,
- FPs: Negatives incorrectly identified Positives, and
- FNs: Positives incorrectly identified Negatives.

These four values can be visualized in a *confusion matrix* as shown in Table 6.1. The confusion matrix is an efficient way of understanding the output of a classifier. The confusion matrix may also be normalized to show relative values in percentages: in that case, each row is divided by the number of observations per class. A perfect classifier would have no false predictions, therefore, the number of true predictions on the diagonal will match the number of observations in the dataset and the other matrix entries are zero. In the (normal) case of an imperfect classifier, the number of false classifications per class may indicate either a bias or preference of

¹MIREX Home. <http://www.music-ir.org/mirex>. Last retrieved on Jun. 18, 2021.

	Pred. Pos.	Pred. Neg.	Σ
GT Pos.	True Pos. (TP)	False Neg. (FN)	# of GT Pos. (TP+FN)
GT Neg.	False Pos. (FP)	True Neg. (TN)	# of GT Neg. (FP+FN)
Σ	# of Pred. Pos.	# of Pred. Neg.	# of True Pred. (TP+TN)

Table 6.1: Confusion Matrix. GT refers to the ground truth annotations [improve visuals](#).

the classifier or highlight classes that are easily confused. That is particularly useful in the case of multi-class problems.

The *Accuracy* of a classifier is the number of correct (true) classifications to the overall number of data points:

$$\text{Acc} = \frac{TP + TN}{TP + TN + FP + FN} \quad (6.1)$$

This is the sum of the diagonal of the (unnormalized) confusion matrix divided by the number of data points. Note that this accuracy is not necessarily a good metric in case of imbalanced classes. In that case, it is possible to average over the classes instead of the number of data points so that each class has the same impact on the measure:

$$\text{Acc}_{\text{Macro}} = \frac{\frac{TP}{TP+FN} + \frac{TN}{TN+FP}}{2} = \frac{TPR + TNR}{2} \quad (6.2)$$

The ratios in the numerator are also referred to as True Positive Rate (TPR) and True Negative Rate (TNR) (similarly, we can also compute the False Positive Rate (FPR) and the False Negative Rate (FNR)). This is the mean of the diagonal of the (normalized) confusion matrix. Both the accuracy and the macro accuracy range from 0 to 1 with 1 being the perfect score; they are identical for balanced class distributions.

Another common metric for binary classifiers is the so-called *F-measure*. The F-Measure is the harmonic mean of the two metrics Precision and Recall. The Precision P measures how many of the predicted positives are correct predictions:

$$P = \frac{TP}{TP + FP}. \quad (6.3)$$

The Recall R measures how many of the ground truth positives were correctly predicted:

$$R = \frac{TP}{TP + FN}. \quad (6.4)$$

The F-measure is then

$$F = 2 \cdot \frac{P \cdot R}{P + R}. \quad (6.5)$$

Similar to the accuracy, the F-Measure ranges from 0 to 1 with 1 being the perfect score. Note that depending on the field, the same measures can have different names. We can already see from Eqs. (6.2) and (6.4) that the Recall is also the TPR and may also be referred to as Sensitivity. The TNR is sometimes referred to as Specificity or Selectivity. A FP is a “Type I error” and a FN a “Type II error.” The F-measure is also named F-score or F1 score. This inconsistency in terminology makes discussions on these metrics often confusing and hard to follow, so it is good practice to reiterate and agree on the terminology used.

Computation of the Accuracy and the F-measure require the output of the classifier to be a binary value. Some classifiers, however, output a value between 0 and 1 representing some measure of class likelihood. This output can be thresholded to yield the expected binary output, however, the choice of the threshold significantly impacts the classifier performance. The Area Under Curve (AUC) is a threshold-independent metric in that it evaluates the classifier-performance for many thresholds. In order to do so, it measures the area under the Receiver Operating Curve (ROC) [Faw06], which plots the TPR over the FPR at different classification thresholds. A lower threshold leads to more positives and a higher threshold to fewer positives with the extreme boundary cases of a threshold of 0 leading to all positives and a threshold of 1 leading to no positives. The ROC of a random classifier is therefore a line with a slope of 1 with a resulting AUC of 0.5. The best classifier

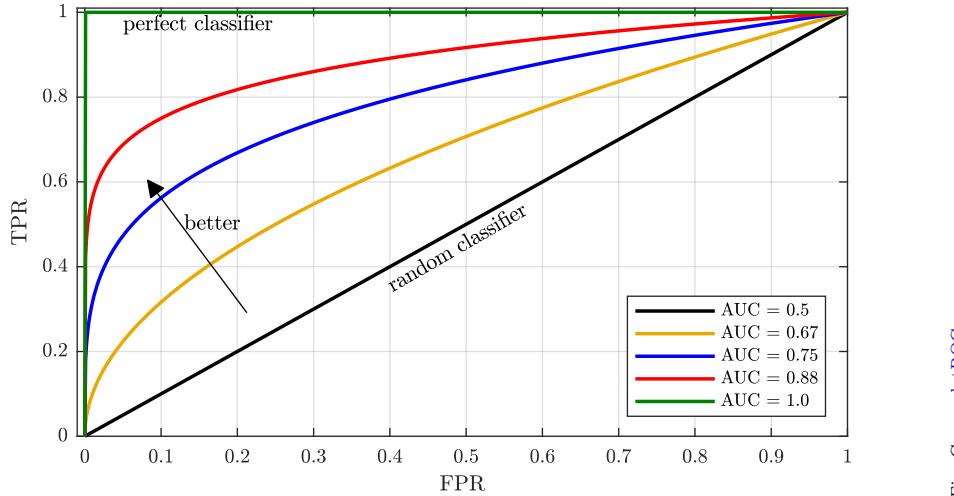


Figure 6.1: Multiple hypothetical ROCs and their corresponding AUCs.

Fig. Gen.: `plotROC.m`

will yield a TPR of 1 regardless of the threshold or number of False Positives; therefore, the AUC will equal 1. Instead of computing the AUC from the ROC (ROC-AUC), the AUC can also be computed from the precision-recall curve (PR-AUC). Figure 6.1 sketches a few prototypical ROCs and lists the resulting AUC values.

6.2.2 Regression

Since a regression algorithm builds a model for predicting a value and not a category like a classifier, different metrics are necessary. More specifically, a prediction cannot be evaluated with a binary match/no-match evaluation, but by measuring some form of deviation from the ground truth. One of the most intuitive metrics for a deviation between ground truth (target) value y and predicted value \hat{y} are the Mean Absolute Error (MAE)

$$MAE = \frac{1}{\mathcal{R}} \sum_{\forall r} |y(r) - \hat{y}(r)|, \quad (6.6)$$

and the MSE

$$MSE = \frac{1}{\mathcal{R}} \sum_{\forall r} (y(r) - \hat{y}(r))^2. \quad (6.7)$$

The range of these metrics depends on the range of the data; a result closer to zero means better prediction. The advantage of the MAE over the MSE is that it gives the error on the scale of the data. The MSE tends to be the more conservative and preferred metric of the two.

The *Coefficient of Determination* R^2 measures the error compared to the error of a baseline prediction simply representing the mean of the ground truth data as a constant output μ_y . The R^2 is computed as

$$R^2 = 1 - \frac{MSE(y - \hat{y})}{MSE(y - \mu_y)}. \quad (6.8)$$

While a perfect prediction leads to an R^2 of 1, it is not bounded by a minimum: very bad predictors can lead to negative R^2 results.

6.2.3 Clustering

If ground truth class labels are available, classification metrics can be applied to the results without any distinction between clustering and classification from an evaluation point of view. The evaluation becomes trickier, however, if no such class labels are available. A good clustering should generally have two properties,

(i) good cluster separation, i.e., maximal distance between different clusters, and (ii) good cluster cohesion, i.e., compact clusters with datapoints belonging to the same cluster data points close together. The *Silhouettes coefficient* is a metric combining these two properties. It is based on the so-called silhouettes [DB79]. Its calculation consists of the following steps:

1. for each data point j , compute the mean distance to all other data points of the same cluster: $d_{\text{intra}}(j)$
2. for each data point j , compute the mean distance to the neighboring cluster by finding the minimum of the mean distance to all data points per cluster: $d_{\text{neighbor}}(j)$
3. for each data point j , compute the *silhouette* s with

$$s(j) = \frac{d_{\text{neighbor}}(j) - d_{\text{intra}}(j)}{\max(d_{\text{neighbor}}(j), d_{\text{intra}}(j))} \quad (6.9)$$

Each silhouette has a range of $[-1; 1]$

4. the *Silhouette coefficient* is then the maximum of the mean silhouettes per cluster [KR05].

A different metric is named after its authors Davies and Bouldin [DB79]. In its simplest form, it is based on the following steps

1. compute the standard deviation $\sigma(k)$ for each cluster k
2. compute the distances $d_\mu(k_1, k_2)$ between all cluster centroids
3. for each pair of clusters, compute

$$r(k_1, k_2) = \frac{\sigma(k_1) + \sigma(k_2)}{d_\mu(k_1, k_2)} \quad (6.10)$$

and for each pair, pick the cluster with the maximum pairwise distance $r_{\max}(k_1)$

4. finally, the *Davies-Bouldin-Index* is average over all maximum distances

6.3 Exercises

[write me](#)

6.3.1 Questions

6.3.2 Assignments

Part II

Music Transcription

Chapter 7

Tonal Analysis

Tonal aspects play an important role in understanding and analyzing music, as can be seen from the vast number of pitch-related publications in music theory. Pitches are the basic building blocks of key, melody, and harmony in music.

7.1 Human Perception of Pitch

The human perception of *pitch* is directly related to the frequency of a signal in the way that higher frequencies will lead to the perception of a higher pitch. If the signal is a combination of sinusoidal components with the frequencies $f_0, 2f_0, 3f_0, \dots$ (which is a reasonable approximation for pitched, quasi-periodic sounds produced by many musical instruments), then the *fundamental frequency* f_0 , the frequency of the longest period, generally dominates the perceived pitch [Che10]. As a matter of fact, humans will usually even perceive the same pitch for this combination of harmonics if the fundamental frequency f_0 has low power or is missing [Fle24, Sch38]. Figure 7.1 shows how a harmonics of a periodic signal might look like. The pitch we hear is usually related to the first harmonic, the fundamental frequency.

7.1.1 Pitch Scales

The relation between the fundamental frequency and the perceived pitch is non-linear; at higher frequencies, two pitches with the same perceived pitch distance will have a larger frequency distance than at lower frequencies. Simply put, the non-linearity is tied to the frequency resolution of the human cochlea. There are different approaches to measure this cochlear frequency map and thus there exist different proposals modeling it. The most common models are the *mel scale* and the *critical band rate*, also called *bark scale* (Fig. 7.2). There has been a number of proposals for analytical functions to approximate the measurement data resulting from listening tests for these scales. In most practical audio analysis applications, the use of one or another approximation

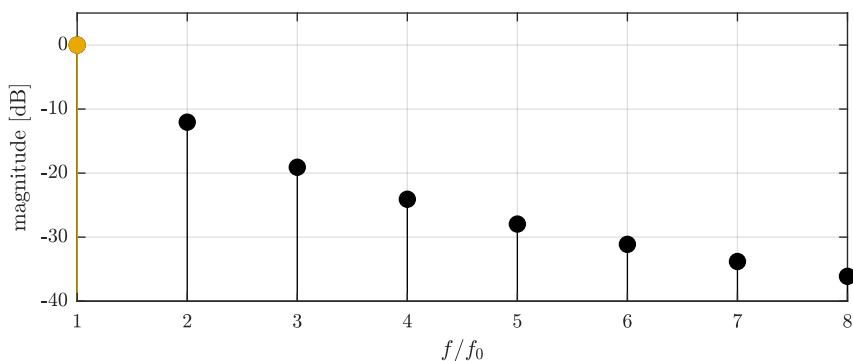


Fig. Gen.: `plotHarmonics.m`

Figure 7.1: Potential harmonics as integer multiples of the fundamental frequency.

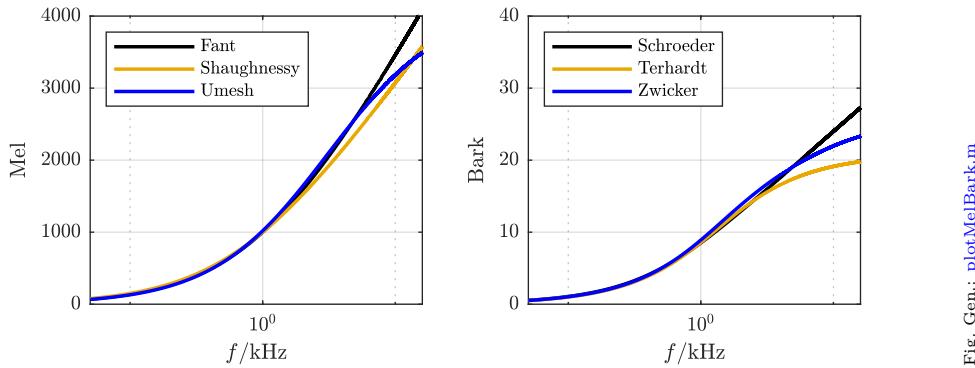


Figure 7.2: Different models for the non-linear mapping of frequency to mel (left) and bark (right).

```
% Fant
function [mel] = acaFant(f)
    mel      = 1000 * log2(1 + f/1000);
end

% Shaughnessy
function [mel] = acaShaughnessy(f)
    mel      = 2595 * log10(1 + f/700);
end

% Umesh
function [mel] = acaUmesh(f)
    mel      = f./(2.4e-4*f + 0.741);
end
```

(a) Matlab

```
# Fant
def acaFant_scalar(f):
    return 1000 * math.log2(1 + f/1000)

# Shaughnessy
def acaShaughnessy_scalar(f):
    return 2595 * math.log10(1 + f/700)

# Umesh
def acaUmesh_scalar(f):
    return f/(2.4e-4*f + 0.741)
```

(b) Python

Figure 7.3: Implementation of models for frequency to Mel conversion.

appears to be circumstantial (compare [SP03, BS05]). Nevertheless, several of these models will be presented below to illustrate the number of options.

The term *mel* was introduced in 1937 by Stevens et al. as the name of a subjective pitch unit [SVN37]. The *mel scale* is a measure of tone height. The empirical data used to build the numerous analytical models of the mel scale stems from only a limited number of psychological experiments: the most important test results have been presented by Stevens et al. [SVN37], Stevens and Volkmann [SV40], and Siegel [Sie65]. Three models will be presented here; the two older models by Fant [Fan73]

$$m_F(f) = 1000 \cdot \log_2 \left(1 + \frac{f}{1000 \text{ Hz}} \right) \quad (7.1)$$

and O'Shaughnessy [O'S87]¹

$$m_S(f) = 2595 \cdot \log_{10} \left(1 + \frac{f}{700 \text{ Hz}} \right) \quad (7.2)$$

are most commonly used.² Figure 7.3 shows the implementation of the models listed above.

There exist multiple other proposed models; for example, the model proposed by Umesh et al. [UCN99] is less widely known and is referenced here mainly to demonstrate the variety of models:

$$m_U(f) = \frac{f}{2.4 \cdot 10^{-4}f + 0.741}. \quad (7.3)$$

The *bark scale* or *critical band rate* is constructed from the bandwidth of measured frequency groups, the critical bands. Zwicker and Fastl suggest that the bark scale is related to the mel scale by 1 bark = 100 mel [ZF99].

¹appears earlier in [MC76]

²Note that O'Shaughnessy's model is sometimes also referenced in the form

$$m_S(f) = 1127 \cdot \log \left(1 + \frac{f}{700 \text{ Hz}} \right).$$

The most prominent models of the bark scale have been proposed by Schroeder et al. [SAH79]

$$\mathfrak{z}_S(f) = 7 \cdot \operatorname{arcsinh} \left(\frac{f}{650 \text{ Hz}} \right), \quad (7.4)$$

Terhardt [Ter79]

$$\mathfrak{z}_T(f) = 13.3 \cdot \operatorname{arctan} \left(0.75 \cdot \frac{f}{1000 \text{ Hz}} \right), \quad (7.5)$$

and Zwicker and Terhardt [Zwi80]

$$\mathfrak{z}_Z(f) = 13 \cdot \operatorname{arctan} \left(0.76 \cdot \frac{f}{1000 \text{ Hz}} \right) + 3.5 \cdot \operatorname{arctan} \left(\frac{f}{7500 \text{ Hz}} \right). \quad (7.6)$$

An example for a model with lower computational complexity is Traunmüller's model [Tra90]

$$\mathfrak{z}_{TM}(f) = \frac{26.81}{1 + 1960/f} - 0.53. \quad (7.7)$$

Many more models have been proposed over the years for the non-linear transformation of frequency to perceptual frequency groups, pitch height, and position on the human *cochlea*.

Moore's model for the *Equivalent Rectangular Bandwidth (ERB)* can be seen as a model alternative to the critical band rate [MG83]

$$\mathfrak{e}(f) = 9.26 \log \left(1 + \frac{f}{228.7} \right). \quad (7.8)$$

Terhardt introduced a function, which he named *SPINC*, as an alternative to the mel scale [Ter92]:

$$\mathfrak{s}(f) = 1414 \operatorname{arctan} \left(\frac{f}{1414 \text{ Hz}} \right), \quad (7.9)$$

and Greenwood proposed the following equation to compute the position (normed to the range of [0; 1]) of a specific frequency on the cochlea [Gre61]:

$$\mathfrak{r}(f) = \frac{1}{2.1} \log_{10} \left(\frac{f}{165.4} + 1 \right). \quad (7.10)$$

7.1.2 Chroma Perception

There is an additional facet to human pitch perception: not only do we perceive pitch height from low to high but we tend to group pitches with specific frequency ratios [She64, DDH08]. More specifically, humans perceive frequencies with a frequency ratio of a power of 2 (such as $f_0, 2f_0, 4f_0, 8f_0, \dots$) as very similar and closely related to each other. This phenomenon is usually called *chroma perception*.

Figure 7.4 visualizes this in a helix plot. On the one hand, the frequency is monotonically increasing on the z axis, modeling the tone or pitch height. On the other hand, points with the same (x, y) coordinates share a frequency ratio of a power of 2 — they share the same scale degree and are in the same pitch class (see Sect. 7.2.1), respectively. The circle which appears when looking directly on the (x, y) plane would thus encompass all pitch classes.

7.2 Representation of Pitch in Music

Much can be said about pitch-related properties of music, covering not only the frequency mapping of specific pitches but also interaction of pitches in chords and melodies, harmony progression, and musical key. It is not the intention of this section to give a comprehensive overview on the (music) theory involved; instead, some basics will be covered in a simplified manner since the understanding of some theoretical background can be of help in the successful design of analysis algorithms.

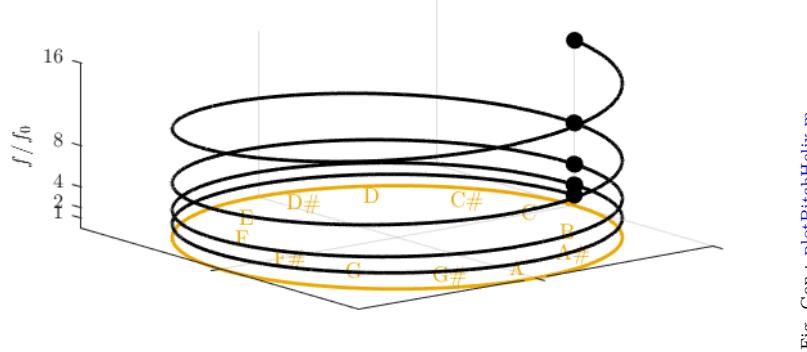


Figure 7.4: Helix visualizing the two facets of pitch perception: pitch height and chroma.

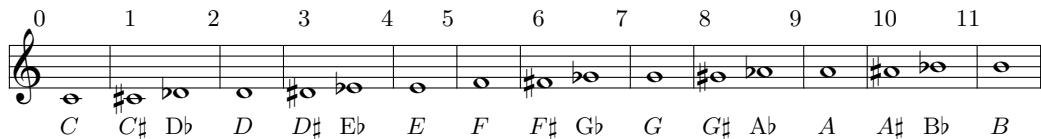


Figure 7.5: Chromatic pitches $C4 \dots B4$ in musical score notation with pitch class indices and pitch class names; enharmonically equivalent pitches are displayed twice

7.2.1 Pitch Classes and Names

The concept of musical pitch in western music theory closely follows the pitch helix (see Fig. 7.4) in that each *octave*, i.e., each range with boundaries with a frequency ratio of $2 : 1$, is divided into the same chunks: the 12 *pitch classes*.

The common labels of these octave-independent pitch classes are shown in Fig. 7.5. Seven of those pitch classes form the so-called *diatonic scale*. Table 7.1 shows such a diatonic scale, more specifically the major mode (see Sect. 7.5) based on a root note C with pitch class index, pitch class name, Solfège name and distance to the lower (scale-inherent) pitch class in semi-tones. As can be seen from the table, the distance between two neighboring pitches is in most cases two semi-tones; twice, however, it is only one semi-tone. Any pitch between the presented diatonic pitches can be constructed by raising the pitch (e.g., $C \rightarrow C\sharp$) or lowering it (e.g., $E \rightarrow E\flat$). For now it will be assumed that, e.g., $C\sharp$ equals $D\flat$ (a relation that is referred to as *enharmonic equivalence*), resulting in 12 pitch classes per octave as shown in Fig. 7.5. The 12 pitch classes can also be visualized on a piano keyboard as shown in Fig. 7.6. The white keys form the C Major mode shown in Table 7.1.

Figure 7.5 displays the pitch classes in one octave in musical score notation. Note that score notation

Table 7.1: Names and distance in semi-tones ΔST of diatonic pitch classes

Index	Name	Solfège Name	ΔST
0	C	Do	1
2	D	Re	2
4	E	Mi	2
5	F	Fa	1
7	G	Sol	2
9	A	La	2
11	B	Si	2

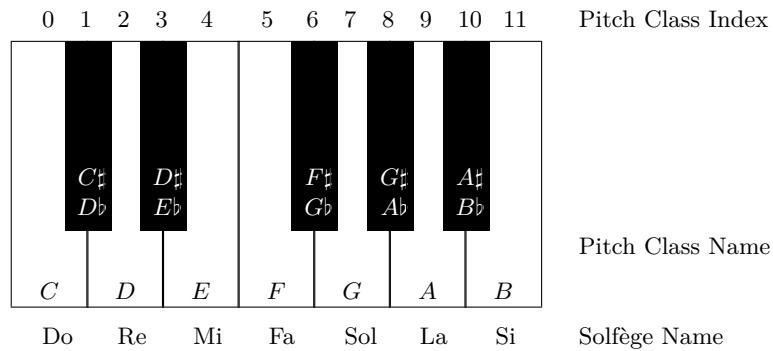


Figure 7.6: One octave on a piano keyboard with annotated pitch class names

Table 7.2: Names of musical intervals, their enharmonic equivalents, and their pitch distance in semi-tones

Interval	Enharmonic Equivalent	ΔST
Unison	Diminished Second	0
Minor Second	Augmented Unison	1
(Major) Second	Diminished Third	2
Minor Third	Augmented Second	3
Major Third	Diminished Fourth	4
(Perfect) Fourth	Augmented Third	5
Augmented Fourth	Diminished Fifth/Tritone	6
(Perfect) Fifth	Diminished Sixth	7
Minor Sixth	Augmented Fifth	8
Major Sixth	Diminished Seventh	9
Minor Seventh	Augmented Sixth	10
Major Seventh	Diminished Octave	11
(Perfect) Octave	Augmented Seventh	12

includes octave information while the pitch class indices do not.

A common convention for naming musical pitches used in the following is simply the pitch class name followed by an octave index, e.g., *C*2 or *A*4. Each new octave starts with a *C* by convention.

7.2.2 Intervals

The distance between two pitches is the musical *interval*. It is used for both the distance of simultaneously sounding pitches and pitches sounding one after another. It is a relative metric without information on absolute pitch height: humans will hear the same interval for different pairs of pitches if the ratio between their fundamental frequencies is the same. Table 7.2 names commonly used intervals and their corresponding distance in semi-tones.

Figure 7.7 displays the most important intervals (rising from pitch *C*4) in musical score notation.

7.2.3 The Frequency of Musical Pitch

The most systematic model for relating musical pitch to frequency and vice versa is using the so-called equal temperament which also results in enharmonic equivalence (other temperaments will be mentioned in Sect. 7.2.3.1). The best example for such a transformation is the MIDI scale in which each semi-tone has a distance of 1 to its nearest neighbor. The equations for the transformation from frequency f to MIDI pitch p

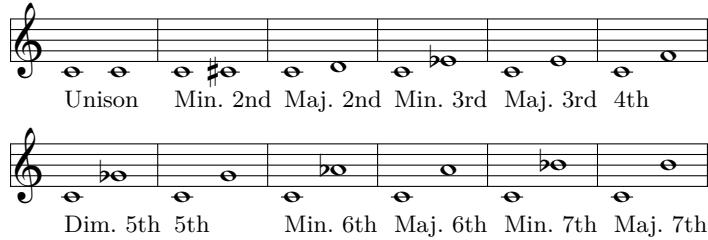


Figure 7.7: Musical intervals in musical score notation

and vice versa are

$$\mathfrak{p}(f) = 69 + 12 \cdot \log_2 \left(\frac{f}{f_{A4}} \right), \quad (7.11)$$

$$f(\mathfrak{p}) = f_{A4} \cdot 2^{\frac{\mathfrak{p}-69}{12}}. \quad (7.12)$$

The reference frequency f_{A4} is the tuning frequency (see Sect. 7.4); using 12 times the logarithm to the base 2 ensures that each octave is divided into 12 parts of equal “width” and the constant 69 results in the pitch $A4$ having the index 69, a convention of the MIDI standard [MID01] to ensure that index 0 maps to the pitch class C and the lowest key on a piano keyboard is covered. Thus, the MIDI pitch can be mapped easily to the pitch class index PC as introduced above by using a modulo operation:

$$PC(\mathfrak{p}) = \mod(\mathfrak{p}, 12). \quad (7.13)$$

The unit cent $\Delta C(f_1, f_2)$ is a distance measure between two pitches or frequencies f_1 and f_2 . It can be computed by

$$\begin{aligned} \Delta C(f_1, f_2) &= 100 \cdot (\mathfrak{p}(f_1) - \mathfrak{p}(f_2)) \\ &= 100 \cdot \left(\left(69 + 12 \cdot \log_2 \left(\frac{f_1}{f_{A4}} \right) \right) - \left(69 + 12 \cdot \log_2 \left(\frac{f_2}{f_{A4}} \right) \right) \right) \\ &= 1200 \cdot \log_2 \left(\frac{f_1}{f_2} \right). \end{aligned} \quad (7.14)$$

A semi-tone interval has thus a distance of 100 cents and an octave has a distance of 1200 cents.

7.2.3.1 Temperament

The temperament defines a system of frequency ratios for intervals.

In the equally tempered case assumed above the frequency ratio between the mid-frequencies of two pitches f_1 and f_2 spaced by N semi-tones (with N being a negative or positive integer) is always

$$\frac{f_1}{f_2} = 2^{N/12}. \quad (7.15)$$

Therefore, the distance between two pitches is constant and independent of tonal and harmonic context; it stays also constant if the enharmonic equivalents of the two pitches are used: the frequency ratio for the interval $B, F\sharp$ is identical to the ratio of the intervals $B, G\flat$ or $C\flat, G\flat$. This, however, is only true for the equal temperament.

The Pythagorean, meantone, and diatonic temperaments are examples of other temperaments in which the pitches are tuned depending on the key. Basically, the Pythagorean temperament is constructed from perfect fifths (frequency ratio 3 : 2), the meantone temperament is constructed with nearly perfect thirds, and the diatonic temperament intends to use as small frequency ratios as possible toward the tonic for every scale degree. Such temperaments are, therefore, dependent on the root note or tonic they are constructed from and an instrument tuned for one root note can easily sound out of tune if played in other keys.

Table 7.3 shows the deviations in cents of different temperaments from the equal temperament.

Table 7.3: Deviations of the Pythagorean, meantone, and two diatonic temperaments from the equally tempered scale in cents at a reference pitch of C (compare Briner [Bri98]) visualize this in a polar plot???

Pitch Class	Equally	Pythagorean	Meantone	Diatonic Major	Diatonic Minor
C	0	0	0	0	0
$C^{\#}$	0	—	—	—	—
D	0	+3.9	-6.9	+3.9	+3.9
E^b	0	—	—	—	+15.6
E	0	+7.8	-13.7	-13.7	—
F	0	-2.0	+3.4	-2.0	-2.0
$F^{\#}$	0	—	—	—	—
G	0	+2.0	-3.5	+2.0	+2.0
A^b	0	—	—	—	+13.7
A	0	+5.9	-10.2	-15.6	—
B^b	0	—	—	—	+17.6
B	0	+9.8	-17.1	-11.7	—

7.2.3.2 Intonation

In contrast to temperament, the frequency of a certain pitch may vary over time depending on the musical context. This deviation from the frequency grid set by the temperament is called (expressive) intonation and is part of the musical performance.

Obviously, only musicians who are not forced to a pre-defined temperament can use expressive intonation. Examples are vocalists as well as players of string, brass, or woodwind instruments, while, for example, piano players have no means of changing the pitch frequency during a performance. It is generally assumed that musicians tend to produce pure frequency relationships such as $f_1/f_2 = 3/2$ for a fifth (seven semi-tones) because it sounds more “natural” [Set05]. Furthermore, if a specific note *leads* musically to the following, the frequency will in many cases be adjusted toward the following pitch. A good example are leading tones where the seventh scale degree leads to the first scale degree (in *C Major*: $B \rightarrow C$).

A special performance phenomenon related to expressive intonation is *vibrato*. Vibrato, a musical ornament, is a periodic frequency modulation of the pitch around its mean frequency. The extent and frequency of a vibrato is somewhat instrument dependent; typical frequencies are in the range of 5–9 Hz and the amplitude may be as large as two semi-tones [Sea38, Sch40, Fle75].

7.3 Fundamental Frequency Detection

Fundamental Frequency Detection in the music domain is also frequently referred to as *pitch detection* or *pitch tracking*. Formally, the difference is that for the former we detect the frequency in Hz while for the latter the musical pitch is detected. Practically, this only requires a simple conversion according to Eq. (7.11) and these terms are in fact used synonymously.

The basic assumption for all approaches to the estimation of the fundamental frequency is that the signal is periodic or quasi-periodic. The periodic state of an acoustic tone can be represented in a Fourier series as introduced in Sect. 3.1.1, meaning that it is a superposition of weighted sinusoids. The frequency of these sinusoids is an integer multiple of the lowest —the *fundamental*— frequency. The different frequency components of a tone are called *harmonics* or *partials*, with the first harmonic being the *fundamental frequency*. Higher harmonics are also called *overtones*. The first six harmonics of the musical pitch *A3* are displayed in Fig. 7.8 in traditional musical score notation.

This frequency structure can be found for most signals generated by acoustic or electronic instruments which are perceived as pitched sounds. However, there are certain deviations from this rule. For example, humans will

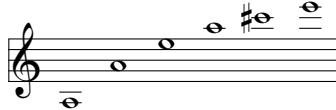


Figure 7.8: Six harmonics of the fundamental pitch *A*3 in musical score notation

hear the fundamental frequency of a harmonic series even without this frequency actually being present in the signal [Fle24, Sch38]. Although an absent fundamental frequency occurs only rarely, it can easily occur that the first harmonic has lower energy than one or more of the higher harmonics.

The piano and string instruments are examples of acoustic signals in which the harmonics are not placed precisely at integer frequency multiples of the fundamental frequency but slightly off. A model of this deviation or inharmonicity is

$$f_k = k f_0 \sqrt{1 + \lambda(k^2 - 1)} \quad (7.16)$$

with k being the index of the harmonic (in this case better called partial as it is not entirely harmonic) and λ being the inharmonicity factor with typical values in the range $[10^{-3}; 10^{-4}]$ [FR98]. There are also instruments which are perceived as being pitched yet show no clean harmonic pattern. Examples for this class of instruments are the xylophone, the vibraphone, and timpani. Nevertheless, the assumption of quasi-periodic states of a music signal has in many cases been proven to be valid and successful for *fundamental frequency detection*.

The common range of fundamental frequencies for musical instruments roughly starts between 20 and 50 Hz (e.g., on the double bass) and ends between 3–5 kHz (e.g., on the piccolo flute). Depending on the instrument, a number of at least three to seven harmonics should be considered as important main components of the sound (if components other than the fundamental frequency are of interest).

7.3.1 Detection Accuracy

Algorithms for fundamental frequency detection work either in the time domain by estimating the period length of the fundamental or in the frequency domain by finding the frequency of the fundamental. For digital signals, both are discrete value domains and have thus a maximum accuracy determined by the distance of two time domain samples and two frequency domain bins, respectively. There are work-arounds for virtually enhancing the resolution such as frequency reassignment (see Sect. B.6), but here we will look at the accuracy of fundamental frequency detection as it results from working on discrete signals.

7.3.1.1 Time Domain

As already mentioned above, the estimated period length of the fundamental frequency is quantized to samples in the time domain, resulting in the estimated period length being a multiple of the distance between two samples

$$T_Q = j \cdot T_S \quad (7.17)$$

with the integer multiplier j . This quantization leads to an error; the error amount depends on both the sample rate and the period length. Figure 7.9 shows the minimum detection error in cent in dependency of the fundamental frequency for two different sample rates. The absolute worst-case error is small for low frequencies and increases with the frequency as the relative impact of the quantization will be more severe for short period lengths. Higher sample rates will result in smaller errors. The sawtooth-like variation comes from the fact that some period lengths will exactly be a multiple of the distance of two neighboring samples T_S and thus have no error, while others have larger errors. Note that while a detection error of a few cents might be easily tolerable, a deviation of a quarter-tone or more at higher frequencies could be problematic for musical pitch detection systems.

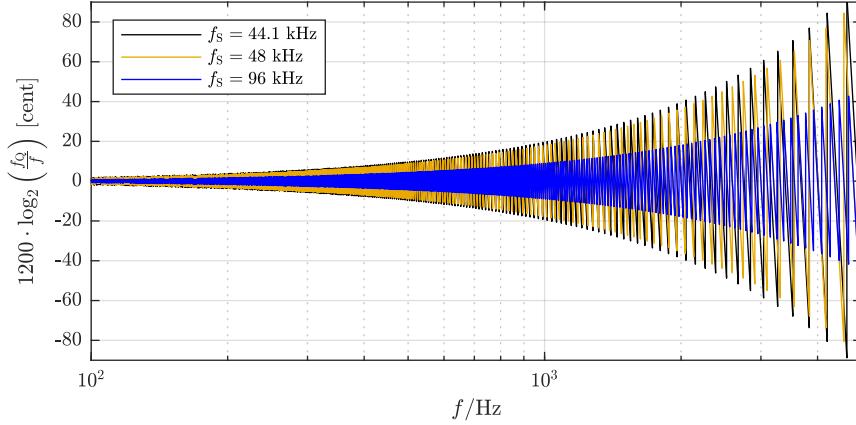


Fig. Gen.: plotPitchErrorTimeDomain.m

Figure 7.9: Detection error in cents resulting from quantization of the period length to a length in samples for two different sample rates.

Table 7.4: Frequency resolution of the STFT for different block lengths at a sample rate of 48 kHz with bin index and frequency of the first bin k_{ST} with a distance to the following bin smaller than half a semi-tone

\mathcal{K}	Δf [Hz]	k_{ST}	$f(k_{\text{ST}})$ [Hz]
256	187.5	35	6562.5
512	93.75	35	3281.25
1024	46.875	35	1640.625
2048	23.4375	35	820.3125
4096	11.7188	35	410.1563
8192	5.8594	35	205.0781
16384	2.9297	35	102.5391

7.3.1.2 Frequency Domain

The trade-off between time resolution and frequency resolution is one of the big issues in STFT-based frequency analysis. While long analysis blocks increase the frequency resolution, they require a periodic and stationary signal during the analysis block in order to be useful (and obviously lead to a lower time resolution). As can be easily seen from Eq. (3.36), the frequency resolution is constant as long as the ratio of sample rate f_S and block length \mathcal{K} stays constant:

$$f_Q = k \cdot \frac{f_S}{\mathcal{K}} \quad (7.18)$$

Table 7.4 displays the frequency resolution for different STFT sizes.

Figure 7.10 visualizes the error in cents for an analysis block length of 2048 samples at the sample rates 44.1 and 96 kHz. As the error is measured in cents (logarithmic scale) it decreases with increasing frequency (linear scale). The graph clearly highlights the problems in accuracy for low-frequency signals. A frequency domain algorithm with a bin resolution accuracy will have considerable inaccuracies that may span multiple semi-tones.

7.3.1.3 Potential Solutions

There exist several workarounds to deal with these resolution issues. On the one hand, it is possible to combine time and frequency domain methods to try to get the best of both worlds [REF]. On the other hand, we can use various forms of interpolation to virtually increase the resolution. As the time domain error decreases with increasing sample rate, one simple possibility is to convert the sample rate of the signal into a higher sample rate (upsampling). Done with proper post-filtering, the resulting signal may allow for a better peak detection

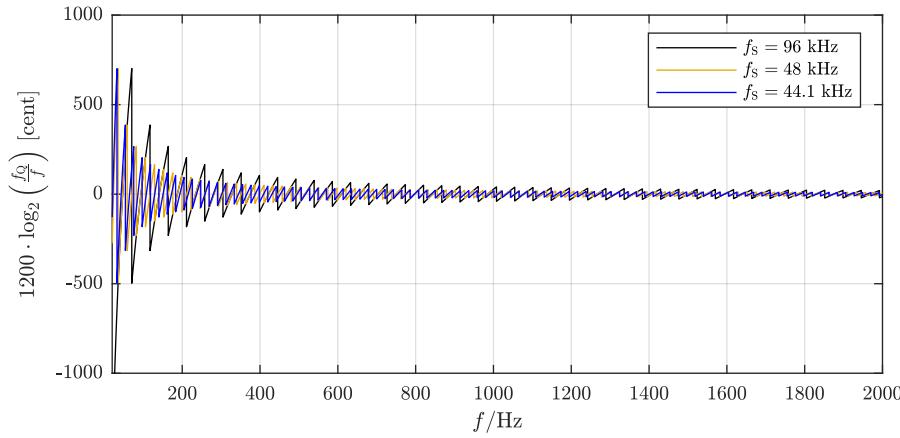


Fig. Gen.: plotPitchErrorFreqDomain.m

Figure 7.10: Detection error in cents resulting from quantization of the fundamental frequency to the STFT bin at an analysis block length of 2048 samples for two different sample rates.

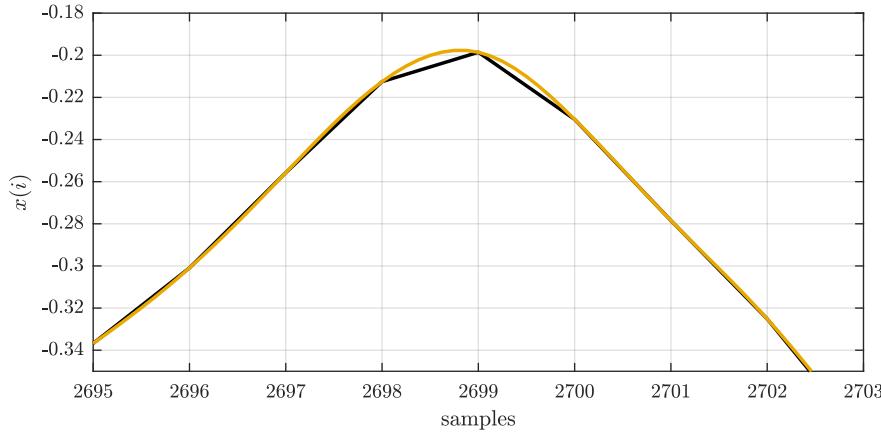


Fig. Gen.: plotTimeInterp.m

Figure 7.11: Very short excerpt of an original and a signal interpolated for peak detection.

as shown in Fig. 7.11. However, if we apply upsampling to, e.g., the input of a correlation-based pitch tracking approach, it comes at the cost of a considerable workload increase. It might also be an option to apply this upsampling step to the correlation result (or only part of it) to get a (possibly less trust-worthy) approximation of a high-resolution result.

In the frequency domain, there are two standard ways of virtually increasing the frequency resolution: (i) interpolation of the spectrum similar to the upsampling above, and (ii) applying *zero-padding* to the windowed block before computing the STFT. As the frequency resolution directly depends on the STFT block length, concatenating zeros to the windowed signal increases the frequency resolution while keeping the spectral content similar (the length difference between window length and block length will reshape the spectral leakage, however). Figure 7.12 shows both a zeropadded magnitude spectrum and a spline interpolated magnitude spectrum in comparison with the original magnitude spectrum.

In the frequency domain, there exist a better way of analyzing frequency components in high resolution if the frequency representation is mostly sparse and not too noisy: computing the *instantaneous frequency*. As explained in more detail in Sect. B.6, this method uses the phase of the spectrum to estimate the “exact” frequency of a spectral component. If applied to a spectral bin that has been identified as a candidate for the fundamental frequency, extracting the instantaneous frequency is more likely to give a more exact frequency estimate than using the bin frequency itself.

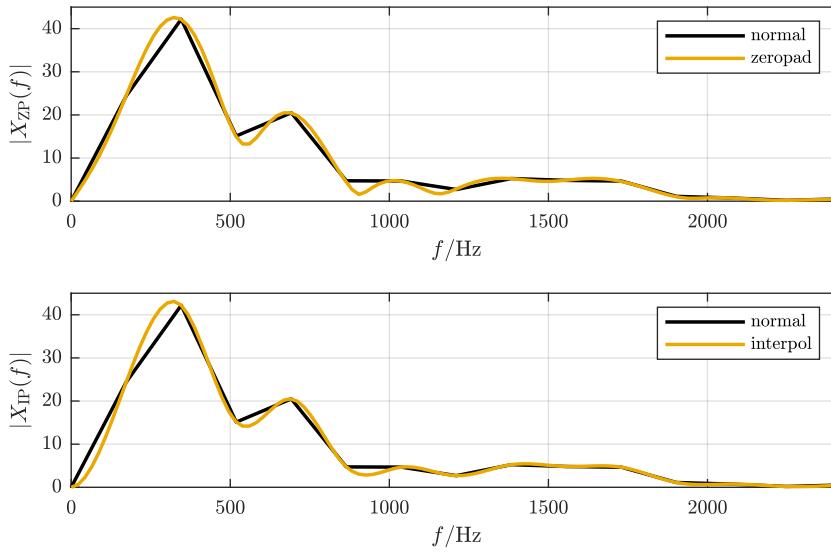
Fig. Gen.: [plotFreqInterp.m](#)

Figure 7.12: Magnitude spectrum and zeropadded magnitude spectrum (top) and magnitude spectrum and interpolated magnitude spectrum (bottom).

7.3.2 Pre-Processing

Depending on the algorithmic approach, a system for fundamental frequency detection might be improved by applying appropriate pre-processing steps such as down-mixing, filtering, and sample rate conversion. It might also be helpful to remove noisy and non-tonal components before the actual detection.

Since the range of fundamental frequencies is quite restricted compared to (half) the sample rates used nowadays in audio signal processing and timbre differences should be irrelevant for fundamental frequency detection systems, higher frequency components are frequently removed by both low-pass filtering and/or down-sampling the input signal. The cut-off frequency of the low-pass filter depends on the fundamental frequency range of the input signal as well as on the usefulness of higher harmonics for the subsequent detection algorithm. The target frequency of the resampling process might also depend on the required resolution for the period length estimation (compare Sect. 7.3.1).

In addition, a high-pass filter removing components near DC may be helpful to clean up the input signal — typical cut-off frequencies lie in the range of 30–300 Hz.

7.3.3 Monophonic Input Signals

A monophonic signal is — as opposed to a polyphonic³ signal — single-voiced. This means there is one fundamental frequency present at a time. The problem of detecting the fundamental frequency in monophonic signals can be solved by detecting the longest periodicity period as the frequencies of the harmonics will be integer multiples of the fundamental frequency. Since many algorithms for monophonic input signals make heavy use of this property, they are of no or only limited use in the context of polyphonic input signals which are mixtures of multiple voices with possibly different and time-variant timbre.

7.3.3.1 Zero Crossing Rate

The *zero crossing rate* has already been introduced in Sect. 3.6.13. There are two ways to estimate the fundamental period length with zero crossings; the first is to relate the number of zero crossings in an analysis

³The term *polyphonic* will be used for signals with multiple voices and possibly multiple instruments. A more restrictive definition used in musicology relates polyphony to quasi-independent voices as opposed to homophony, where the multiple voices move together in harmony.

```

i_start      = (n-1)*iHopLength + 1;
i_stop       = min(length(x),i_start + iBlockLength - 1);
i_tmp        = diff(find(x(i_start:i_stop-1).*x(i_start+1:i_stop) < 0));
% average distance of zero crossings indicates half period
f(n)         = 2*mean(i_tmp); % or histogram max, ...

```

(a) Matlab

```

# get current block
if not x[np.arange(i_start, i_stop + 1)].sum():
    continue
else:
    x_tmp = x[np.arange(i_start, i_stop + 1)]
# compute zero crossing indices
x_tmp = x_tmp[np.arange(0, iBlockLength - 1)] * x_tmp[np.arange(i_start, i_stop + 1)]
i_tmp = np.diff(np.argwhere(x_tmp < 0), axis=0)
# average distance of zero crossings indicates half period
if i_tmp.size:
    f[n] = f_s / np.mean(2 * i_tmp)

```

(b) Python

Figure 7.13: Estimation of the fundamental period length of one block of audio with zero crossings.

block to its length, an approach that will only produce acceptable results for very large block sizes:

$$T_0(n) = \frac{2 \cdot (i_e(n) - i_s(n))}{f_s \cdot \sum_{i=i_s(n)}^{i_e(n)} |\text{sign}[x(i)] - \text{sign}[x(i-1)]|}, \quad (7.19)$$

and the second is to measure the interval $\Delta t_{\text{ZC}}(j)$ between neighboring zero crossings. This interval relates directly to the fundamental period length. If Z zero crossings have been detected in the analysis block, the fundamental period length can be estimated with

$$T_0(n) = \frac{2}{Z-1} \sum_{z=0}^{Z-2} \Delta t_{\text{ZC}}(z). \quad (7.20)$$

In case of large block lengths, the time intervals can also be sorted into a histogram. This can make the estimate more robust as spurious outliers do not influence the result. The estimated period is then the location of the histogram maximum multiplied by a factor of 2.

Figure 7.13 shows the simplicity of estimating the fundamental frequency with zero crossings. The computational efficiency does, however, come at the cost of unreliable results, especially for signals with a large number of harmonics and/or noise. This problem can be somewhat addressed by applying a low-pass filter to the input signal. It is also possible to investigate the distance not only between zero crossings but between local extrema such as the maximum and minimum. This can increase robustness as explained by Rabiner and Schafer [RS78].

7.3.3.2 Autocorrelation Function

The normalized ACF (compare Sects. A.3 and 3.6.12) is probably the most widely used approach to fundamental period length estimation. As the ACF indicates the self-similarity (i.e., the periodicity) of the signal, the lag of the maximum value is a direct estimation of the fundamental period length. Certain restrictions to maximum detection as exemplified in Sect. 3.6.12 can be applied to increase the algorithm's reliability. Figure 7.14 shows the implementation of an ACF-based pitch extraction with some additional constraints. Figure 7.15 shows the result of this ACF-based fundamental frequency estimation for an example audio block.

Low-pass filtering the input signal can improve the robustness of the output considerably. Other pre-processing steps that have been proposed are, for example:

- *pre-whitening*:

If the analyzed signal can be assumed to originate from a pulse-like excitation signal filtered with a transfer function, a common assumption in speech signal processing, a reasonable approach is to reverse the filtering process by estimating the smoothed spectral envelope of the current analysis block and apply the inverse filter to the signal — *pre-whitening* [Kla03a, JNC19]. The goal of the inverse filtering process is to convert the signal back to the initial pulse train in order to improve the results of the following correlation analysis by reducing the impact of the signal timbre. There exist numerous ways to estimate

```
% calculate the acf maximum
afCorr      = xcorr(x(i_start:i_stop), 'coeff');
afCorr      = afCorr((ceil((length(afCorr)/2))+1):end);

% ignore values until threshold was crossed
eta_tmp    = find (afCorr < fMinThresh, 1);
if (~isempty(eta_tmp))
    eta_min = max(eta_min, eta_tmp);
end

% only take into account values after the first minimum
afDeltaCorr = diff(afCorr);
eta_tmp    = find(afDeltaCorr > 0, 1);
if (~isempty(eta_tmp))
    eta_min = max(eta_min, eta_tmp);
end

[fDummy, f(n)] = max(afCorr(1+eta_min:end));

% convert to Hz
f(n) = f_s ./ (f(n) + eta_min);

```

(a) Matlab

```
# calculate the acf
if not x[np.arange(i_start, i_stop + 1)].sum():
    continue
else:
    x_tmp = x[np.arange(i_start, i_stop + 1)]
    afCorr = np.correlate(x_tmp, x_tmp, "full") / np.dot(x_tmp, x_tmp)

afCorr = afCorr[np.arange(iBlockLength, afCorr.size)]

# update eta_min to avoid main lobe
eta_tmp = np.argmax(afCorr < fMinThresh)
eta_min = np.max([eta_min, eta_tmp])

afDeltaCorr = np.diff(afCorr)
eta_tmp = np.argmax(afDeltaCorr > 0)
eta_min = np.max([eta_min, eta_tmp])

# find the coefficients specified in eta
f[n] = np.argmax(afCorr[np.arange(eta_min + 1, afCorr.size)]) + 1

# convert to Hz
f[n] = f_s / (f[n] + eta_min + 1)
```

(b) Python

Figure 7.14: Source code for the extraction of the fundamental frequency from a block of audio samples with the ACF.

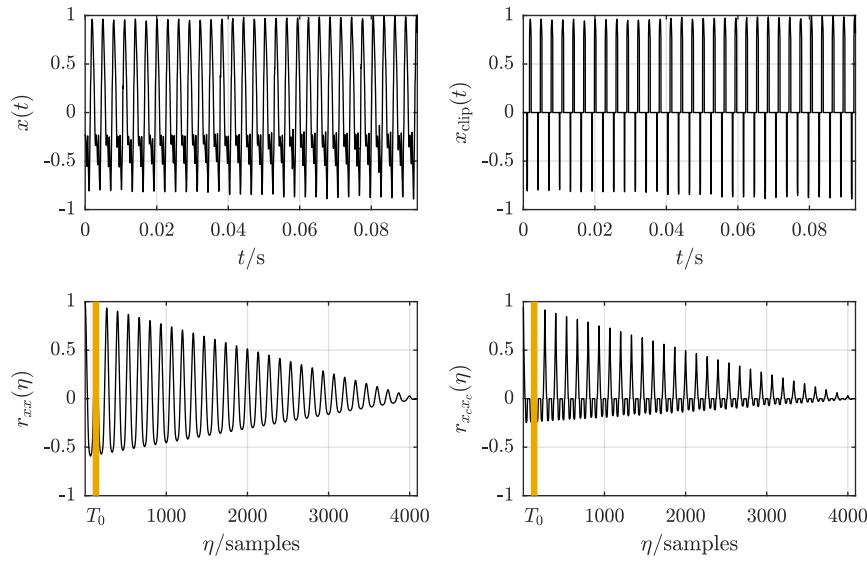


Fig. Gen.: plotFOAcf.m

Figure 7.15: F0 estimation via the Autocorrelation Function (left) and the center-clipped Autocorrelation Function (right). Top: time series, Bottom: one-sided ACF with highlighted fundamental period T_0 .

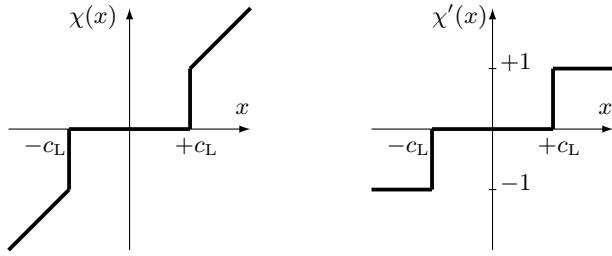


Figure 7.16: Non-linear pre-processing for ACF-based pitch period estimation: standard center clipping (left) and 3-level center clipping (right)

the spectral envelope, including a low-order linear predictive filter [Mak75], an averaged power spectrum [Wel67], or cepstrum-based iterative methods [RR05].

- *center clipping:*

The robustness of the detection can sometimes be improved by using so-called *center clipping* [RS78]. The non-linear function χ as shown in Fig. 7.16 (left) is applied to the input signal $x(i)$

$$x_c(i) = \chi(x(i)). \quad (7.21)$$

The idea of center clipping is to yield a result with more easily identifiable peaks. Typical thresholds c_L are chosen between 30% and 60% of the (instantaneous) maximum amplitude. An alternative 3-level center clipping function χ' is shown in Fig. 7.16 (right). This function allows a very efficient computation of the ACF because the input signal then only has the three possible amplitudes $-1, 0, +1$. This performance optimization is usually not necessary anymore on modern hardware.

7.3.3.3 Average Magnitude Difference Function

The *Average Magnitude Difference Function (AMDF)* is similar to the ACF but avoids the use of multiplications and is therefore computationally efficient. The AMDF is computed by [RSC⁺74]

$$\text{AMDF}_{xx}(\eta, n) = \frac{1}{i_e(n) - i_s(n) + 1} \sum_{i=i_s(n)}^{i_e(n)-\eta} |x(i) - x(i + \eta)|. \quad (7.22)$$

The estimated fundamental period length is then chosen to be the lag of the overall minimum if its value is also smaller than a certain (signal adaptive) threshold. The popular pitch tracking algorithm YIN utilizes a variant of the AMDF [CK02].

The AMDF can also be combined with the ACF in an AMDF-Weighted Autocorrelation Function. There are indications that this weighting leads to more robust results [SK01]:

$$r'_{xx}(\eta, n) = \frac{r_{xx}(\eta, n)}{\text{AMDF}_{xx}(\eta, n) + 1}. \quad (7.23)$$

Figure 7.17 shows the implementation of computing the AMDF (note that the code for finding the minimum is omitted in this example. Figure 7.18 visualizes the resulting AMDF (left) as well as the AMDF-weighted ACF (right) with the resulting fundamental frequency estimates. Note that in case of this mostly simple, unweakened implementation, the AMDF incorrectly returns a multiple of the correct fundamental frequency. With the weighting, however, the correct fundamental frequency is estimated.

```

AMDF      = ones(1, K);
for (eta=0:min(K-1,eta_max-1))
    AMDF(eta+1) = sum(abs(x(1:K-1-eta)-x(eta+2:end)))/K;

```

(a) Matlab

```

afAmdf = np.ones(K)
for eta in range(0, np.min([K, eta_max + 1])):
    afAmdf[eta] = np.sum(np.abs(x[np.arange(0, K - 1 - eta)] - x[np.arange(eta + 1,

```

(b) Python

Figure 7.17: Source code for the extraction of the fundamental frequency from a block of audio samples with the AMDF.

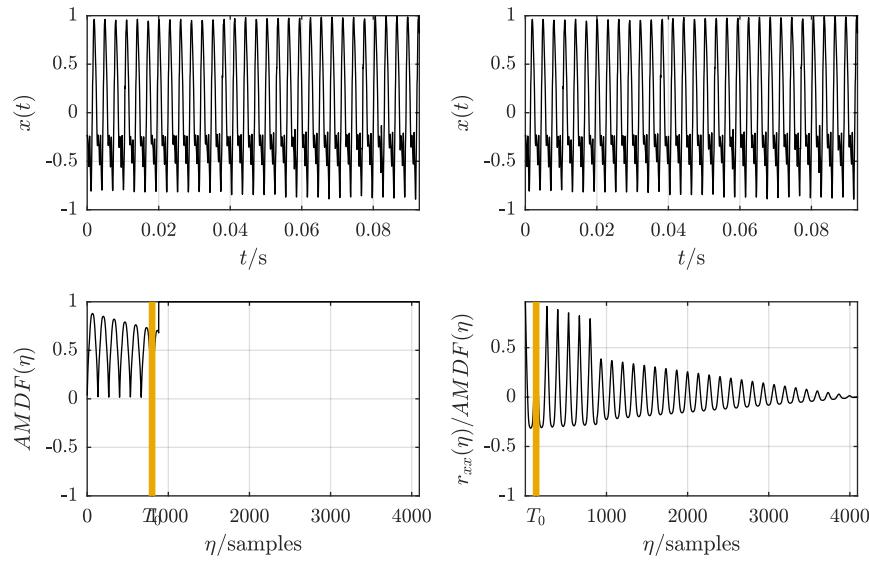


Fig. Gen.: [plotF0Andf.m](#)

Figure 7.18: F0 estimation via the Average Magnitude Difference Function (left) and the AMDF-weighted Autocorrelation Function (right). Top: time series, Bottom: one-sided ACF with highlighted estimated fundamental period T_0 **change xticklabel to \hat{T}_0 to clarify its the estimate.**

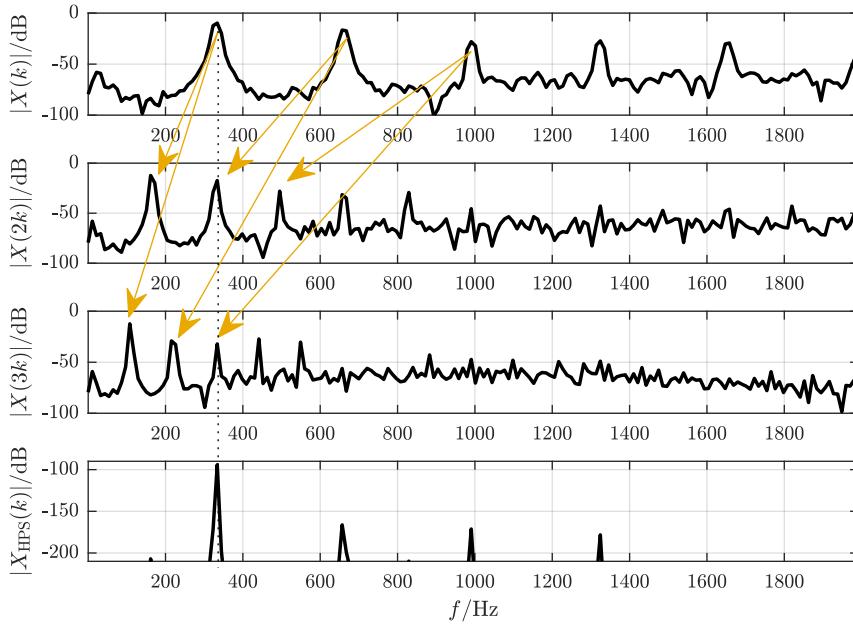


Fig. Gen.: plotF0HpsMethod.m

Figure 7.19: Compressed spectra for the computation of the HPS (bottom).

7.3.3.4 Harmonic Product Spectrum and Harmonic Sum Spectrum

For detecting the fundamental frequency, the frequency domain has the disadvantage of insufficient frequency resolution for low frequencies (see Sect. 7.3.1.2) but has the advantage of being a more accessible and intuitive representation for pitch tracking: it is (for tonal signals) a sparse representation with clear peaks at the harmonics and the frequency axis allows direct access to the target value. The most trivial way of estimating the fundamental frequency from the magnitude spectrum is picking the maximum value. This is, however, a very unreliable way of estimation as the fundamental frequency is often not the most salient harmonic (compare Pg. 112). Therefore, most frequency-domain methods for fundamental frequency estimation try to take advantage of the comb-like structure of the spectrum of a pitched signal.

The *Harmonic Product Spectrum (HPS)* is an efficient method for finding this periodic harmonic pattern [Sch68, Nol69]. It is defined by

$$X_{\text{HPS}}(k, n) = \prod_{j=1}^{\mathcal{O}} |X(j \cdot k, n)|^2. \quad (7.24)$$

The parameter \mathcal{O} is the order of the HPS. Alternative implementations of the HPS use the magnitude spectrum.

The idea of the HPS is that the compression of the frequency axis by integer factors j causes higher harmonics at multiples of the fundamental frequency to coincide at the bin of the fundamental frequency. Thus, the first \mathcal{O} harmonics will be mapped to their fundamental frequency. Since the harmonics can be assumed to have significantly higher power than any other signal components, the resulting (harmonic product) spectrum $X_{\text{HPS}}(k, n)$ should have a clearly identifiable peak at the fundamental frequency. Figure 7.19 visualizes the compression of the spectrum and how it leads to the higher harmonics mapped onto lower harmonics.

Figure 7.20 exemplifies this: the resulting peak in the HPS has a significantly higher distance to the second highest peak than in the original spectrum. Figure 7.21 shows the implementation of fundamental frequency extraction with HPS.

While being efficient and algorithmically “neat,” the HPS has some drawbacks. If the fundamental frequency of the input signal is not located exactly on a frequency bin but between two bins, then the maxima of higher order harmonics will not be taken into account for the decimated spectra. The likelihood of missing the locations of the maxima increases with j . Two possible work-arounds can be used, but both will result in less efficient and more complicated implementations:

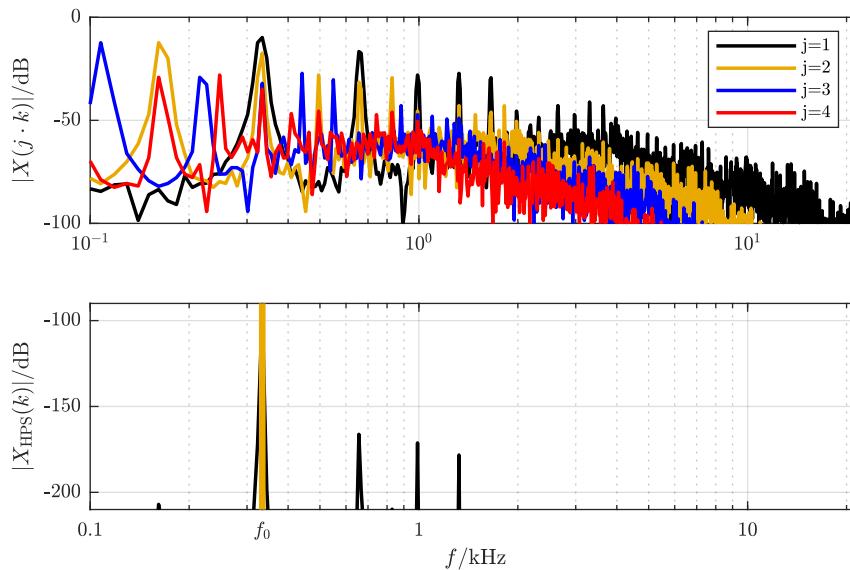


Figure 7.20: F0 estimation via the Harmonic Product Spectrum: Compressed spectra with $j = 1, 2, 3, 4$ (top) and resulting HPS (bottom).

Fig. Gen.: plotFOHps.m

```

afHps    = X;
k_min   = round(f_min/f_s * 2 * (size(X,1)-1))+1;

% compute the HPS
for (j = 2:iOrder)
    afHps    = afHps .* [X(1:j:end,:); zeros(size(X,1)-size(X(1:j:end,:),1), size(X,2))];
end

% find max index and convert to Hz
[fDummy,f] = max(afHps(k_min:end,:),[],1);
f          = (f + k_min - 2) / (size(X,1)-1) * f_s/2;
f(sum(afHps,1) == 0) = 0;

```

(a) Matlab

```

iLen = int((X.shape[0] - 1) / iOrder)
afHps = X[np.arange(0, iLen), :]
k_min = int(round(f_min / f_s * 2 * (X.shape[0] - 1) / 2))

# compute the HPS
for j in range(1, iOrder):
    X_d = X[:,(j + 1), :]
    afHps *= X_d[np.arange(0, iLen), :]

f = np.argmax(afHps[np.arange(k_min, afHps.shape[0])])

```

find max index and convert to Hz
f = (f + k_min) / (X.shape[0] - 1) * f_s / 2

(b) Python

Figure 7.21: Source code for the extraction of the fundamental frequency from a STFT with HPS.

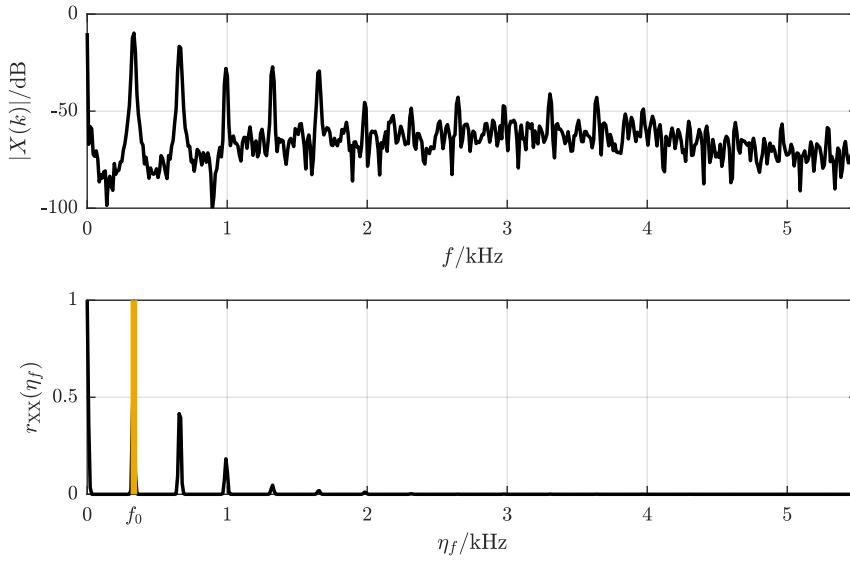
Fig. Gen.: [plotF0ActOffFft.m](#)

Figure 7.22: F0 estimation via the Autocorrelation Function of the magnitude spectrum: Input magnitude spectrum (top) and resulting ACF (bottom).

- increase the frequency resolution by using longer STFT block sizes or by interpolating (up-sampling) the spectrum, or
- take the maximum within a bin range for the multiplication. The bin range will have to increase by ± 1 bin with every increment of j .

If one of the harmonics is zero or near zero, the detection of the fundamental frequency will fail as the HPS at the fundamental frequency bin will be scaled with (near) zero. One approach to avoid this is to compute the *Harmonic Sum Spectrum (HSS)* with a definition similar to the HPS [Nol69]:

$$X_{\text{HSS}}(k, n) = \sum_{j=1}^{\mathcal{O}} |X(j \cdot k, n)|^2. \quad (7.25)$$

While the HSS is more robust against missing harmonics, the resulting maximum is usually not as pronounced as in the HPS.

The cumulative nature of both HPS and HSS frequently leads to octave errors.

7.3.3.5 Autocorrelation Function of the Magnitude Spectrum

The usage of the ACF is, as pointed out in other sections (see Sects. A.3, 3.6.12, and 7.3.3.2), a useful and intuitive approach to finding periodicities. Since the harmonics are equally spaced in the magnitude spectrum and the distance of neighboring harmonics equals the fundamental frequency, finding this periodicity is equivalent to finding the fundamental frequency. The lag of the maximum of the ACF is an estimate of the fundamental frequency in spectral bins. Figure 7.22 shows the result of the ACF of a magnitude spectrum. Figure 7.23 shows the implementation of fundamental frequency extraction with ACF of the STFT.

The robustness of the detection might be improved by computing a circular correlation of the symmetric spectrum and by manually adding a maximum at frequency bin 0 to fill a potential 'gap' in the harmonic series.

7.3.3.6 Cepstral Pitch Detection

Cepstral pitch detection is based on the assumption that the analysis signal $x(i)$ is the result of a convolution of an excitation signal $e(i)$ with a transfer function $h(i)$

$$x(i) = e(i) * h(i). \quad (7.26)$$

```
% use spectral symmetry for robustness
X(1,:) = max(max(X));
X       = [fliplr(X); X];

% compute the ACF
for (n = 1: size(X,2))
    eta_min = round(f_min/f_s * (size(X,1)-2));
    afCorr  = xcorr(X(:,n),'coeff');
    afCorr  = afCorr((ceil((length(afCorr)/2))+1):end);

    % find local maxima
    [fDummy,eta_peak] = findpeaks(afCorr);

    eta_min = max(eta_min, find(eta_peak > eta_min,1));
    [fDummy, f(n)] = max(afCorr(eta_min:end));
end

% find max index and convert to Hz (note: X has double length)
f      = (f + eta_min - 1) / (size(X,1)-2) * f_s;

```

(a) Matlab

```
# use spectral symmetry for robustness
X[0, :] = np.max(X)
X = np.concatenate((np.fliplr(X), X), axis=0)

# compute the ACF
for n in range(0, X.shape[1]):

    if X[:, n].sum() < 1e-20:
        continue

    eta_min = int(round(f_min / f_s * (X.shape[0] - 2))) - 1

    afCorr = np.correlate(X[:, n], X[:, n], "full") / np.dot(X[:, n], X[:, n])
    afCorr = afCorr[np.arange(X.shape[0], afCorr.size)]

    # find the highest local maximum
    iPeaks = find_peaks(afCorr, height=0)
    if iPeaks[0].size:
        eta_min = np.max([eta_min, iPeaks[0][0] - 1])
    f[n] = np.argmax(afCorr[np.arange(eta_min, afCorr.size)]) + 1

    # find max index and convert to Hz (note: X has double length)
    f[n] = (f[n] + eta_min) / (X.shape[0] - 2) * f_s


```

(b) Python

Figure 7.23: Source code for the extraction of the fundamental frequency from a STFT with the ACF.

This is a common model in speech signal processing; the excitation signal originates from the air streaming through the *glottis*. This excitation signal $e(i)$ consists of quasi-periodic pulses in the case of voiced (tonal) sounds [RS78]. The vocal tract and the nasal tract act as tubes that shape the frequency spectrum with the transfer function $h(i)$.

Equation (7.26) can be rephrased in the frequency domain (compare Sect. B.1.3) as

$$X(j\omega) = E(j\omega) \cdot H(j\omega). \quad (7.27)$$

If a (complex) logarithm is applied to this equation, the result is

$$\begin{aligned} \log(X(j\omega)) &= \log(E(j\omega) \cdot H(j\omega)) \\ &= \log(E(j\omega)) + \log(H(j\omega)). \end{aligned} \quad (7.28)$$

Applying the logarithm allowed us to replace the multiplication with an addition. We define the *cepstrum* $c_x(i)$ to be the inverted logarithmic spectrum

$$\begin{aligned} c_x(i) &= \mathfrak{F}^{-1}\{\log(X(j\omega))\} \\ &= \mathfrak{F}^{-1}\{\log(E(j\omega)) + \log(H(j\omega))\} \\ &= \mathfrak{F}^{-1}\{\log(E(j\omega))\} + \mathfrak{F}^{-1}\{\log(H(j\omega))\}. \end{aligned} \quad (7.29)$$

The inverse-transformed spectrum thus consists of signals $e(i)$ and $h(i)$ being *added* (although logarithmically) instead of being convolved. In order to emphasize the difference between the original time domain and the cepstral domain, the term *quefrency* is suggested to be used as axis label.

The cepstrum can be approximated by only using the magnitude spectrum

$$\hat{c}_x(i_s(n) \dots i_e(n)) = \sum_{k=0}^{\kappa/2-1} \log(|X(k, n)|) e^{jki\Delta\Omega} \quad (7.30)$$

and avoiding the use of the complex logarithm.

The cepstrum has two properties that are of particular interest in the context of pitch detection [Nol64, Sch68]. First, a pulse-like excitation signal will also lead to a pulse in the cepstrum, and, second, the cepstrum will decay rapidly for large i . Therefore, the detection of a peak (or more accurately a pulse train) in the cepstrum should give the period length of the fundamental frequency. Figure 7.24 shows the cepstrum of an exemplary magnitude spectrum.

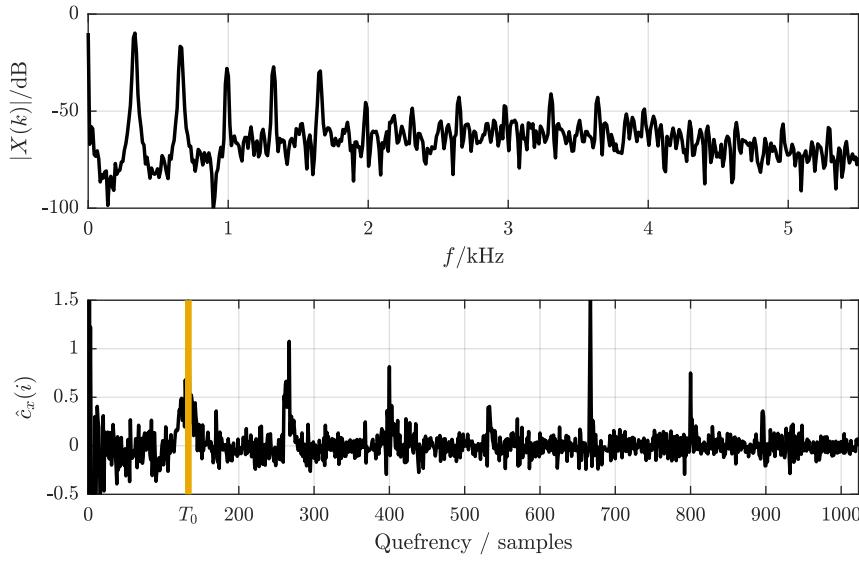
Fig. Gen.: [plotF0Cepstrum.m](#)

Figure 7.24: F0 estimation via the Cepstrum: Input magnitude spectrum (top) and cepstrum of this magnitude spectrum (bottom).

7.3.3.7 Auditory Motivated Pitch Tracking

A class of pitch detection algorithms use models of human pitch perception to determine the pitch of a signal. Meddis and O'Mard describe the processing stages of such algorithms as [MO97]:

- (i) band-pass filtering,
- (ii) HWR [see Eq. (3.69)] and band processing,
- (iii) within-band periodicity extraction, and
- (iv) across-band aggregation of periodicity estimates.

The filterbank used for band-pass filtering is frequently a gammatone filterbank, one of the “standard” filterbanks for auditory processing. Gammatone filters have been introduced in Sect. 3.4.4.

Klapuri pointed out the importance of HWR on the filter channel outputs for fundamental pitch estimation [Kla06]. Figure 7.25 shows this effect for one band containing the 13th–17th harmonics of a periodic sound: HWR results in new frequency components being generated at the distance of frequency components in the signal and thus around the fundamental frequency. Subsequent low pass filtering of this band output removes the (original and new) high frequency components. Further band processing might include gain compression by, for instance, scaling the variance to unity [Kla06].

A periodicity analysis of the filterbank outputs $z_c(i)$ may then be used for the detection of the fundamental period length by, for example, applying an ACF to each channel

$$r_{zz}(c, n, \eta) = \sum_{\eta=0}^{\mathcal{K}-1} z_c(i) \cdot z_c(i + \eta) \quad (7.31)$$

and summing the resulting ACFs

$$r_A(n, \eta) = \sum_{c=0}^{\mathcal{C}-1} r_{zz}(c, n, \eta). \quad (7.32)$$

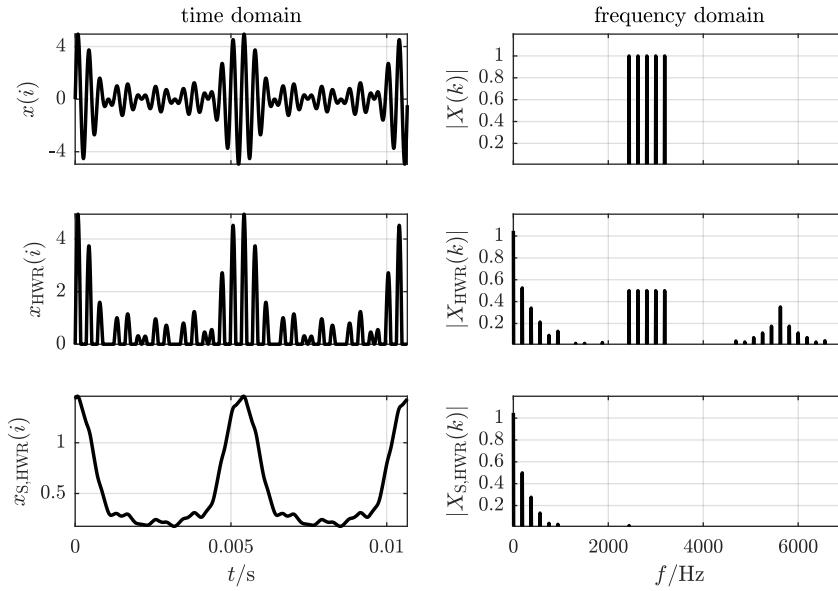
Fig. Gen.: [plotF0Auditory.m](#)

Figure 7.25: Time domain (left) and frequency domain (right) for a signal consisting of the 13th to 17th partial of a sound with a fundamental frequency of 187.5 Hz (top), the corresponding signal subjected to HWR (mid), and the low-pass filtered signal subjected to HWR (bottom).

7.3.4 Polyphonic Input Signals

Most of the algorithms for fundamental frequency detection outlined above will not work well in the case of polyphonic signals with multiple simultaneous fundamental frequencies. The number of simultaneous pitches in polyphonic signals depends on genre, epoch, and musical context. In most cases the number of independent voices will be between one and eight.

One of the first *multi-pitch detection* systems was presented by Chafe et al. in 1985 [CJK⁺85]. They proposed to pick spectral magnitude peaks in a multi-resolution FT and derive candidates for fundamental frequencies by grouping the peaks with respect to their frequency ratio. The detected candidates are then tracked over time in the spectrogram to discard spurious detections. Nowadays, multi-pitch detection is a lively research field with numerous methods and approaches of which only a few basic ones will be described below.

A special case that will not be covered in the following is pre-dominant melody extraction: in this case, the input data is polyphonic but only the salient (often vocal) melody pitches are extracted [Got00].

7.3.4.1 Iterative Subtraction

One class of multi-pitch detection systems is based on the principle of iterative subtraction. Here, a fundamental frequency detection algorithm for *monophonic* input signals is applied to detect the predominant fundamental frequency, then find a way to subtract this and related (mostly harmonic) frequency components from the original signal and repeat the process on the residual until the criterion for termination has been reached.

An early reference to such an algorithm in the spectral domain has been published by Parsons [Par76] who — inspired by the work of Schroeder [Sch68] — constructed a histogram of spectral peaks and their integer submultiples, chose the largest peak for the first fundamental frequency estimate, and removed this estimate and its multiples from the histogram to detect the second fundamental frequency.

Klapuri et al. proposed two adaptations on the iterative subtraction procedure. They use an auditory-motivated monophonic fundamental frequency detection algorithm (see Sect. 7.3.3.7) to find the predominant, most salient fundamental frequency and estimate the spectrum of this pitch to subtract it from the original spectrum for the detection of additional pitches [Kvh00, Kla01].

Cheveigné proposed a system for the tracking of multiple pitches in the time domain [CK99, Che06]. First,

the squared AMDF (compare Sect. 7.3.3.3) is computed with

$$\text{ASMDF}_{xx}(\eta, n) = \frac{1}{i_e(n) - i_s(n) + 1} \sum_{i=i_s(n)}^{i_e(n)} (x(i) - x(i + \eta))^2. \quad (7.33)$$

Then, the most salient period length is found at the lag η_{\min} of the ASMDF minimum. In order to remove the detected frequency and its harmonics from the signal, a comb cancellation filter is applied to the signal with a delay corresponding to the lag of the detected minimum. The Finite Impulse Response (FIR) comb filter has the impulse response

$$h(i) = \delta(i) - \delta(i - \eta_{\min}) \quad (7.34)$$

and attenuates both the detected fundamental frequency and its harmonics at integer multiples. The remaining residual signal can then be used as input for the detection of a second fundamental frequency. It is also possible to implement this approach non-iteratively by using an exhaustive search. In this case, two cascaded cancellation filters can be applied to the signal in all possible combinations of η_1 and η_2 . The most likely pair of fundamental period lengths is the combination of η_1 and η_2 minimizing the overall output power of the output signals.

A similar idea has been presented by Miwa et al., who propose to use a cascade of comb filters, each comb filter subtracting a fundamental frequency with its harmonics from the input signal [MTS99]. When the output of one filter equals 0, they know that the signal only contains the pitch at this filter frequency plus an unknown number of pitches represented by the preceding filters. Rerunning the analysis after shuffling the filter order then allows for detecting all pitches in the signal.

Meddis and Hewitt use an auditory approach similar to the one described in Sect. 7.3.3.7 for detecting one fundamental frequency and then use all remaining filter channels, i.e., filter channels not showing a peak at the detected frequency, to detect more fundamental frequencies [MH92]. The iteration process is terminated if more than 80% of the channels have been removed.

Klapuri also proposes a variation of the auditory approach by computing the normalized filterbank outputs after HWR [Kla05]. He then computes the STFT of each filter band, sums their magnitudes

$$Z(k, n) = \sum_{c=0}^{C-1} |Z_c(k, n)|, \quad (7.35)$$

and weights the resulting overall spectrum $Z(k, n)$ with a low-pass filter transfer function. To identify possible fundamental frequency candidates, a set of delta pulse templates is used representing every detectable fundamental frequency with its harmonics. Similar to the calculation of the HSS, these templates are multiplied with the spectrum, and the result can be used as an estimate for the salience of each fundamental frequency. The components of the most salient frequency are then being removed from the spectrum to find the next fundamental frequency in an iterative process.

Another auditory-inspired multi-pitch detection algorithm focusing on computational efficiency has been published by Karjalainen and Tolonen [KT99, TK00]. They propose to use only two auditory bands with a cut-off frequency of 1 kHz and follow the general process described in Sect. 7.3.3.7 with HWR and smoothing. The approach has a few tweaks, however, that are interesting to consider. First, they apply pre-whitening to flatten the spectral envelope in a pre-processing step. Second, they modify the ACF calculation by setting $\beta = 2/3$ in Eq. (A.44). Finally, the summary ACF is then harmonically processed with an idea similar to HPS. The harmonic post-processing is an iterative process in which an interpolated, time-scaled and half-wave rectified version is subtracted from the half-wave rectified ACF $r(\eta, n) = r(j, \eta, n)$. The repeated application of HWR in combination with scaling aims at discarding frequencies other than the fundamental frequencies so that peaks only remain at actual fundamental frequencies and not their multiples.

7.3.4.2 Non-negative Matrix Factorization

Non-negative Matrix Factorization (NMF), introduced by Lee and Seung [LS01a], has attracted considerable attention in the context of multi-pitch detection during the last decade. It decomposes a time-frequency

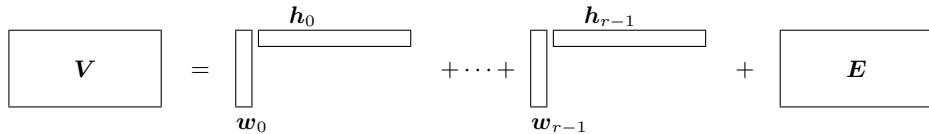


Figure 7.26: Visualization of Eq. (7.37) with example dimensions

representation into a matrix containing the spectra of the individual sounds and another matrix containing the information on when each of the individual spectra is active.

NMF is an iterative algorithm which estimates an approximation $\hat{\mathbf{V}}$ of a given matrix \mathbf{V} through a multiplication of the iteratively estimated matrices \mathbf{W} and \mathbf{H} :

$$\hat{\mathbf{V}} \approx \mathbf{W} \cdot \mathbf{H}. \quad (7.36)$$

If the dimension of both \mathbf{V} and $\hat{\mathbf{V}}$ is $k \times n$, then the dimensions of \mathbf{W} and \mathbf{H} will be $k \times r$ and $r \times n$, respectively, with r indicating the NMF rank. No matrix can contain negative entries. Alternatively, the problem can be formulated as [CZPA09]

$$\mathbf{V} = \sum_{i=0}^{r-1} \mathbf{w}_i \cdot \mathbf{h}_i + \mathbf{E}, \quad (7.37)$$

with $\mathbf{V} \in \mathbb{R}^{k \times n}$, $\mathbf{W} = [\mathbf{w}_0, \mathbf{w}_1, \dots, \mathbf{w}_{r-1}] \in \mathbb{R}^{k \times r}$, and $\mathbf{H} = [\mathbf{h}_0, \mathbf{h}_1, \dots, \mathbf{h}_{r-1}]^T \in \mathbb{R}^{r \times n}$. Figure 7.26 visualizes these dimensions. Once the elements of the error matrix \mathbf{E} are minimized, the matrix \mathbf{V} can be represented by the combination of the *dictionary* matrix \mathbf{W} and the *activation* matrix \mathbf{H} . The goal of the optimization algorithm is to iteratively minimize the error matrix $\mathbf{E} = \mathbf{V} - \hat{\mathbf{V}}$, which is the distance of the original matrix \mathbf{V} and the estimated matrix $\hat{\mathbf{V}} = \mathbf{W} \cdot \mathbf{H}$. The two most common loss functions are the squared Euclidean distance

$$d_{\text{EU}}(\mathbf{V}, \hat{\mathbf{V}}) = \left\| \mathbf{V} - \hat{\mathbf{V}} \right\|_2^2 = \sum_{kn} v_{kn} - \hat{v}_{kn}, \text{ and} \quad (7.38)$$

the generalized Kullback-Leibler divergence

$$d_{\text{KL}}(\mathbf{V}, \hat{\mathbf{V}}) = \sum_{kn} v_{kn} \log \left(\frac{v_{kn}}{\hat{v}_{kn}} \right) - v_{kn} + \hat{v}_{kn}. \quad (7.39)$$

Also compare Eqs. (4.5) and (4.9).

Gradient descent can be used to minimize the distance; in practice, the matrices \mathbf{W} and \mathbf{H} can be updated alternating with multiplicative update rules [LS01a].

Given the introduction of NMF above, the question remains what practical use this algorithm has for pitch tracking. Smaragdis and Brown answered this question by applying NMF to the approximation of the non-negative magnitude spectrogram [SB03]. Under the assumption that individual sound components such as pitches simply add up in the magnitude spectrum, the magnitude spectrum is decomposed in r spectral templates of the sound components in the dictionary matrix \mathbf{W} and their salience over blocks in the activation matrix \mathbf{H} .

Figure 7.27 shows an example for a factorization. The input signal is constructed from three signals: the pitch D2 played by a horn, the pitch F4 played by an oboe, and the pitch B4 played by a violin. First, the three signals are concatenated (horn 0–3 s, oboe 3–4 s, violin 4–6 s), then mixed to appear simultaneously at a lower level (6–9 s). If we look at the automatically estimated dictionary matrix, we see that the algorithm successfully separated the three pitches into the individual templates \mathbf{w}_0 (oboe), \mathbf{w}_1 (violin), \mathbf{w}_2 (horn). The activation matrix on the right indicates clearly the contribution of each template to the mixture at each point in time.

Thus, NMF is capable of separating different pitches in a polyphonic mixture. However, to apply this approach to pitch tracking, each dictionary template has to be labeled with the corresponding pitch. If the factorization worked well, this should be an easy task as each template contains only one harmonic series. As the templates are unordered and the pitch content might not be known before-hand, this has to be done after the

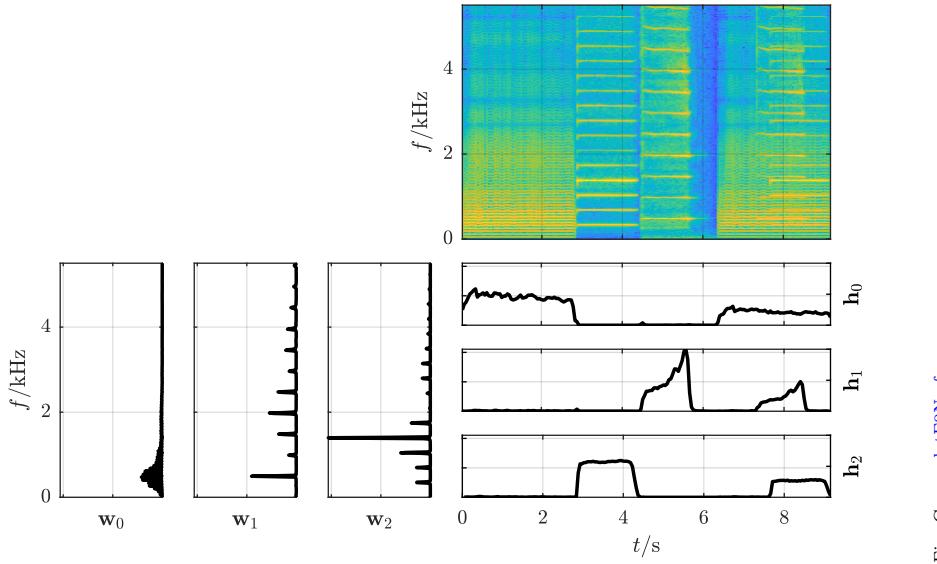
Fig. Gen.: [plotF0Nmf.m](#)

Figure 7.27: Example for the utilization of NMF for detecting the individual pitches; Input spectrogram (top), estimated template vectors (bottom left), and estimated activation vectors (bottom right) for a signal with three pitches played first individually and then mixed together.

factorization. It is also possible to assign the pitches only once instead of doing that after each factorization; in this case, the dictionary matrix is kept fixed after the assignment and the factorization of new signals only updates the activation matrix. Each row of the resulting matrix will then indicate the salience of one pitch over time.

Figure 7.28 shows the implementation of the iteration loop of NMF.

The algorithmic simplicity and efficiency of the NMF approach make it an appealing method for fundamental pitch extraction. There are a number of challenges, however, that potentially impact the usability and robustness of the default algorithm. First, there is an obvious tradeoff between frequency resolution and time resolution (compare Sect. 7.3.1.2) that usually leads to insufficient resolution for low frequency signals. Second, the rank r is a hyper-parameter that should preferably be adjusted for each individual input signal depending on the pitch content of the signal. Choosing a high rank will likely lead to multiple templates per pitch, making the pitch assignment hard. Choosing a low rank will negatively impact how well the spectrogram can be reconstructed and will lead to one template representing multiple pitches. Third, there will be not only periodic, pitched components in a real world audio signal. Percussive, noise-like components have to be modeled as well in some form, requiring additional unpitched templates. Fourth, differences in intonation cannot be easily modeled, as the template only contains one harmonic series per pitch. Transposing an input signal by, e.g., a quartertone will result in different templates. Playing techniques such as vibrato cannot be properly modeled due to this pitch quantization. Fifth, multi-timbral sources with multiple instruments can be hard to model. If two instruments distinct in timbre play the same pitch, the template for this pitch will probably converge to some compromise spectral shape. Sixth, the basic NMF approach assumes that the spectral shape of a template is time-invariant and only scaled up and down by the activation; therefore, it cannot model component spectra that change over time. Thus, instrument sounds with timbre changes within one note are difficult to model. Seventh, when adapting the templates to real-world music data, an imbalanced pitch distribution in the data might lead to inaccurate or even missing templates for infrequently occurring pitches as the algorithm only minimizes the global, overall error.

After the introduction of NMF to polyphonic pitch tracking [SB03], many publications followed refining the pitch tracking process with NMF. One very common modification of the standard algorithm in the context of pitch tracking is the use of pre-trained templates with each template corresponding to one specific pitch. This makes post-analysis for the pitch-mapping of each template unnecessary, because each template can be assigned

```
%start iteration
while (count < iMaxIteration)

    % current estimate
    approx = W*H;

    % update
    if H_update
        H = H .* (W.* (V./approx))./(W.*rep);
    end
    if W_update
        W = W .* ((V./approx)*H')./(rep*H');
    end

    %normalize
    for i = 1:(iRank)
        W(:,i) = W(:,i)./(norm(W(:,i),1));
    end

    %calculate variation between iterations
    count = count + 1;
    err(count) = KLDivergence(V, (W*H)) + fSparsity * norm(H, 1);

    if (count >=2)
        if (abs(err(count) - err(count - 1)) / (err(1) - err(count) + realmin)) < 0.001
            break;
        end
    end
end
```



```
%start iteration
while (count < iMaxIteration)

    % current estimate
    approx = W*H;

    % update
    if H_update
        H = H .* (W.* (V./approx))./(W.*rep);
    end
    if W_update
        W = W .* ((V./approx)*H')./(rep*H');
    end

    %normalize
    for i = 1:(iRank)
        W(:,i) = W(:,i)./(norm(W(:,i),1));
    end

    %calculate variation between iterations
    count = count + 1;
    err(count) = KLDivergence(V, (W*H)) + fSparsity

    if (count >=2)
        if (abs(err(count) - err(count - 1)) / (err(1) - err(count) + realmin)) < 0.001
            break;
        end
    end
end
```

(a) Matlab

(b) Python

Figure 7.28: Implementation of NMF.
implement python

a specific pitch. During inference, the template matrix is kept fixed and does not have to be updated, only the activation matrix is iteratively updated [Con06, DCL10].

Many other improvements to the NMF approach for multi-pitch tracking have been proposed over time. Replacing the loss function, for example, can improve the results for specific use cases [FBD09, DCL10]. Other researchers proposed modifying an established loss functions; Cont applied a sparsity constraint on the activation matrix to ensure a minimum of active templates at each time [Con06]. The factorization approach can also be modified on a more fundamental level by utilizing probabilistic approaches [SRSR08, BBV10, SMSB17] and introducing shift-invariance [SRSR08].

NMF has also been applied to various related and unrelated other tasks in ACA. The most prominent example is probably source separation: if individual templates can be assigned to specific instruments and/or speakers, approximating the magnitude spectrogram of only one source is possible by setting the activations of all other sources to zero. This source signal can then be reconstructed by using the original phase for computing the time domain signal from the masked magnitude spectrogram [Vir07, KYKC11, GE11, SFM⁺14]. The method has also been utilized for instrument recognition in general [BKK06b, BKK06a] and for the transcription for non-pitched drum sounds [APF09, WL15]. A more detailed overview on instrument recognition can be found in Sect. 15. Other application examples range from identifying structural components in music [KS10, WB10, NJ13], recognizing mixed samples [GL17], and denoising [FBD09].

7.3.4.3 Other Approaches

Probabilistic approaches aim at modeling the pitch tracking problem by means of a statistical framework. To give only one example, Kameoka et al. model a tone in the frequency domain as a superposition of weighted Gaussian distributions at integer multiples of the fundamental frequency and its power envelope function in the time domain by overlapping Gaussian distributions [KNS07]. The spectrogram is then decomposed into clusters which model individual notes.

While the majority of state-of-the-art (Multi-)Pitch Detection are now based on Neural Networks, the adaption of DNNs has been slower than for other tasks. Marolt successfully introduced neural networks to the task of piano transcription in the early 2000s [Mar04]. Later, different input representations and network architectures have been proposed (e.g., [BS12, KDK⁺16, HES⁺18, HSS⁺21]). Many of these methods focus exclusively on piano music and progress seems to be slowed by the lack of accurately annotated training data [BDDE19].

7.3.5 Evaluation

The evaluation of pitch tracking systems seems easy enough: predict the pitch and verify it against the ground truth data. A closer look at the ground truth of some available datasets, however, reveals a more varied picture. This is partly due to specific task constraints; the available datasets can be roughly categorized into classes of increasing task complexity: (i) the detection of the pitch of single instrument notes, (ii) the detection of the pitches of monophonic melody, (iii) the detection of the pitches of a pre-dominant melody in a polyphonic mixture of instruments, and (iv) the detection of all pitches in a polyphonic mixture of instruments.

Similarly, the datasets have to vary depending on whether the goal is to extract the fundamental frequency trajectory or to transcribe to a score-like result. For instance, comparing two pitch tracking datasets we find that the TONAS dataset [GB13] provides an fundamental frequency value in Hz every 512 audio samples, while MusicNet provides the start and stop times in seconds of individual midi pitches [THK17]. These two examples showcase different views of how the goals of “pitch tracking” as a task are defined and how pitch tracking systems can be evaluated. First, they differ in frequency granularity: MusicNet targets reconstructing a musical score and thus needs only one value per musical pitch, while TONAS provides a “continuous” floating point value for the fundamental frequency, enabling the analysis of intonation and pitch deviations. Second, the time resolution of MusicNet depends on the note length while TONAS provides equidistant annotations with comparably high time resolution. Although both these datasets have been designed for the seemingly identical task of pitch tracking, they differ considerably in their ground truth annotations. Inquiries such as the analysis of intonation, time-variant pitch deviations, and performance techniques such as vibrato, however, are only possible with high granularity. Thus, not only could it be argued that they represent (slightly) different tasks but they also require different evaluation approaches and metrics.

7.3.5.1 Metrics

The non-linearity of human pitch perception needs to be reflected by the evaluation. Therefore, all metrics should be computed in the pitch domain rather than the frequency domain to ensure that the computed evaluation results are at least somewhat meaningful from a perceptual point of view. It is common to run the evaluation in the MIDI pitch domain as introduced in Eq. (7.11).

If the ground truth provides only (quantized) MIDI pitches, the evaluation simplifies to a simple classification task in which each pitch is either correct or incorrect. In that case, the detected fundamental frequencies are mapped to the closest equally tempered pitch and classification metrics can be reported. Typical classification-based metrics are the accuracy and F-Measure, which are based on the number of correct and incorrect predictions (compare Sect. 6.2.1). Given a set of \mathcal{N} ground truth pitch labels p_{GT} and the corresponding estimated pitches \hat{p} , Salamon et al. [SGER14] summarized a set of commonly used metrics as raw pitch accuracy

$$RPA = \frac{\sum_{\forall n} TP_n}{\mathcal{N}} \quad (7.40)$$

with

$$TP_n = \begin{cases} 0, & \text{if } |p_{GT}(n) - \hat{p}(n)| \geq 0.5 \\ 1, & \text{otherwise} \end{cases}$$

and raw chroma accuracy counting octave errors as match

$$RCA = \frac{\sum_{\forall n} TP_{chroma,n}}{\mathcal{N}} \quad (7.41)$$

with

$$TP_{chroma,n} = \begin{cases} 0, & \text{if } \mod(|p_{GT}(n) - \hat{p}(n)|, 12) \geq 0.5 \\ 1, & \text{otherwise} \end{cases} .$$

In addition, they propose an overall accuracy metric combining the pitch detection accuracy with the voicing detection accuracy. There are other metrics that have been proposed for multi-pitch detection systems, including

a substitution error, miss and false alarm errors, and an overall error score [PE06]. Note that in some cases, an additional variable to be evaluated is the voicing (voice present or not), in the case of vocals also referred to as *(Vocal) Activity*. Please see Sect. 15 for evaluating such an activity detection.

If the ground truth provides fundamental frequencies (or more precisely, floating point resolution pitch values), the classification metrics above can only be computed by assuming a detection tolerance (± 50 cent in the example above). Deviation-based metrics, however, may provide more insights into the algorithm accuracy. A typical standard metric is, for example, the MAE (compare Sect. 6.2.2). Other metrics, such as the MSE or the standard deviation from the ground truth are also potential measures for accuracy. Bittner and Bosch proposed to generalize the binary metrics above to deviation-based metrics [BB19].

The metrics mentioned above are all *frame-level* metrics, i.e., they are based on each ground truth value representing a short span of time. The time resolution of the ground truth will vary between datasets and is unlikely to match the time resolution of the system to be evaluated. While there is no obvious solution to this problem beyond changing the system’s output time resolution, an interpolation of the predicted values is probably the best way to deal with it. Any non-trivial interpolation that does not violate the sampling theorem should be applicable. Obviously, the ground truth needs to remain unchanged.

Note that evaluating only the correctly detected pitches might be insufficient for music transcription systems, as there are other (related) results that might need to be evaluated, for example, voicing and metrical structure [MS18].

7.3.5.2 Datasets

Although pitch transcription is a core task in the analysis of musical audio, the amount, size, and breadth of available annotated data does not necessarily reflect this importance. The issue of annotation, requiring considerable effort from experts, means that dataset annotations are either automatically generated (e.g., MusicNet [THK17]), created through instrument sensors (e.g., MAESTRO [HSR⁺19]) or small (e.g., CSD [CGML18]). There are only few datasets beyond piano and singing (e.g., Guitarsset [XBP⁺18]). In the past, this lack of both data and diversity has impeded the progress of sophisticated methods for pitch transcription.

7.3.5.3 Results

Given the data limitations mentioned above, any evaluation results will only have limited meaning themselves. However, the existing results can still give us an impression of the general capabilities of modern systems. In a recent evaluation, most systems yielded overall accuracies between 0.5 and 0.7 on one dataset and accuracies from 0.2 to 0.5 on a different dataset.⁴ While these data most decidedly contain challenging, multi-instrument recordings, it seems safe to say that current state-of-the-art systems do not appear to be sufficiently reliable for general music transcription.

7.4 Tuning Frequency Estimation

The *tuning frequency* f_{A4} is the frequency of the *concert pitch A4* (also: *standard pitch*) which is used for tuning one or more musical instruments. The tuning frequency is standardized internationally to 440 Hz [16:75], but the exact frequency used by musicians can vary due to various reasons such as the use of historic instruments or timbre preferences. Two performances of the same piece of music using different tuning frequencies will differ in their average pitch height, but will not change the frequency ratios between pitches like temperament.

The range of typical tuning frequencies decreased over the centuries. Table 7.5 shows this range for the past three centuries as deviation from 440 Hz [Bri98].

Nowadays, while for many electronic music productions the “default” tuning frequency of 440 Hz is used, the tuning frequencies of orchestras still deviate from this standard tuning frequency. For example, the Chicago Symphony Orchestra and the New York Philharmonic tune at 442 Hz, while the Berliner Philharmoniker and the

⁴compare the Multiple Fundamental Frequency Estimation results of MIREX 2019 https://www.musicir.org/mirex/wiki/2019:MIREX2019_Results, last retrieved July 4, 2021

Table 7.5: Typical range of deviation of the tuning frequency from 440 Hz over three centuries

Year	<i>Lower Deviation</i>	<i>Upper Deviation</i>
1750	-50 Hz	+30 Hz
1850	-20 Hz	+20 Hz
1950	-5 Hz	+10 Hz

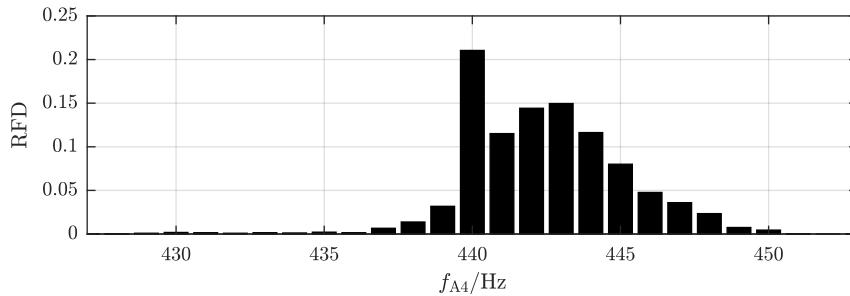
Fig. Gen.: `plotTuningFreqs.m`

Figure 7.29: Distribution of tuning frequencies (from [Ler06]).

Wiener Philharmoniker have a tuning frequency of 443 Hz.⁵ At least in the case of both European orchestras, the tuning frequency was higher in previous decades. The frequencies 442 and 443 Hz correspond to deviations of 7.85 and 11.76 cents from the standard tuning frequency, respectively. In addition to musicians consciously choosing a tuning frequency, historic recordings might also be reproduced at non-standard tuning frequencies because of speed deviations between recording and playback equipment.

There does not exist much data on the actual tuning frequency of real-world recordings. Zhu et al. processed a database of 60 popular and 12 classical pieces⁶ and found only three pieces of this database with a deviation of approximately 2–4 cents from the standard tuning frequency [ZKG05]. Lerch investigated the tuning frequencies in a large proprietary dataset of classical music, consisting of more than 3000 tracks and an overall playing time of approximately 291 hours [Ler06]. The result is shown in Fig. 7.29: the distribution has a maximum at a tuning frequency of 440 Hz; the maximum itself consists of about 21% of the test database. The arithmetic mean of the distribution, however, is at 442.38 Hz (its standard deviation is 2.75 Hz) as the results are clearly skewed towards higher tuning frequencies: the majority of the results (95%) is in the range from 439 to 448 Hz while the percentage of files below 439 Hz is about 3.3%. Note that only 50% of the results have a tuning frequency in the range of 440–443 Hz.

While the studies above assume a static tuning frequency once the instruments have been tuned, this is not necessarily true. The tuning frequency may change over the course of a performance or a recording session. On the one hand, the tuning frequency could be slowly decreasing as it sometimes happens at *a cappella* performances, on the other hand, the tuning frequency may slightly increase, for example, due to a rising involvement of the musicians during the concert. The maximum range of this deviation, however, can be assumed to be small in the case of professional musicians (about 3–5 cents).

Most pitch-based ACA make utilize the tuning frequency, however, many systems build on the assumption of a fixed tuning frequency at 440 Hz. While this assumption might work reasonably well in some cases, using the correct tuning frequency should generally improve detection accuracy [Ler06]. The estimation of the tuning frequency f_{A4} is thus a prerequisite for every mapping from frequency to musical pitch given in Eq. (7.11). Examples of tasks utilizing this mapping are key detection, chord recognition, and pitch tracking. For most non-pitch-related tasks, the tuning frequency is irrelevant, however, it has been identified as a potential confounding variable being learned from data [QL19].

⁵ According to the orchestra's archivists, March and April 2006.

⁶ The term *classical music* is in this context understood as “non-popular” music (that is generally not beat-based), as opposed to the epoch itself.

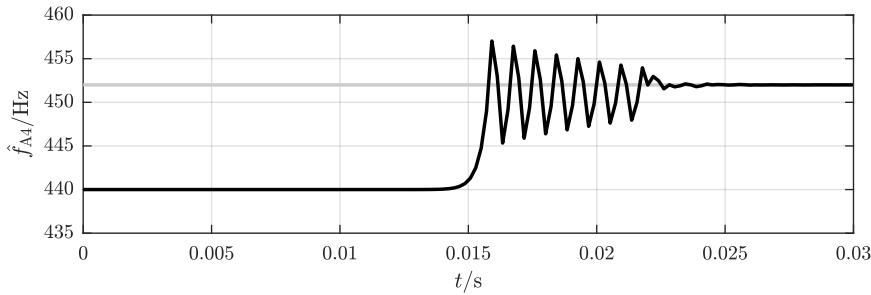


Fig. Gen.: plotfA4Rprop.m

Figure 7.30: Adaptation of the tuning frequency estimate from an initial setting of 440 Hz to the target frequency of 452 Hz with the RPROP algorithm.

7.4.1 Approaches to Tuning Frequency Estimation

A standard approach to *tuning frequency estimation* has the following processing steps, (i) a time-frequency transformation to extract the frequencies of the tonal frequency components with a high frequency resolution, (ii) the computation of a deviation of the extracted frequencies from an equal tempered pitch grid based on the assumed tuning frequency, and (iii) an optimization process minimizing the average deviation by adjusting the tuning frequency. To give a specific example, we present an early system for tuning frequency estimation. Lerch proposed using a bank of steep resonance filters for detecting the tuning frequency with adjustable mid frequencies [Ler04, Ler06]. Each (equal tempered) semitone in the range of two octaves is represented by a group of narrowly spaced filters, with each group centered around the semitone frequency which is based on the most recent tuning frequency estimate. Under the assumption that the filter output energies of the filters left of the semitone frequency should equal the output energies right to the semitone frequency in case of a correct tuning frequency, the symmetry of the sum of the output energies of all “left” filters compare to the sum of all “right” filters is evaluated as model error or objective function. If the energies are unequal, the tuning frequency, and thus all filter mid-frequencies, is adapted with the goal of symmetrizing the energy distribution in the future. The adaption rule used does not require a signal-dependent adaption step size [RB93]. Rather, the adaption rule for the adjustment of the estimated tuning frequency \hat{f}_{A4} of the next processing block $n + 1$ is

$$\hat{f}_{A4}(n + 1) = (1 + \text{sign}(E(2) - E(0)) \cdot \lambda) \cdot \hat{f}_{A4}(n) \quad (7.42)$$

with η being scaled up if the direction of the previous adaptation was the same and scaled down otherwise. The advantage of the scaling is an increasing step size as long as the sign does not change and a decreasing step size otherwise. Figure 7.30 shows the adaptation from the initial tuning frequency of 440 Hz to the real frequency of 452 Hz. Adaptation can be tuned either for speed or for accuracy. The advantage of constant adaption is (i) applicability to real-time systems, and (ii) capabilities to track a slowly changing tuning frequency over time.

The assumption of equal temperament is common to most approaches although not necessarily true; only a score-informed approach with knowledge of the used temperament would render this assumption unnecessary. One such score-based approach was proposed by Scheirer, who swept set of narrow band-pass filters over a small frequency range around handpicked frequencies match pitches from the previously analyzed score to identify the frequency maximizing the output energy [Sch95].

If an STFT is used for the frequency representation, it is most likely necessary to compute the instantaneous frequency of the detected peaks or some other form of spectral interpolation to ensure sufficient frequency accuracy [Dix96, DTB11]. Alternatively, a CQT is a good choice to yield a constant pitch resolution (as opposed to frequency resolution) [ZKG05, EM11].

The objective function for estimating the tuning frequency is commonly either based on energy per deviation. One way of doing so is to create a histogram-like representation of the deviations from the closest equal tempered pitch. The location of the maximum [ZKG05, Pee06] or the location of the “histogram mass center” [Ryy04] can then be chosen to derive the estimate tuning frequency. The deviation can also be formulated in circular statistics as proposed by Dressler and Streich [DS07].

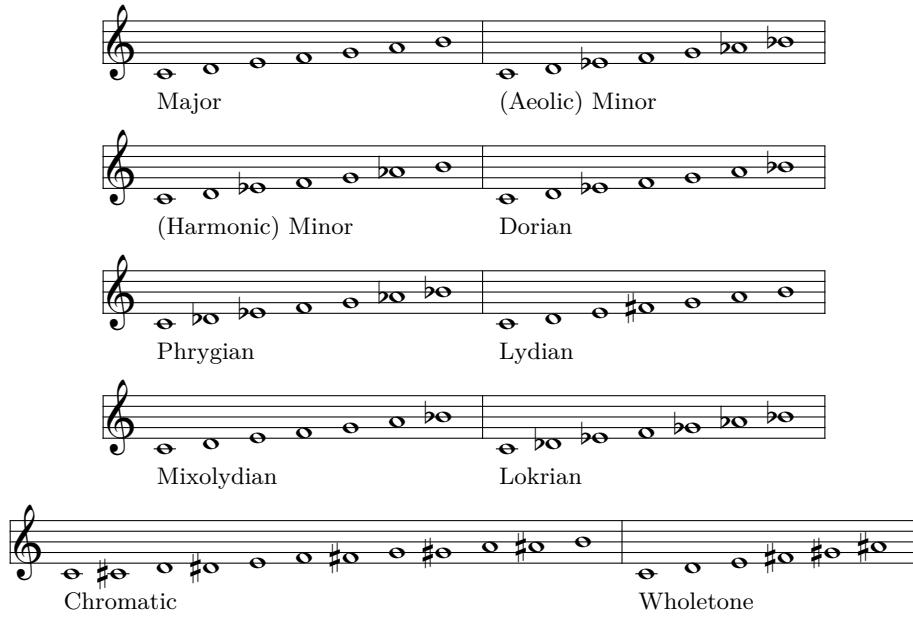


Figure 7.31: Different modes in musical score notation starting at the tonic C

7.4.2 Evaluation

The creation of a dataset with real-world audio data and reliable ground truth tuning frequency annotations is nearly impossible, and no such dataset exists [DDLM14]. To complicate matters further, even when tuning frequency data are known, such as in the case of some orchestras, there is no guarantee that the available recordings always match. For instance, historic recordings can suffer from tape speed mismatches but recording and reproduction, resulting in audio tuned slightly higher or slightly lower. Furthermore, the annotation of existing recordings with their tuning frequencies requires experts with good ears and is so tedious that it is hardly doable. The only possible solution to this dilemma is the use of synthesized audio as digital oscillators and/or sound libraries can be expected to be tuned to 440 Hz. This has the inherent disadvantage that the differences between synthesized audio and recorded audio might lead to overestimation of the reliability of tuning frequency algorithms in real-world scenarios.

As tuning frequency estimation is a regression task, deviation-based metrics can be used, although metrics for classification are also reported when assuming a tolerance. Note that the evaluation domain should be the unit Cent to account for (i) the non-linearity of frequency perception and (ii) the pitch-independence of tuning frequency.

7.5 Key Detection

The musical *key* of a tonal piece of music is defined by both its *mode* and a *tonic*. The *tonic* is the most important pitch class in a specific key. It is also referred to as the *first scale degree* and will usually appear most frequently in a piece of music. The *mode* defines a set of relative pitch relationships; an example would be: the distance between first and second scale degree is a major second, between first and third scale degree is a major third, etc. The most common modes are the *major mode* and the *minor mode*. Figure 7.31 displays an example set of different modes, all starting from the tonic C . All modes except major mode and the two minor modes — aeolic mode and harmonic mode — are only of interest in specific musical styles and are usually not considered to be important in the context of ACA. The key thus defines the set of pitch classes which are used to construct the tonal aspects of a piece of music. In popular music it is common for a piece to have exactly one key. There are many exceptions to this rule: the key can change within a piece (when a so-called *modulation* occurs) and non-key pitch classes may be used for musical reasons.

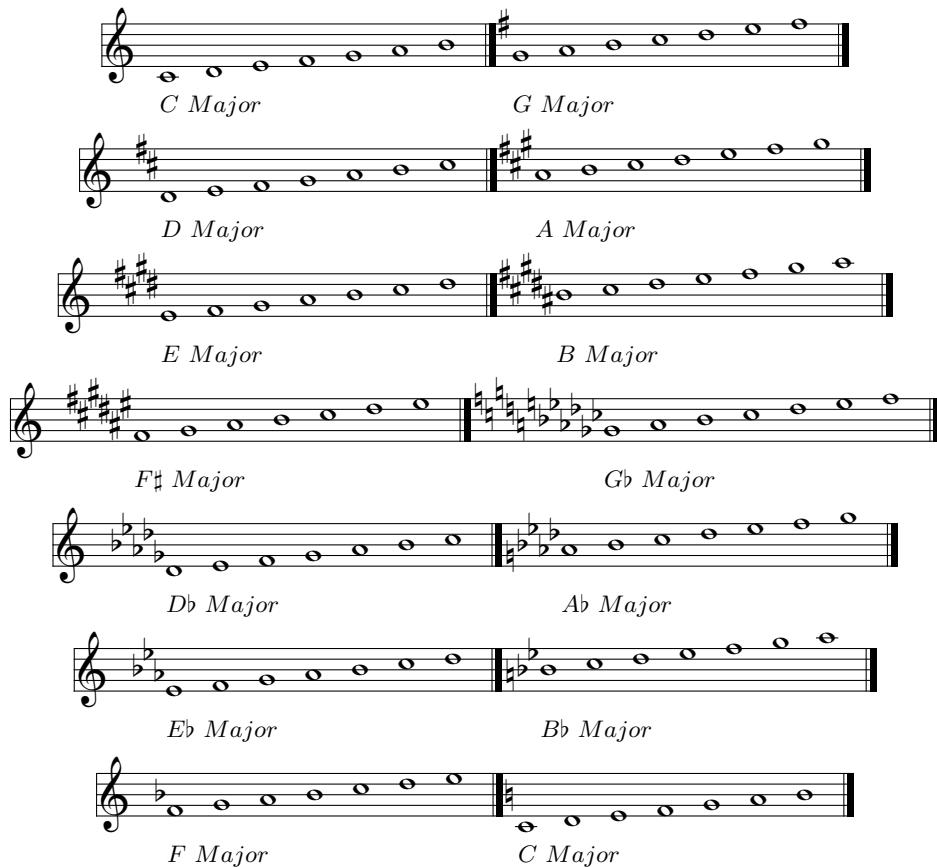


Figure 7.32: The twelve major keys in musical score notation, notated in the 4th octave

Depending on the tonic and mode, up to six different accidentals have to be used to raise or lower the pitches in the musical score. The notational convention allows writing all key-inherent accidentals at the begin of a staff — the *key signature* — and to add only accidentals within the score where non-key-inherent pitches are used.

Figure 7.32 shows major modes with their key signatures starting from all possible tonics. As can be seen from the score, the tonics of keys that differ only in one accidental are always spaced by a fifth ($F \leftrightarrow C$, $C \leftrightarrow G$, $G \leftrightarrow D$, etc.). This relationship can be visualized by the so-called *circle of fifths* (Fig. 7.33). Strictly speaking, this circle is only closed in the case of enharmonic equivalence when, e.g., $F\sharp$ equals $G\flat$ (see Sect. 7.2.3.1).

Neighboring keys on the circle of fifths have all pitch classes but one in common; for example, *G Major* has the same pitches as *C Major* except for the *F* which is raised to an *F \sharp* .

The circle of fifths also exists for the (aeolian) minor keys with *a minor* being the key without accidentals. Keys construed from the same set of pitch classes such as *C Major* and *a minor* are called relative keys; in the circle of fifths their distance is 0. In order to build an analytical model for key distances with a non-zero distance between relative keys, the circle of fifths can be enhanced to a three-dimensional model. This model would feature two parallel planes, one containing the circle for major keys and the other for minor keys. Parallel keys thus share the same (x, y) coordinates but have a different z coordinate (which is then the distance between the two parallel keys).

Since the circle of fifths shows the relation of different keys, it hints also at what modulations are more or less likely. To give an example, the most likely key changes from *F Major* would be either *C Major*, *B \flat Major*, or *d minor*. The circle of fifths can thus be understood as a model for a distance map between keys.

The musical key is an important tonal property of a piece of tonal music as it signifies the tonal center and restricts the pitch classes used within this tonal context. In classical music, the key is used as one descriptor used to identify a specific piece of music, complementing name, genre/instrumentation, and opus. In modern

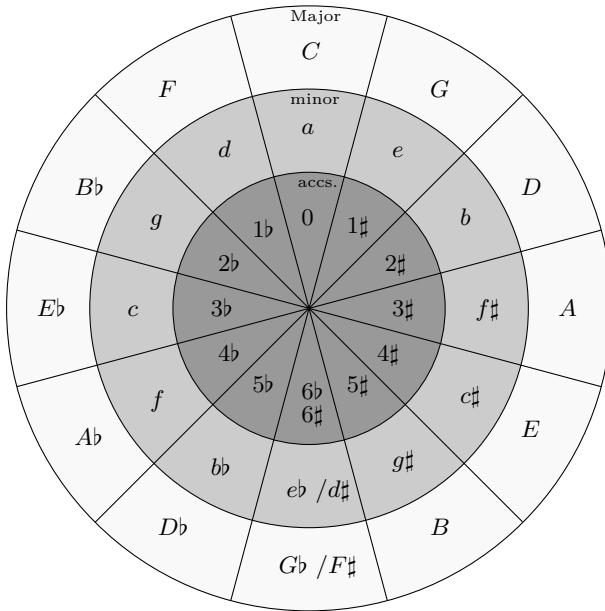


Figure 7.33: Circle of Fifths for both major keys and minor keys, plus the number of accidentals of the key signature per key.

software applications for DJs the key can be used to display the tonal compatibility between two tracks, i.e., to visualize how much “tonal overlap” they have to aid so-called harmonic mixing for the creation of so-called Mash-ups.

Since the key can be seen as a set of “allowed” pitch classes, the usual approach is to make use of an octave-independent representation of pitch for its automatic detection. The most common representation is the so-called pitch chroma, which is introduced in the following.

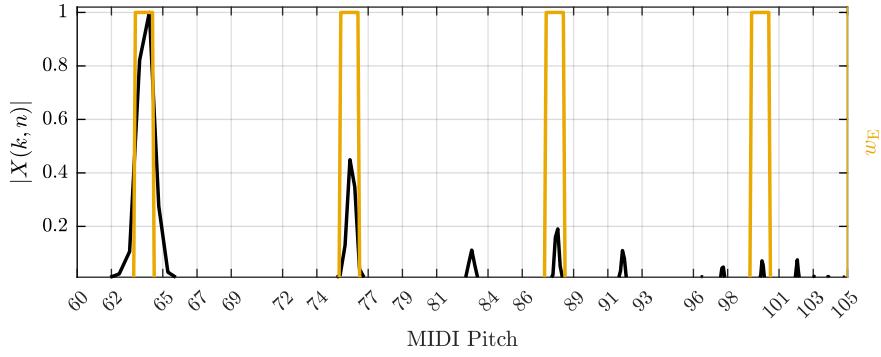
7.5.1 Pitch Chroma

The *pitch chroma* (sometimes also referred to as *pitch chromagram* or *pitch class profile*) is a histogram-like 12-dimensional vector with each dimension representing one pitch class ($C, C\sharp, D, \dots, B$; compare Sect. 7.2.1). It can be seen as a *pitch class distribution* for which the value of each dimension may represent both the number of occurrences of the specific pitch class in a time frame and its energy or velocity throughout the analysis block.

The pitch chroma is probably the most common instantaneous feature used to describe the tonal content of audio. Similar to most of the timbre descriptors introduced in Sect. 3.6, it is computed from a spectrum (often from a STFT). As mentioned above, it is a multi-dimensional feature (twelve pitch classes). There are several advantages of using pitch chroma-based analysis. It is less dependent on timbre fluctuations and noise than other features describing tonal content. The pitch chroma is also robust against loudness fluctuations (if normalized accordingly) as well as against octave errors—a typical problem of pitch detection algorithms—with the self-evident disadvantage that all octave information is lost. It is, for example, not possible to distinguish between a pair of chromas representation a note repetition and a pair representing an octave interval with the pitch chroma representation.

While a representation of pitch similar to the pitch chroma has been frequently used in the past (see, e.g., Krumhansl’s tonal distributions [Kru90] and compare the ‘perceptual’ introduction in Sect. 7.1.2), Fujishima might have been the first to propose its use in the context of audio signal processing [Fuj99]. The pitch chroma representation then quickly became popular [BW01, TEC02] and continues to be frequently found in ACA publications.

The exact algorithmic description of the pitch chroma computation varies from implementation to implementation; in all cases

Fig. Gen.: `plotPitchChromaGrouping.m`Figure 7.34: Mask function for pitch class E for pitch chroma computation (octaves 5–8).

- a frequency representation of the audio signal block is grouped into semi-tone bands,
- a measure of salience is computed in each band, and finally
- the sum of all bands (over all octaves) corresponding to a specific pitch class is calculated.

The simplest way to extract the pitch chroma sums the squared STFT magnitudes in each semi-tone band with the boundary indices k_l, k_u , and the result in every octave o is added to the corresponding pitch chroma entry with pitch class index j :

$$\nu(j, n) = \sum_{o=o_l}^{o_u} \left(\frac{1}{k_u(o, j) - k_l(o, j) + 1} \sum_{k=k_l(o, j)}^{k_u(o, j)} |X(k, n)|^2 \right), \quad (7.43)$$

$$\boldsymbol{\nu}(n) = [\nu(0, n), \nu(1, n), \nu(2, n), \dots, \nu(10, n), \nu(11, n)]^T. \quad (7.44)$$

The indices k_l, k_u are located at a distance of 50 cents from the mid-frequency of each equally tempered pitch.

Figure 7.34 shows the semi-tone bands for the pitch class E (typically pitch class index 4) in the octaves 5–8. Note that the window bandwidth is constant on the pitch axis shown here; on the linear frequency axis it would increase with increasing frequency.

Frequently, the pitch chroma is normed so that the sum of all possible pitch classes equals 1:

$$\boldsymbol{\nu}_N(n) = \boldsymbol{\nu}(n) \cdot \frac{1}{\sum_{j=0}^{11} \nu(j, n)}. \quad (7.45)$$

Alternatively, the pitch chroma is interpreted as a vector and is normed to a length of 1 instead:

$$\boldsymbol{\nu}_N(n) = \boldsymbol{\nu}(n) \cdot \sqrt{\frac{1}{\sum_{j=0}^{11} \nu(j, n)^2}}. \quad (7.46)$$

However, it should be kept in mind for normalization that the pitch chroma is neither a series of observations nor a position in a space with 12 unrelated dimensions; it is a *distribution*.

Figure 7.35 shows the magnitude spectrogram and the pitch chroma of a monophonic audio file. It allows the identification of individual pitch classes and thus an octave-independent reconstruction of the melody. The aggregated average pitch chroma on the bottom right shows which pitches are more or less dominant over the whole recording. The input audio is identical to Fig. 3.13.

The pitch chroma can be extracted from the spectrogram with a simple matrix multiplication; the matrix contains the mask function for each pitch and has a dimension of $12 \times \mathcal{K}$. Figure 7.36 shows the code for this matrix multiplication and the initialization of the mask matrix. Note the implicit tuning frequency assumption

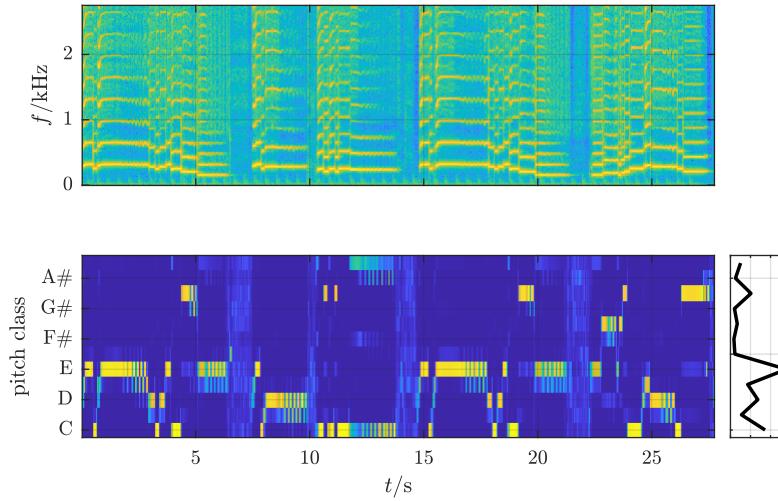


Fig. Gen.: plotPitchChroma.m

Figure 7.35: Magnitude spectrogram (top) and the pitch chroma (bottom) of a monophonic saxophone recording. The bottom right shows the aggregated average pitch chroma for the whole recording.

```
% generate filter matrix
H = GeneratePcFilters(size(X,1), f_s);

% compute pitch chroma
v_pc = H * X.^2;

% norm pitch chroma to a sum of 1
v_pc = v_pc ./ repmat(sum(v_pc,1), 12, 1);

% avoid NaN for silence frames
v_pc(:,sum(X,1) == 0) = 0;
end

%> generate the semi-tone filters (simple averaging)
function [H] = GeneratePcFilters(iSpecLength, f_s)

% initialization at C4
f_mid = 261.63;
iNumOctaves = 4;

%sanity check
while (f_mid*2^iNumOctaves > f_s/2)
    iNumOctaves = iNumOctaves - 1;
end

H = zeros(12, iSpecLength);

for (i = 1:12)
    afBounds = [2^(-1/24) 2^(1/24)] * f_mid * 2* (iSpecLength-1)/f_s;
    for (j = 1:iNumOctaves)
        iBounds = [ceil(2^(j-1)*afBounds(1)) floor(2^(j-1)*afBounds(2))] + 1;
        H(i,iBounds(1):iBounds(2)) = 1/(iBounds(2)+1-iBounds(1));
    end
    % increment to next semi-tone
    f_mid = f_mid*2^(1/12);
end

# generate filter matrix
H = generatePcFilters(X.shape[0], f_s)

# compute pitch chroma
v_pc = np.dot(H, X**2)

# norm pitch chroma to a sum of 1 but avoid
norm = v_pc.sum(axis=0, keepdims=True)
norm[norm == 0] = 1
v_pc = v_pc / norm

return np.squeeze(v_pc) if isSpectrum else v_pc
```

```
def generatePcFilters(iSpecLength, f_s):

# initialization at C4
f_mid = 261.63
iNumOctaves = 4
iNumPitchesPerOctave = 12

# sanity check
while (f_mid * 2**iNumOctaves > f_s / 2.):
    iNumOctaves = iNumOctaves - 1

H = np.zeros([iNumPitchesPerOctave, iSpecLength])

# for each pitch class i create weighting f
for i in range(0, iNumPitchesPerOctave):
    afBounds = np.array([2**(-1 / (2 * iNumOctaves)) * f_s])
    for j in range(0, iNumOctaves):
        iBounds = np.array([math.ceil(2**(j-1)*afBounds(1)) floor(2^(j-1)*afBounds(2))] + 1);
        H[i, iBounds[0]:iBounds[1]] = 1/(iBounds(2)+1-iBounds(1));

    # increment to next semi-tone
    f_mid = f_mid * 2**((1 / iNumPitchesPerOctave) * i)
```

(a) Matlab

Figure 7.36: Computation of the pitch chroma with a matrix multiplication.

in the code by setting $f_{C4} = 261.63 \text{ Hz}$; this means that this pitch chroma will be less usable with detuned audio inputs.

As with many other common features, implementations of the pitch chroma can significantly differ. This starts with different parametrizations of the basic pitch chroma mentioned above: different STFT block and hop lengths, a different number of octaves to integrate lead to different results, or using the magnitude spectrum instead of the power spectrum. Other implementation differences, however, can be more substantial. Most commonly found are the use of a different frequency transform, the selection of spectral content to aggregate, and the normalization.

With respect to frequency transforms, the CQT as introduced in Sect. 3.4.2 can also be used for pitch chroma computation [PBO00]. In this case the number of bands per semi-tone will equal 1 or be constant. A modification of the STFT has been used by Cremer and Derboven [CD04] for the computation of the pitch chroma: they utilize a so-called frequency-warped STFT as introduced by Oppenheim et al. which uses a chain of first-order all-pass filters to achieve non-equidistant spacing of the frequency bins [OJS71]. Alternative frequency transforms such as resonance filterbanks (see Fig. 3.18) have been proposed but did not find widespread adoption for pitch chroma extraction.

When using a STFT as frequency transform, different windows with triangular, trapezoid, or sinusoidal shapes can be applied to each semi-tone band to weight bins in the center of the band higher than bins at the boundaries, as opposed to the rectangular window implicitly used in Eq. (7.43) and plotted in Fig. 7.34. Frequency components at the boundaries of each band might be noise-like components or high harmonics and thus not necessarily of interest for the pitch chroma extraction. To focus the extraction on tonal components, the pitch chroma can be computed from a peak-picked or tonalness-weighted magnitude spectrum (compare, e.g., Sect. 3.6.11). Instead of using expert knowledge to optimize the pitch chroma manually chroma features can also be learned from data [KW16a].

The window length for aggregation of the pitch chroma is another parameter allowing variations between different implementations. While in the simplest case the STFT block length equals the texture window length [PP07, WD08], other algorithms combine a fixed number of short analysis blocks [PVM08] or adapt the texture window length to the period between two neighboring beats [BP05] or to a measure of harmonic change in the audio [HSG06]. The aggregation texture window length should obviously also be adjusted depending on the task at hand; key detection requires comparably long windows as opposed to chord recognition which usually requires window lengths between a beat and a bar length.

Various pitch chroma implementations have been compared in the context of chord recognition by Jiang et al. [JGKM11].

7.5.1.1 Pitch Chroma Properties

The wide-spread use of the pitch chroma makes a more detailed discussion of its properties and characteristics worthwhile. In most implementation, the pitch chroma is assumed to only show fundamental frequencies (and no overtones). In reality, all frequency content in the pre-defined frequency range of interest is mapped to the pitch chroma, regardless of it being a fundamental frequency or a higher harmonic. This leads to two possible problems:

- High non-power-of-two harmonics lead to distortions in the pitch chroma by adding undesired components. Figure 7.37 displays a series of 10 harmonics (amplitude weighting $1/k$) with the fundamental pitch $A3$ on top and the resulting pitch chroma at the bottom. The pitch chroma shows spurious components at E and $C\sharp$ and to a lesser degree at G and B . It is possible to partly compensate for this effect by using an amplitude model for the harmonics and modifying the pitch chroma accordingly [PBO00]. The effect can also be attenuated by applying a weighting function to de-emphasize higher frequencies (see e.g. [Pau04]).
- High harmonics may even be able to distort the pitch chroma in a different way: since the harmonics will deviate from the equal temperament — the temperament the pitch chroma computation is based on — they may map to pitches not really part of the tonal context. This effect is, however, of limited influence

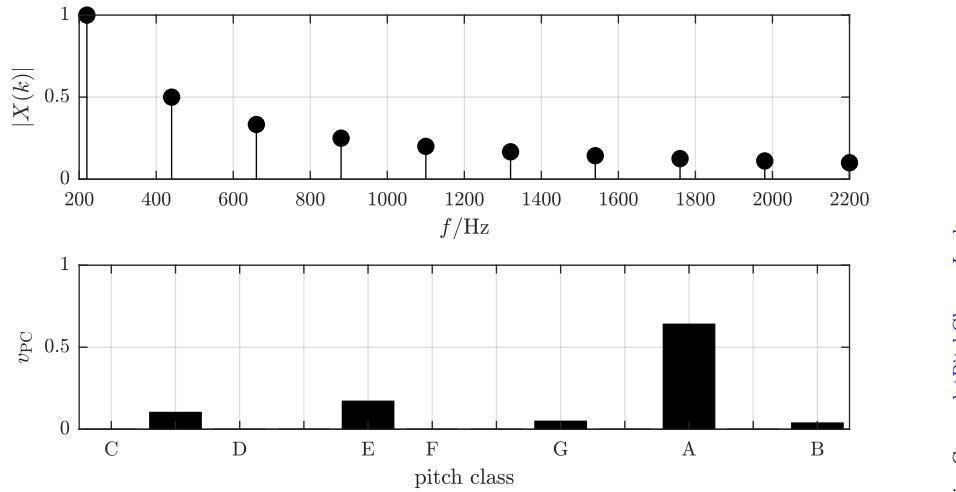


Fig. Gen.: plotPitchChromaLeakage.m

Figure 7.37: Theoretical pitch chroma of the pitch $A3$ with 10 harmonics.

Table 7.6: Deviation (in cents) of seven harmonics from the nearest equally tempered semi-tone mid-frequency

Harmonic	$ \Delta C(f, f_T) $
$f = f_0$	0
$f = 2 \cdot f_0$	0
$f = 3 \cdot f_0$	1.955
$f = 4 \cdot f_0$	0
$f = 5 \cdot f_0$	13.6863
$f = 6 \cdot f_0$	1.955
$f = 7 \cdot f_0$	31.1741
$\mu_{ \Delta C }$	6.9672

since the deviation of the lower harmonics is relatively small as shown in Table 7.6 and the magnitude of high harmonics is commonly low.

One possibility to avoid these artifacts is to use a multi-pitch detection system as a pre-processing step which ensures that only fundamental frequencies are mapped to the pitch chroma (compare [VPM08, PVM08, RK08]). The requirement of a reliable multi-pitch detection system, however, might counter-balance the advantages of the pitch chroma as a comparably robust and simple to extract low-level feature.

Several authors proposed to resort the pitch chroma entries to reflect (key) relationships between the pitch classes. One way to do so is to replace the standard modulo operation for computing the pitch chroma index as given in Eq. (7.13) by a shifted version:

$$\text{PC}_S(p) = \mod(7 \cdot p, 12). \quad (7.47)$$

This results in related keys with a distance of seven semi-tones (i.e., a Fifth) to be close to each other. Table 7.7 shows the pitch class order of the rearranged chroma. Certain similarity measures such as the Cross Correlation Function (CCF) will behave more gracefully or musically meaningful when using this resorted pitch chroma [TEC02, ZKG05]. Note, however, that the rearranged pitch chroma does not take into account the cyclic nature as indicated by the circle of fifths.

Table 7.7: Pitch class order in the original and the rearranged pitch chroma

PC	<i>C</i>	<i>C</i> \sharp	<i>D</i>	<i>D</i> \sharp	<i>E</i>	<i>F</i>	<i>F</i> \sharp	<i>G</i>	<i>G</i> \sharp	<i>A</i>	<i>A</i> \sharp	<i>B</i>
PC _S	<i>C</i>	<i>G</i>	<i>D</i>	<i>A</i>	<i>E</i>	<i>B</i>	<i>F</i> \sharp	<i>C</i> \sharp	<i>G</i> \sharp	<i>D</i> \sharp	<i>A</i> \sharp	<i>F</i>

7.5.1.2 Features Derived from the Pitch Chroma

Similar to extracting features from a spectrum or a distribution, features can be extracted from the 12-dimensional pitch chroma as well. There is no established set of features to be extracted from the pitch chroma, although the set of three features presented by Tzanetakis et al. proved to be simple yet effective [TEC02]. They use both the index and the amplitude of the pitch chroma maximum as well as interval between its two highest bins. For the latter feature, the pitch chroma is rearranged in order to place related keys as neighbors (see above).

There exists a multitude of other features to extract from a pitch chroma that might be useful in certain applications. Possibilities include a pitch chroma crest factor or the centroid of the resorted pitch chroma. The computation of statistical features such as the standard deviation is only of limited use due to the small number of elements.

7.5.2 Approaches to Key Detection

Key detection approaches are commonly based on the two assumptions that (i) the occurrence of key inherent pitch classes is more likely than the occurrence of non-key pitch classes and that (ii) the number and energy of occurrences is an indication of the key. When detecting the key of pieces with modulations, i.e., of pieces with a changing key, analyzing only small sections at the start and end of the piece of music will improve results as it is more than common that the piece will start and end in the same (main) key [Ler04, Pau04, Izm05].

Following the basic structure of an ACA system (compare Fig. 2.1), automatic key detection systems are usually based on two steps:

1. the extraction of some kind of pitch class distribution such as the average pitch chroma, and
2. the estimation of the most likely key given this distribution.

Most approaches attempt to identify 2 modes with 12 tonics each, leading to 12 *Major* and 12 *minor* keys.

Similar to many other tasks, modern key detection approaches tend to be end-to-end systems based on DNNs [KW17, SM19], however, traditional approaches based on pattern recognition are still common. A prototypical approach of detecting the musical key is introduced in the following. The first step, extracting the pitch chroma has already been discussed in Sect. 7.5.1. The time frame over which the pitch chroma is averaged can vary from the whole file to a shorter texture window length in case of local key changes. It is also possible to extract the pitch chroma only at the beginning and end of the audio file because the key is usually less ambivalent in these regions [Ler04]. The second step estimates the most likely key given a *template* pitch chroma, in the following referred to as key profile by finding the minimum distance between the extracted pitch chroma and all key profiles.

7.5.2.1 Key Profiles

The *key profile* is a pitch class distribution used as reference for each key. The assumption is that the key profiles of identical modes but different tonics are identical but (circularly) shifted.

The key profile templates can be constructed in various ways. The most trivial approach is the *Orthogonal* template ν_o . The orthogonal template assumes that the tonic is the most salient component of the pitch chroma. The two main disadvantages are that this template allows no distinction between major mode and minor mode and that all other keys have the same distance (and are thus considered unrelated).

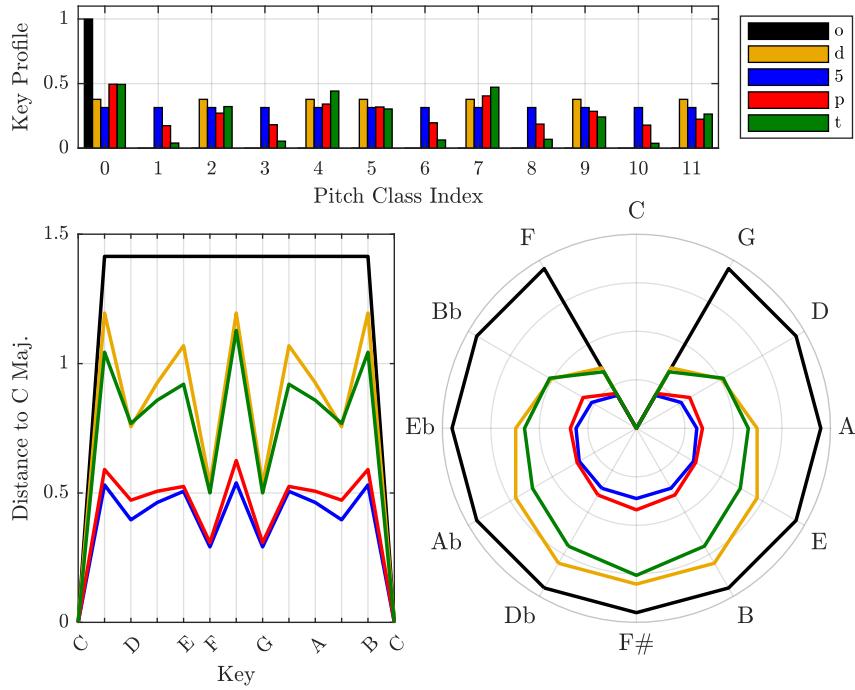
Fig. Gen.: `plotKeyProfiles.m`

Figure 7.38: Key profile vectors (top left) and the resulting inter-key distances (Major) to *C Major* (bottom left and right).

The latter problem can be solved by using *Diatonic* template ν_d , which sets all key-inherent pitch classes to non-zero values. This template is musically sound as key distances increase linearly like an “unwrapped” circle of fifths, however, it still does not allow differentiation between major mode and minor mode.

The distance of two keys can also be modeled by a geometric model of the circle of fifths with radius r : the *Circle of Fifths* template ν_5 . The coordinates of each key are given by the angle of the key in the circle of fifths. Each key has the same distance r to the point of origin. The model can be expanded into a third dimension to also include a distance between major mode and minor mode.

The key profile can also be based on tonal perception like the frequently-used *Probe Tone Ratings* ν_p . Although Krumhansl’s probe tone ratings are not directly a key description but the result of a listening test evaluating how a pitch fits into a given tonal context [Kru90], Krumhansl showed in experiments that these probe tone ratings correlate well with the number of occurrences of the pitch classes in real pieces of music.

Finally, a data-driven approach leads to *Extracted Key Profiles* ν_t . Temperley has proposed this common key profile extracted from symbolic data [Tem07]. Note that average key profiles can also be extracted from audio and might be genre-specific [OL15].

The introduced key profiles are shown in both Table 7.8 and in the top plot of Fig. 7.38. Note that the Table and the Figure differ in the normalization used: the figure normalizes to a vector length of 1 with Eq. (7.46) while the table normalizes to a sum of 1.

7.5.2.2 Similarity Measure between Template and Extracted Vector

The most likely key can be estimated by finding the minimum distance between the extracted pitch chroma ν_e and the key profile template. Under the reasonable assumption that the template key profiles for different keys are identical but shifted versions of the *C Major* profile, only 1 template needs to be stored per mode. Detecting modes other than major mode and minor mode is often considered not to be of relevance in practice; thus, only two template vectors have to be stored to generate an overall set of 24 shifted templates.

The index m of the most likely key can then be computed by finding the minimum distance d between the

Table 7.8: Various key profile templates, normalized to a vector length of 1 truncate decimals to two

	ν_o	ν_d	ν_5	ν_p	ν_t
$\nu(0)$	1	0.37796	$r \cdot e^{j2\pi \frac{0}{12}}$	0.49483	0.49355
$\nu(1)$	0	0	$r \cdot e^{j2\pi \frac{-5}{12}}$	0.17377	0.039589
$\nu(2)$	0	0.37796	$r \cdot e^{j2\pi \frac{2}{12}}$	0.27118	0.32199
$\nu(3)$	0	0	$r \cdot e^{j2\pi \frac{-3}{12}}$	0.18157	0.054105
$\nu(4)$	0	0.37796	$r \cdot e^{j2\pi \frac{4}{12}}$	0.34131	0.44208
$\nu(5)$	0	0.37796	$r \cdot e^{j2\pi \frac{-1}{12}}$	0.31872	0.30352
$\nu(6)$	0	0	$r \cdot e^{j2\pi \frac{6}{12}}$	0.19637	0.063343
$\nu(7)$	0	0.37796	$r \cdot e^{j2\pi \frac{1}{12}}$	0.40443	0.47177
$\nu(8)$	0	0	$r \cdot e^{j2\pi \frac{-4}{12}}$	0.18624	0.068621
$\nu(9)$	0	0.37796	$r \cdot e^{j2\pi \frac{3}{12}}$	0.28521	0.24149
$\nu(10)$	0	0	$r \cdot e^{j2\pi \frac{-2}{12}}$	0.17845	0.03761
$\nu(11)$	0	0.37796	$r \cdot e^{j2\pi \frac{5}{12}}$	0.22443	0.26393

extracted pitch chroma ν_e and the set of 24 template key profiles ν_t :

$$m = \min_{0 \leq s < 24} (d(s)). \quad (7.48)$$

Typical distance measures can be the Euclidean distance or the Cosine distance (compare Sect. 13). In the latter case, the identification of the key can also be expressed as a circular CCF.

The distance between different keys not only depend on the distance measure itself but also on the predefined key profile. Figure 7.38 plots the inter-key distances between the shifted key profile templates given above with respect to *C Major*; the distances are shown both in a line plot (bottom left) and within the circle of fifths (bottom right). The Euclidean distance has been used to compute these diagrams; therefore the input key profiles have been normed to a vector length of 1 (as opposed to a sum of 1). The distances generally behave as anticipated with the distance to *C Major* being the minimum. While the orthogonal template has a high distance to other keys, it does not result in smaller distances for related keys. All remaining profiles basically show the tendency of greater distances for keys more distant according to the circle of fifths. This is demonstrated by the cardioid shape of these distances in the polar diagram. It should be pointed out that the graph only visualizes the distances for keys in major mode and that none of the templates but the probe tone ratings and the extracted key profiles are able to separate between major mode and minor mode keys (although the model based on the circle of fifths can be adapted to a 3D model).

Theoretically, the use of these vector-based inter-key distance measures is problematic as the pitch chroma is a distribution and no vector. However, vector distances have been used and appear to work reasonably well in the absence of other fitting distance measures.

It is possible to smooth the pitch chroma to avoid maximum distances for nearby pitches [MSLS11] which in the case of tonality perception is probably most effective if the pitch chroma is reordered in a way that related keys have similar pitch chroma indices as described in Sect. 7.5.1.1.

There are other more complex mathematical models for estimating a distance between two keys or pitch class distributions; one example is Chew's *spiral array model* [Che98, CC05]. Another model targeted specifically at automatic tonalness detection is the multi-dimensional *tonal centroid* proposed by Harte et al. [HSG06] for which the pitch chroma is converted into a six-dimensional representation called tonal centroid based on the so-called harmonic network or *Tonnetz*. If enharmonic equivalence can be assumed, the Tonnetz can be transformed into three two-dimensional planes representing the circle of fifths, major thirds, and minor thirds. Another model is Gatzsche's *circular pitch space* [GMS08].

7.5.3 Evaluation

As the musical key is a common annotation in Western music, ground truth data can be more easily accessible than for other tasks. Western classical music, for example, often uses the key as one identifier next to the name. The caveat in this case, however, is that the key might change frequently throughout the piece depending on form, genre, and epoch. These modulations break the assumption of one, static key per piece of music that many approaches are based on. For popular music, this assumption is much more likely to be correct, and there exist internet databases for looking up the keys of various songs.

7.5.3.1 Metrics

As key detection is a classification task, normal classification-based metrics can be used (see Sect. 6.2.1). The most commonly used metric is the accuracy. Occasionally, a weighted accuracy score is used that aims to account for the severeness of errors by treating the erroneous classification into related keys as a partly correct result. The proposed weightings are: 1.0 if the key is correctly detected, 0.5 if the estimated key is one Fifth above the ground truth key, 0.3 if the estimated key is a relative key (sharing the same pitch classes) of the ground truth key, and 0.2 if the estimated key is a parallel key (sharing the same tonic) of the ground truth key. The actual reasons or heuristics behind these weightings have been lost in history, but they do not necessarily match the errors that typical key detection systems produce (compare, e.g., [Ler04, KW17]).

7.5.3.2 Datasets

Despite the availability of ground truth annotations, the number of complete audio key detection datasets remains limited. As the task of musical key detection requires the whole or a significant segment of the song, researchers cannot easily release commercial real-world data without violating licensing restrictions. For example, most of the key data available on Isophonics⁷ does not come with associate audio data, only track information. Focusing on electronic dance music, the GiantSteps dataset [KFH⁺15] provides key labels. There are also datasets for other tasks that have been annotated with musical keys by other researchers, many of such are listed online on the accompanying website.⁸

7.5.3.3 Results

As with other tasks the results vary largely between different datasets. A state of the art system seems to be able to yield an accuracy in the general range of 70 – 90%.⁹

7.6 Chord Recognition

The simultaneous use of (usually no less than three) different pitches creates a chord. Chords built of three pitches are referred to as triads. The most common chord types are built of Third intervals — a few examples are displayed in Fig. 7.39 with respect to a root note of *C*. Note that the same naming convention is used to label both the key and some common chords (e.g., *C Major*), so one has to derive from the context which of the two is meant.

Chord-inherent pitches can be doubled in different octaves without changing the chord type. Most frequently, the chord's root note is doubled. If the lowest note of a sounding chord is not its root note, this chord is called *inverted*. The first chord inversion of a *C Major* triad would therefore have the pitch classes (bottom-up) *E – G – C*. Figure 7.40 shows the two possible inversions of a *D Major* triad. The second inversion usually appears less frequent than the first.

Each chord can have one or more musical (harmonic) functions depending on the musical key as well as dependent on other musical context. The chord sharing the root note with the current key is referred to as

⁷<http://isophonics.net/datasets>, last access date Aug 6, 2021

⁸<https://www.AudioContentAnalysis.org/data-sets>, last access date Aug 6, 2021

⁹https://www.music-ir.org/mirex/wiki/2019:Audio_Key_Detection_Results, last access date Aug 6, 2021

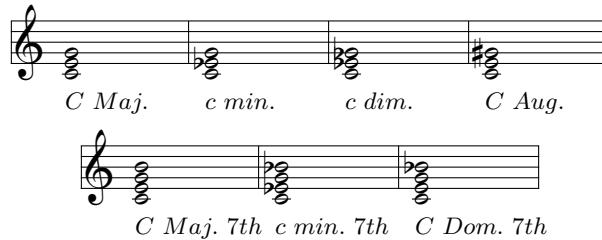


Figure 7.39: Common chords in musical score notation on a root note of *C*



Figure 7.40: The two inversions of a *D Major* triad in musical score notation

the *Tonic chord* and most commonly represents the tonal center. Other important harmonic functions are the so-called *dominant chord* with the key's fifth scale degree as the root note and the *subdominant chord* with the key's fourth scale degree as the root note. Dominant chords often induce an expectancy of a following tonic chord in the listener.

7.6.1 Approaches to Chord Recognition

Detecting the sequence of chords from a musical audio signal can be challenging, because the signal will not only contain the pitched chord content, but also noisy content, e.g., from drums, and non-chord tones in other voices. Furthermore, a chord might be present in music without the pitches actually sounding simultaneously, for example, in the case of arpeggiated chords. This makes the recognition of chords a non-trivial problem.

The general structure of a typical *chord recognition* system is visualized in Fig. 7.41. Pauwels et al. note in a recent survey article [POGS19] that this general structure goes back to the method and discussion in Fujishima's seminal paper on chord recognition [Fuj99]. Cho and Bello investigate the impact of variations within the different stages on the overall chord recognition accuracy [CB14]. As with nearly all ACA tasks, many newer approaches to this task cannot easily be structured in this way as they follow an end-to-end approach with DNNs [HB12, BLBV13, ZL15, KW16b, NAG19].

Similar to key detection systems, the automatic recognition of chords most frequently utilizes the pitch chroma (see Sect. 7.5.1). Unlike in key detection systems, the pitch chroma is extracted from short segments in the piece of music as the focus is on the local tonal context as opposed to the global context. Some researchers

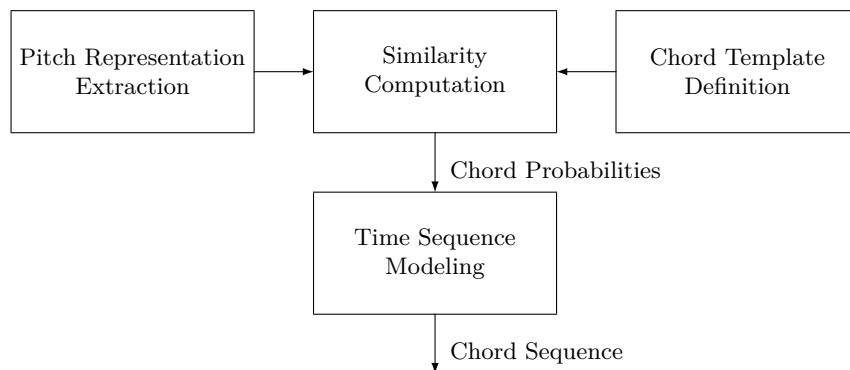


Figure 7.41: Flowchart of a simple chord detection system

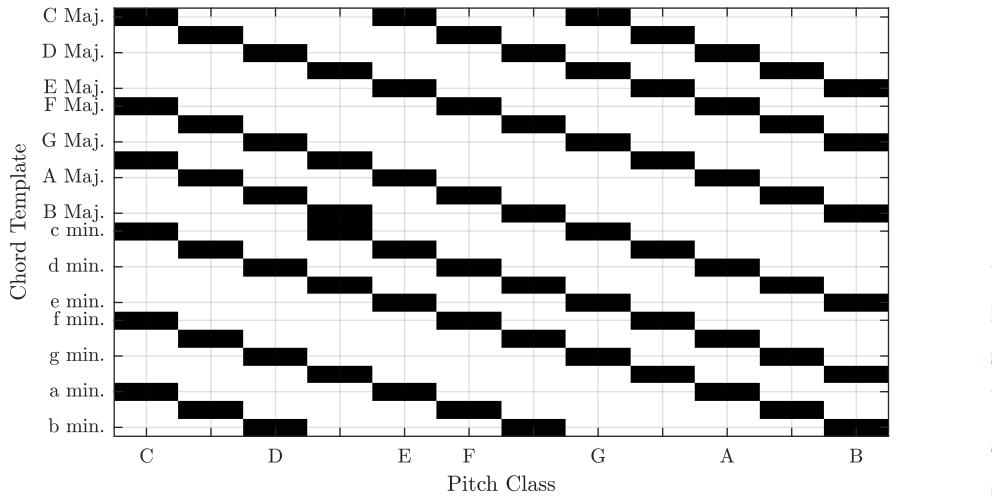


Figure 7.42: Simple chord template matrix for chord estimation based on pitch chroma.

propose to extract one pitch chroma for the period between each pair of neighboring beats [BP05, SW05]. Although chords can obviously change anytime, changes tend more likely on the beat position, and the long aggregation window seems to increase the robustness of the pitch representation.

The simplest chord template is a binary template; all templates can be grouped in a matrix Γ with the dimensions $T \times 12$ with T being the number of chord templates. Each row of the matrix represents one chord template; the simplest template would be to weight each pitch which is part of the chord by 1 and the remaining pitches with 0. Each row could be normalized to the sum of all entries in this row in order to compute the arithmetic mean. In this case, the template for a *C Major* triad would be $[1/3, 0, 0, 0, 1/3, 0, 0, 1/3, 0, 0, 0]$. This chord template matrix is shown in Fig. 7.42. The number of chord templates will be $T = 24$ if only all major and minor triads are allowed. Other proposed methods for constructing chord templates are, for instance, based on a Tonnetz representation [HSG06] or modeling the templates from data [SE03].

The likelihood of each chord can then be estimated by computing a distance or similarity between the instantaneous observed pitch chroma and the chord template. One way of doing so is with a linear transformation by the chord template matrix Γ :

$$\mathbf{p}_E(n) = \Gamma \cdot \mathbf{v}_{pc}(n) \quad (7.49)$$

This operation can also be interpreted as computation of the correlation coefficient between pitch chroma and each template (each row of Γ). This is similar to the key detection approach mentioned in Sect. 7.5 in the case of the cosine distance used as distance measure. The resulting vector per block $\mathbf{p}_E(n)$ then has the dimension $T \times 1$ and is a measure of the salience or likelihood of a specific chord given the pitch chroma $\mathbf{v}_{pc}(n)$.

These estimated likelihoods tend to be somewhat noisy and unreliable, especially in the case of short block sizes for pitch chroma extraction. There are various reasons for this, some of them technical (pitch chroma is only an imperfect representation of the tonal content), and some of them are musical (played or sung non-chord pitches, arpeggiations and incomplete chords, etc.). To alleviate this problem the results can be smoothed with a low pass (or median) filter applied to each chord likelihood [OFG11]. Much more common, however, has been the use of the Viterbi algorithm which not only has the ability to smooth the results but also to model transitions between neighboring chords. Thus, modeling the chord progression with a HMM has been repeatedly proven to increase stability of the results. The Viterbi algorithm as applied to chord recognition is introduced in detail in the Sect. 7.6.2. This post-processing step can be modeled in neural networks by using, e.g., recurrent architectures [BLBV13].

Figure 7.43 shows the implementation of a simple block-wise chord estimation utilizing this template, which returns two results, one without post-processing and one applying the Viterbi algorithm to the output.

```
% extract pitch chroma
[v_pc, t] = ComputeFeature ('SpectralPitchChroma',...
    afAudioData, ...
    f_s, ...
    [], ...
    iBlockLength, ...
    iHopLength); ...

% estimate chord probabilities
P_E = T * v_pc;
P_E = P_E ./ sum(P_E,1);

% assign series of labels/indices starting with 0
aiChordIdx = zeros(2,length(t));
[~, aiChordIdx(1,:)] = max(P_E,[],1);

% transition probabilities
[P_T] = getChordTransProb ();

% compute path with Viterbi algorithm
[aiChordIdx(2,:), -] = ToolViterbi(P_E, ...
    P_T, ...
    ones(24,1)/24, ...
    true);
```

(a) Matlab

```
% extract pitch chroma
[v_pc, t] = ComputeFeature ('SpectralPitchChroma',...
    afAudioData, ...
    f_s, ...
    [], ...
    iBlockLength, ...
    iHopLength); ...

% estimate chord probabilities
P_E = T * v_pc;
P_E = P_E ./ sum(P_E,1);

% assign series of labels/indices starting with 0
aiChordIdx = zeros(2,length(t));
[~, aiChordIdx(1,:)] = max(P_E,[],1);

% transition probabilities
[P_T] = getChordTransProb ();

% compute path with Viterbi algorithm
[aiChordIdx(2,:), -] = ToolViterbi(P_E, ...
    P_T, ...
    ones(24,1)/24, ...
    true);
```

(b) Python

Figure 7.43: Implementation of a simple chord detection approach with a chord template matrix.
implement python

7.6.2 Viterbi Algorithm

The *Viterbi algorithm* provides a path-finding solution that identifies the globally most likely sequence of pre-defined states [Vit67], given observations. The algorithm can be used for a wide variety of tasks; here, we will focus on its use for chord recognition, but the principle can be easily transferred to other applications. The Viterbi algorithm is often considered an integral part of an HMM. It uses the following variables (the dimensionality given is based on the previous example — 12 pitch chroma dimensions with \mathcal{N} observations and 24 chord templates):

- *States*: The sequence of states is being estimated and thus unknown (or *hidden*). In the case of chord recognition, each state will be a chord to be estimated (i.e., 24 states in the example above).
- *Observations*: The observations are the extracted features. In our example, each pitch chroma will be one observation, resulting in an observation matrix \mathbf{V}_{pc} with dimensions $12 \times \mathcal{N}$.
- *Emission probabilities*: The emission probabilities model the relationship between observations and states, more specifically, the probability of an observation given a specific state. It is the likelihood of a measured pitch chroma, given a specific chord. The emission probabilities can be captured in an emission probability matrix \mathbf{P}_E with dimensions $24 \times \mathcal{N}$.
- *Start probabilities*: Different states might be more likely to occur than others as initial state. For example, in a key dependent chord recognition model of popular music, the tonic chord is more likely to appear at the song start than other chords. The start probabilities will be a vector \mathbf{p}_S with dimensionality 24,
- *Transition probabilities*: The fact that transitions between certain states are more likely than between others is modeled with the transition probabilities. For example, a transition from *A Major* to *D Major* can be expected to have a higher probability than a transition from *A Major* to *D♯ Major*. The transition probabilities for each pair of chords can be captured in a transition probability matrix \mathbf{P}_T with dimensions 24×24 .

For an implementation, we furthermore need an empty accumulated probability matrix \mathbf{P}_{res} storing the highest likelihood to reach a state, and an empty backtracking matrix \mathbf{I} storing the most likely preceding state for each state with dimensions $24 \times \mathcal{N}$.

The model for the probabilities at the core of the HMM is usually derived from training data, although that is not always the case. These data can either be annotated audio [BP05], MIDI-synthesized audio [LS06],

```
% initialization
I = zeros(size(P_E));
P_res = zeros(size(P_E));
P_res(:,1) = P_E(:,1).*p_s;

% recursion
for (n = 2:size(P_E,2))
    for (s = 1:size(P_E,1))
        % find max of preceding times trans prob
        [p_max,I(s,n)] = max(P_res(:,n-1).*P_T(:,s));
        P_res(s,n) = P_E(s,n) * p_max;
    end
end

% traceback
p = zeros(1,size(P_E,2));
% start with the last element, then count down
[p,prob] = max(P_res(:,end));
for n = size(P_E,2)-1:-1:1
    p(n) = I(p(n+1),n+1);
end
(a) Matlab
```

```
% initialization
I = zeros(size(P_E));
P_res = zeros(size(P_E));
P_res(:,1) = P_E(:,1).*p_s;

% recursion
for (n = 2:size(P_E,2))
    for (s = 1:size(P_E,1))
        % find max of preceding times trans prob
        [p_max,I(s,n)] = max(P_res(:,n-1).*P_T(:,s));
        P_res(s,n) = P_E(s,n) * p_max;
    end
end

% traceback
p = zeros(1,size(P_E,2));
% start with the last element, then count down
[p,prob] = max(P_res(:,end));
for n = size(P_E,2)-1:-1:1
    p(n) = I(p(n+1),n+1);
end
(b) Python
```

Figure 7.44: Implementation of the Viterbi algorithm.[implement python](#)

or symbolic data [PP07]. Without data, musicologically driven models can be used: A very simple (but key independent) model for the chord progression likelihood is to use a circle of fifths [BP05]. A related approach is to utilize the correlation between Krumhansl's key profiles as chord distances [PP07]. Note that in these cases, we model the chord relations through key relations, which is systematically questionable. The preference for specific keys and root notes in specific genres often makes training data augmentation through pitch shifting (Sect. 5.2) desirable.

The Viterbi algorithm identifies the most likely state sequence from left to right by estimating the probability of each state at each time instance given all previous state probabilities using dynamic programming principles. This avoids the calculation of every possible path through the state sequence. We execute the following steps:

1. Initialize transition and start probabilities through model (either algorithmic or data-driven).
2. Compute observations, i.e., the pitch chroma.
3. Compute emission probabilities by, e.g., applying Eq. (7.49) $\mathbf{P}_E = \boldsymbol{\Gamma} \cdot \mathbf{V}_{pc}$.
4. Initialize the first column of state probabilities given the start probabilities

$$P_{\text{res}}(s, 0) = p_S(s) \odot P_E(s, 0) \quad (7.50)$$

with \odot being element-wise multiplication (Hadamard product).

5. Now start iterating from the second column over all columns n :

$$P_{\text{res}}(s, n) = P_E(s, n) \cdot \max_{\forall j} (P_E(j, n-1) \cdot P_T(j, s)) \quad (7.51)$$

$$I(s, n) = \operatorname{argmax}_{\forall j} (P_E(j, n-1) \cdot P_T(j, s)). \quad (7.52)$$

6. The best final state is then $s_{\text{final}} = \operatorname{argmax}_{\forall s} (P_{\text{res}}(s, N-1))$.
7. Finally, the path through the emission probability matrix with the highest probability can be backtracked from by walking back to the left using the indices stored in I starting from $I(s_{\text{final}}, N-1)$.

For long sequences the probabilities P_{res} will approach zero. The most common solution is the use the logarithmic likelihood. Alternatively, each column could also be normalized to a sum of 1 during the iteration if the overall sequence probability is not of interest.

Figure 7.44 shows an implementation of this algorithm.

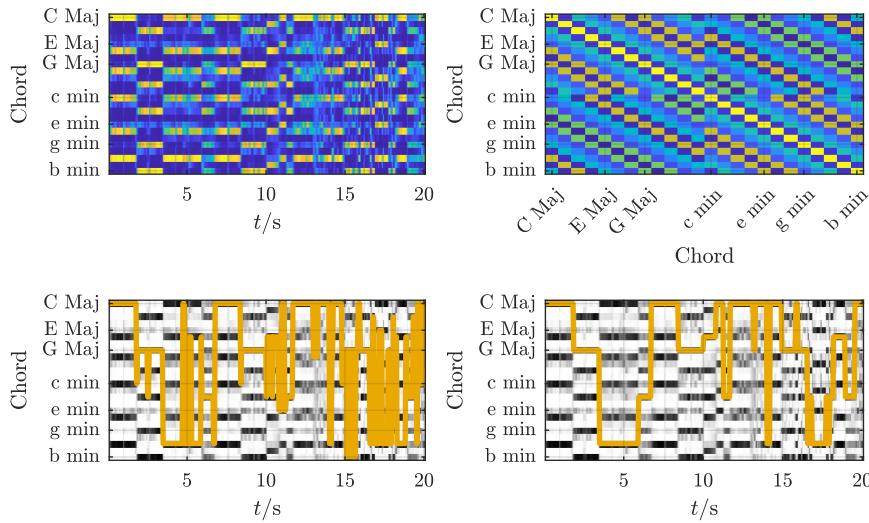
Fig. Gen.: [plotChordDetection.m](#)

Figure 7.45: Chord emission probability matrix (top left), transition matrix (top right), instantaneous chord detection results by taking the maximum emission (bottom left) and output of the Viterbi algorithm (bottom right).

Figure 7.45 visualizes a chord detection example. In this example, the emission probabilities are computed through the template matrix above, and the transmission probabilities are approximated through key distances with the circle of fifths. The figure shows the first 20 s of a pop song; the emission probabilities are shown on the top left, and the resulting chord sequence (*without* Viterbi post-processing) is shown in the bottom left; the top right shows the used chord transition matrix, and the bottom right the output of the Viterbi algorithm. Comparing the two paths, we see that one effect of applying Viterbi (with a high self transition probability) smooths the path considerably. The resulting chords on the bottom right equal the ground truth for the first ten seconds. The detection gets somewhat erroneous when the vocals start after that.

Note that the Viterbi algorithm follows the same dynamic programming principles as dynamic time warping (see Sect. 10.1); in some ways, it can be seen as a generalization of Dynamic Time Warping (DTW). In both cases the goal is to find the path through a matrix which maximizes the probability or minimizes the cost. The main differences between DTW and the Viterbi algorithm are (i) that DTW has additional constraints on the path, which has to start in one corner, end in the opposite corner, and move only to neighboring elements, constraints that the Viterbi algorithm models more generally through start and transition probabilities, (ii) that DTW’s cost matrix can be seen as the inverse of the accumulated probability matrix, (iii) that neither the transition cost of DTW nor the construction of its distance matrix are commonly driven by training data (compare emission and transition probabilities), and (iv) that the DTW path length is not pre-defined.

7.6.3 Evaluation

The objective for the evaluation of chord recognition systems is the comparison of the estimated labels with the ground truth annotations. This task is complicated by several challenges. First, the annotation of chord labels can be error-prone if human subjects identify the chords through listening. Second, chord labels can be subjective to a certain degree; for example, whether to understand the melody pitch as part of a chord or not might be a matter of opinion. Third, chords and chord types can be musically ambiguous without context, especially in the enharmonic settings commonly applied in ACA; depending on musical context, the same pitch chroma representation might represent different chords with different root notes. This can lead to considerable disagreements between annotators [KHB⁺19]. Fourth, not all types/segments of music contain chords (e.g., monophonic parts, fugues), complicating the labeling and recognition.

Given these challenges and subjectivities, the annotations in general cannot be considered absolute truth,

complicating the systematic evaluation [PP13].

Other complications arise from the definition of the chord vocabulary used and potential mismatches between the ground truth chord vocabulary and the chords detected by the system. If our system detects only the 24 major and minor triads mentioned above, how will we deal with ground truth annotations of a seventh chord? Can we reduce the seventh chord to a triad or should we remove this annotation from the evaluation? Furthermore, the more detailed and complicated the chord annotations become, the more pronounced the imbalance between the various chords in the data will become. Burgoyne et al. report that just the seven most popular chords (*C Major, D Major, G Major, A Major, E Major, F Major, B♭ Major*) cover more than 30% of the observations and that the five most popular chord types cover more than 80% of the data [BWF11].

7.6.3.1 Metrics

Similar to other tasks such as pitch detection, chords will be annotated with start and end times as opposed to a chord label for a short time segment, the output format of a conventional chord detection system. Frequently, the ground truth is processed and quantized into equivalent time segments to allow for computation of classification metrics (see Sect. 6.2.1). The class imbalance requires the use of macro-measures to account for less prominent chord classes.

The classification metrics do not necessarily have to be computed on the chord classes; it has been suggested to evaluate broader categories of the ground truth such as root note, chord type, inversion, etc. individually [RMH⁺14]. A different aspect of evaluation can be the time segmentation, more specifically of the system tends to over- or undersegment the chord labels.

7.6.3.2 Datasets

Although there is a reasonable amount of time-aligned annotated data available by now, often the annotations come without audio data [BLEW04, BWF11, CT11, KHB⁺19] except for a minority of datasets [Got02, DJA⁺18].

7.6.3.3 Results

The issues mentioned above make automatic chord recognition a challenging with a lot of room to improve. Furthermore, the chord imbalance combined with the number of metrics and huge variations related to the test datasets make reporting a specific number range questionable. The interested reader is referred to previous MIREX campaigns for detailed results.¹⁰

7.7 Exercises

7.7.1 Questions

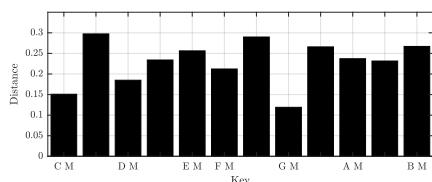
1. Describe the relationship between frequency and perceived pitch.
2. How many pitch classes is an octave divided into (for western music)?
3. If the frequency of 440 Hz is mapped to the MIDI pitch 69, what is the MIDI pitch corresponding to a frequency of 880 Hz?
4. What is the pitch distance in Cent between the two frequencies 600 Hz and 900 Hz?
5. Name 3 approaches traditionally used for monophonic fundamental frequency detection.
6. Why can pre-processing with a low-pass filter potentially increase the robustness of ACF-based fundamental frequency detection? How do you choose an appropriate cut-off frequency?

¹⁰https://www.music-ir.org/mirex/wiki/2018:Audio_Chord_Estimation_Results, last access date Aug 9, 2021

7. Consider the simplest implementation of an ACF-based fundamental frequency detection. If the first non-zero maximum occurs at lag 100, what is the estimated fundamental frequency? Assume a sample rate of 40 kHz.
8. You have a monophonic audio signal likely missing some harmonics. Would the HPS or the HSS be a better method to detect the fundamental frequency? Why?
9. The frequency resolution of FFT-based fundamental frequency detection determines the maximum frequency accuracy of your result. Is this potential problem more pronounced for lower input frequencies or higher input frequencies. Describe two ways of addressing this problem.
10. Which of the following signals might result in unreliable instantaneous frequency estimates?
- an amplitude-modulated sinusoidal
 - two sinusoids with very close frequencies
 - a high frequency rectangular wave
11. Assuming a sample rate of 40 kHz and a hop size of 400 samples, what is the instantaneous frequency in Hz if the phase difference at one bin is $2\pi \cdot 0.01$?
12. List and highlight the main challenges of polyphonic pitch detection as compared to monophonic pitch detection.
13. Describe the general process of the 'iterative subtraction' paradigm for polyphonic pitch detection.
14. Given a spectrogram of size 1025×2000 (frequency \times blocks), name the dimensions of both the template matrix \mathbf{W} and the activation matrix \mathbf{H} when estimating $\hat{\mathbf{X}} = \mathbf{WH}$ with NMF with a rank $r = 15$.
15. Given the following matrix, name one optimal (no error) factorization result (\mathbf{W}, \mathbf{H}) .

$$\mathbf{X} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 10 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 15 & 15 & 0 \end{bmatrix}$$

16. Describe how a tuning frequency estimate might improve a pitch-based audio analysis task like chord detection.
17. What are the dimensions of the average pitch chroma computed over N blocks?
18. Given the distance plot below between an extracted pitch chroma and a shifted template, which is the most likely key of this signal?



19. You observe a pitch chroma vector that reads $v_{pc} = [0.000.350000.26000.2900.1]$. What chord is most likely?
20. Is the following statement true? "If an emission probability is high for a given pair of observation and state, it ensures the occurrence of this state in the final decoded state sequence."

21. Consider a HMM with the two chords C, D as the hidden states and two pitch chroma observations $v_{pc,1}, v_{pc,2}$. Assume the initial state is *Start* and we know the following parameters:

$$\begin{aligned} P(C \rightarrow D) &= 0.3 \\ P(D \rightarrow D) &= 0.2 \\ P(v_{pc,1}|C) &= 0.1 \\ P(Cv_{pc,2}|D) &= 0.4 \\ P(Start \rightarrow C) &= 0.9. \end{aligned}$$

- (a) What is $P(Start \rightarrow D)$?
- (b) What is $P(C \rightarrow C)$?
- (c) What is $P(D \rightarrow C)$?
- (d) What is $P(v_{pc,1}|D)$?
- (e) What is $P(v_{pc,2}|C)$?

7.7.2 Assignments

1. Block-wise pitch tracking with the ACF

- (a) Blocking and fundamental frequency extraction
 - Implement a function `block_audio(x, blockSize, hopSize, fs)` which returns a matrix \mathbf{XB} (dimension $\mathcal{N} \times \mathcal{K}$) and a vector t_B (dimension \mathcal{N}) for blocking the input audio signal into overlapping blocks. t_B will refer to the start time of each block in seconds.
 - Implement a function `comp_acf(inputVector, bIsNormalized)` which computes the ACF for each block and returns the non-redundant (right) part of the result. Normalization is controlled by parameter `bIsNormalized`. Do not call any functions not implemented by you.
 - Implement a function `get_f0_from_acf(r_acf, fs)` that takes the output of `comp_acf` and computes and returns the fundamental frequency estimate f_0 of that block in Hz.
 - Implement a 'main' function `track_pitch_acf(x, blockSize, hopSize, fs)` that reads an audio signal, calls the three functions above, and returns two vectors `f0` and `timeInSec`.
- (b) Testing and Evaluation
 - Implement code for
 - generating a sinusoidal test signal (0-1 s: $f = 441$ Hz, 1-2 s: $f = 882$ Hz, $f_S = 44.1$ kHz),
 - calling your `track_pitch_acf` function with this test signal,
 - plotting both, the resulting f_0 and the error in Hz of the output, and discuss possible reasons for the deviation.
 - Download the tonas dataset [add reference](#).
 - Implement a function `convert_freq2midi(freqInHz)` that returns a variable `pitchInMIDI` of the same dimension as input `freqInHz`. Note that the dimension of `freqInHz` can be a scalar, a vector, or a matrix. Assume $f_{A4} = 440$ Hz.
 - Implement a function `eval_pitchtrack(estimateInHz, groundtruthInHz)` that computes the RMS of the error in Cent in the pitch domain (not frequency) and returns this as `errCentRms`. Ignore blocks with `groundtruthInHz = 0`.
 - Implement a function `run_evaluation(complete_path_to_data_folder)` returning the overall `errCentRms`. This requires implementing a directory parser, file reader, and a loop calling your pitch tracker. The annotation form for tonas is `[onset_seconds, duration_seconds, pitch_frequency]`.
 - Run your implemented pitch tracker with the dataset and report the overall `errCentRms`. Discuss whether the value is high or low and speculate why.

(c) Bonus

- Implement a function `track_pitch_acfmod(x, blockSize, hopSize, fs)` that modifies your method and improves the performance of your basic ACF pitch tracker thereby providing better f_0 estimations. You may apply constraints to your approach, pre-processing or post-processing steps or any other additional processing steps that might improve the result. Verify the improved result by running your evaluation.

2. Block-wise pitch tracking in the spectral domain

(a) Fundamental frequency detection with the maximum of the magnitude spectrum

- Implement a function `[X, fInHz] = compute_spectrogram(xb, fs)` that computes the magnitude spectrum for each block of audio in `xb` (calculated using `block_audio()`) and returns the magnitude spectrogram `X` (dimensions $\mathcal{K}/2 + 1 \times \mathcal{N}$) and a frequency vector `fInHz` (dimension $\mathcal{K}/2 + 1$) containing the central frequency of each bin. Parameter `blockSize` specifies the length of the FFT \mathcal{K} . Note: remove the redundant part of the spectrum. Also note that you will have to apply a von-Hann window of appropriate length to the blocks before computing the FFT.
- Implement a function: `[f0, timeInSec] = track_pitch_fftmax(x, blockSize, hopSize, fs)` that estimates the fundamental frequency `f0` of the audio signal based on a block-wise maximum spectral peak finding approach. Note: This function should use `compute_spectrogram()`.

(b) Fundamental frequency detection with the HPS

- Implement a function `[f0] = get_f0_from_Hps(X, fs, order)` that computes the block-wise fundamental frequency `f0` given the magnitude spectrogram `X` and the sampling rate based on a HPS approach of order `order`.
- Implement a function `[f0, timeInSec] = track_pitch_hps(x, blockSize, hopSize, fs)` that estimates the fundamental frequency `f0` of the audio signal block based on HPS based approach. Use `blockSize = 1024` in `compute_spectrogram()`. Use `order = 4` for `get_f0_from_Hps()`.

(c) Evaluation

- Evaluate your results with the function and data above.

3. Voicing detection

(a) Voicing detection with an RMS threshold

- Implement a function `[rmsDb] = extract_rms(xb)` which takes the blocked audio as input and computes the RMS amplitude of each block.
- Implement a function `[mask] = create_voicing_mask(rmsDb, thresholdDb)` which takes a vector of decibel values for the different blocks of audio and creates a binary mask based on the threshold parameter. Note: A binary mask in this case is a simple column vector of the same size as `rmsDb` containing 0's and 1's only. The value of the mask at an index is 0 if the `rmsDb` value at that index is less than `thresholdDb` and the value is 1 if `rmsDb` value at that index is greater than or equal to the threshold.

(b) Voicing evaluation

- Implement a function `[f0Adj] = apply_voicing_mask(f0, mask)` which applies the voicing mask to the previously computed `f0` so that the `f0` of blocks with low energy is set to 0.
- Implement a function `[pfp] = eval_voiced_fp(estimation, annotation)` that computes the percentage of false positives for your fundamental frequency estimation. The denominator is the number of blocks for which `annotation = 0`. The numerator would be how many of these blocks were classified as voiced (with a fundamental frequency not equal to 0) is your estimation.

- Implement a function `[pfn] = eval_voiced_fn(estimation, annotation)` that computes the percentage of false negatives for your fundamental frequency estimation. In this case the denominator is number of blocks which have non-zero fundamental frequency in the annotation, the numerator is number of blocks out of these that were detected as zero in the estimation.
- Modify the `eval_pitchtrack()` method above to `[errCentRms, pfp, pfn] = eval_pitchtrack_v2(etc)` to return all the 3 performance metrics for your fundamental frequency estimation. Note: the `errorCentRms` computation will slightly change now so you have to update considering that your estimation might also contain zeros.

Chapter 8

Intensity

Intensity, magnitude, and loudness-related features constitute one of the most commonly used classes for the description of audio content. Most audio editors and digital audio workstations illustrate the audio signal in its waveform view, its amplitude variation over time. There also exists a variety of instruments for level, volume, and loudness measurements frequently used in recording studio environments.

8.1 Human Perception of Intensity and Loudness

It is important to distinguish the meaning of the terms *intensity* and *loudness*. Intensity means a physical, measurable entity such as the magnitude of a sound while loudness refers to a perceptual entity that can only be measured via responses of human observers [Ste38]. Unfortunately, the term *loudness* is also used for *algorithmic models* of the perceived loudness.

Human perception of intensity is related to the magnitude of the audio signal in a way that if the signal's magnitude is scaled up, the perceived loudness will increase as well. It has been discovered very early that this relationship is non-linear; a linear increase in the signal's magnitude or power will not result in a linear increase in perceived loudness. An approximately linear relation could be found by using the pseudo-unit *decibel* (dB) which is computed by taking the logarithm of the intensity feature $v(n)$ (computed from a block of samples, see below):

$$v_{\text{dB}}(n) = 20 \cdot \log_{10} \left(\frac{v(n)}{v_0} \right) = 10 \cdot \log_{10} \left(\frac{v^2(n)}{v_0^2} \right) \quad (8.1)$$

with v_0 representing a reference constant. In the digital domain, dealing with audio amplitudes in the range of $[-1; 1]$, it is commonly set to $v_0 = 1$. The resulting level unit is then referred to as dBFS (dB *full scale*) and has a range of $-\infty \leq v_{\text{dB}}(n) \leq 0 \text{ dB}$.¹ The preceding scaling factor has been chosen to scale the non-linear function so that 1 dB roughly represents the level difference a human can easily recognize. This is only a rough approximation as the actual so-called *Just Noticeable Difference in Level (JNDL)* depends on the stimulus level and partly on stimulus frequency and masking effects [ZF99].

As the input magnitudes decrease to silence, the function decreases to ∞ . To avoid this (and very low values for inputs close to 0), two approaches are common. First, a threshold ϵ can be defined so that all inputs lower than ϵ will be set to ϵ :

$$v_{\text{trunc}}(n) = \begin{cases} v(n), & \text{if } v(n) \geq \epsilon \\ \epsilon, & \text{otherwise.} \end{cases} \quad (8.2)$$

This approach comes at the cost of an additional `if` statement per feature value.

Second, the calculation of $\log(0)$ can be avoided by adding a small constant ϵ :

$$v_{\text{approx,dB}}(n) = 20 \cdot \log_{10}(v(n) + \epsilon). \quad (8.3)$$

¹Note that higher dBFS levels are possible if the implementation allow for them.

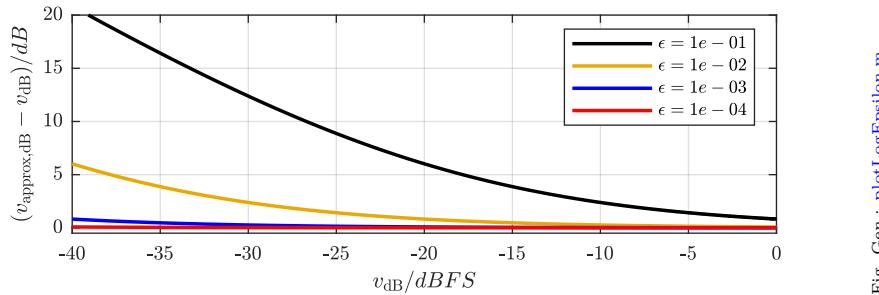


Fig. Gen.: plotLogEpsilon.m

Figure 8.1: Level error introduced by adding a small constant ϵ to the argument of a logarithm.

The choice of ϵ determines the measurement accuracy at low-level inputs. The measurement error increases for decreasing $v(n)$ approaching ϵ . It will be 6 dB for $v(n) = \epsilon$ and increase with lower levels. Figure 8.1 visualizes the error amount for different ϵ .

The decibel scale is not a loudness scale since equal-sized steps on the decibel scale are not perceived as equal-sized loudness steps by human listeners: doubling the level in dB does not result in doubling the perceived loudness of a sound. Stevens, summarizing several loudness perception studies, proposed a simple rule of thumb stating that a doubling of the perceived loudness corresponds to a level increase of 10 dB [Ste55].

More accurate models of the human perception of loudness take both the input signal's frequency and the cochlea's frequency resolution into account as has been shown by several studies throughout the last century. The most important researchers in the history of loudness perception are probably Fletcher and Munson [FM33, Fle40], Stevens [Ste38, Ste55], Moore [MG83], and Zwicker and Fastl [ZF99].

8.2 Representation of Dynamics in Music

In a traditional musical score, loudness-related performance instructions are rather vague. Usually only five to eight different dynamic steps are used to describe musical dynamics (e.g., **p**: *pianissimo*, **p**: *piano*, **mf**: *mezzoforte*, **f**: *forte*, **ff**: *fortissimo* for the dynamic range from “very soft” to “very strong”), complemented by indications of smooth loudness transitions (e.g., *crescendo* or *decrescendo* for increasing and decreasing loudness, respectively) and dynamic accents (e.g., **sf**: *sforzando*). These written instructions do not directly refer to absolute loudness as it also depends on a number of other influencing factors such as instrumentation, timbre, number of voices, performance, and musical tension and musical context. The indifference to absolute loudness may also be illustrated by the fact that while listening to a recording on a hi-fi system, the reproduction volume may be manipulated without losing the *piano* or *forte* character of the performance. Still, Nakamura has shown that measures of intensity or loudness can, to a certain degree, be used as indications of musical dynamics [Nak87]. A loudness-related performance attribute is the *tremolo*, the periodic modulation of loudness over time. It often appears in combination with a *vibrato*.

A more technical representation of dynamics in music is the *velocity* as standardized in the MIDI protocol [MID01]. It consists of 128 volume steps with the highest representing maximal intensity. But although the number of velocity steps is standardized, there is no standardized relationship between MIDI velocity and intensity: Goebel and Bresin found that for the Yamaha Disklavier and the Bösendorfer SE System (both pianos allowing for the monitoring of performance data), the relationship between MIDI velocity and (logarithmic) sound pressure level is nearly linear when disregarding very low and high values [GB03]. Dannenberg investigated the RMS peak level of various synthesizers and software instruments and found great differences among different synthesizers [Dan06]. He identified a general trend for the velocity to be related to the square root of the RMS peak instead of its logarithm. Using one single electronic instrument, Taguti measured the A-weighted sound pressure level dependent on velocity and key [Tag03]. The results, displayed over various keys for different input velocities, showed non-systematic deviations of up to 10 dB from a constant level among keys.

8.3 Features

Many of the intensity features presented here are instantaneous features similar to the features introduced in Chap. 3.6. Therefore, the same post-processing options (see Sect. 3.7) can be applied to most of these features. The features presented in this section can be roughly structured into three categories: physical measures of sound intensity, approaches to measure intensity in recording studio environments, and psycho-acoustically motivated features modeling the human perception of loudness.

8.3.1 Root Mean Square

The *RMS* is one of the most common intensity features and is sometimes directly referred to as the sound intensity. It is calculated from a block of audio samples by

$$v_{\text{RMS}}(n) = \sqrt{\frac{1}{K} \sum_{i=i_s(n)}^{i_e(n)} x(i)^2}. \quad (8.4)$$

Note that for blocks with zero mean, the RMS resembles the variance (see Sect. 3.1.3.4). Typical block lengths for the RMS calculation are in the range of several hundred milliseconds. The length in seconds is also referred to as the *integration time*.

The result of the calculation is a value within the range $0 \leq v_{\text{RMS}}(n) \leq 1$ (as long as the input amplitude 1 represents full scale. It will equal 0 if the input is silence and will approach 1 for both a square wave with maximum amplitude and a constant DC offset at ± 1 . Sharp transients in the signal will be smoothed out by an RMS measure due to the comparably long integration time. The RMS is often reported in dBFS.

```

T_i          = .3;
alpha        = 1-exp(-2.2/f_s/T_i);

% number of results
iNumOfBlocks = floor ((length(x)-iBlockLength)/iHopLength + 1);

% compute time stamps
t            = ((0:iNumOfBlocks-1) * iHopLength + (iBlockLength/2))/f_s;

% allocate memory
vrms         = zeros(2,iNumOfBlocks);

% single pole implementation
v_sp         = filter(alpha, [1 -(1-alpha)],x.^2);

for (n = 1:iNumOfBlocks)
    i_start   = (n-1)*iHopLength + 1;
    i_stop    = min(length(x),i_start + iBlockLength - 1);

    % calculate the rms
    vrms(1,n) = sqrt(mean(x(i_start:i_stop).^2));
    vrms(2,n) = max(sqrt(v_sp(i_start:i_stop)));

```

(a) Matlab

```

# create blocks
xBlocks = pyACA.ToolBlockAudio(x, iBlockLength, iHopLength)

# number of results
iNumOfBlocks = xBlocks.shape[0]

# compute time stamps
t = (np.arange(0, iNumOfBlocks) * iHopLength + (iBlockLength / 2)) / f_s

# allocate memory
vrms = np.zeros(iNumOfBlocks)

for n, block in enumerate(xBlocks):
    # calculate the rms
    vrms[n] = np.sqrt(np.dot(block, block) / block.size)

    # convert to dB
    epsilon = 1e-5 # -100dB

    vrms[vrms < epsilon] = epsilon
    vrms = 20 * np.log10(vrms)

```

(b) Python

Figure 8.2: Implementation of Rms.

The calculation of the RMS can be computationally inefficient for large block lengths and small hop sizes \mathcal{H} . If the hop size is one sample, it is possible to reduce the number of computations by using the method of recursive implementation introduced in the context of the MA filter in Sect. A.2.6.1:

$$v_{\text{RMS}}^2(n) = \frac{x(i_e(n))^2 - x(i_s(n-1))^2}{i_e(n) - i_s(n) + 1} + v_{\text{RMS}}^2(n-1), \quad (8.5)$$

$$v_{\text{RMS}}(n) = \sqrt{v_{\text{RMS}}^2(n)}. \quad (8.6)$$

This implementation is computationally efficient but still requires a significant amount of memory to be allocated for large block lengths. An approximation of the RMS v_{RMS}^* with hop size $\mathcal{H} = 1$ can be implemented with a single-pole filter (compare Sect. A.2.6.2):

$$v_{\text{tmp}}(i) = \alpha \cdot v_{\text{tmp}}(i-1) + (1-\alpha) \cdot x(i)^2 \quad (8.7)$$

$$v_{\text{RMS}}^*(i) = \sqrt{v_{\text{tmp}}(i)}. \quad (8.8)$$

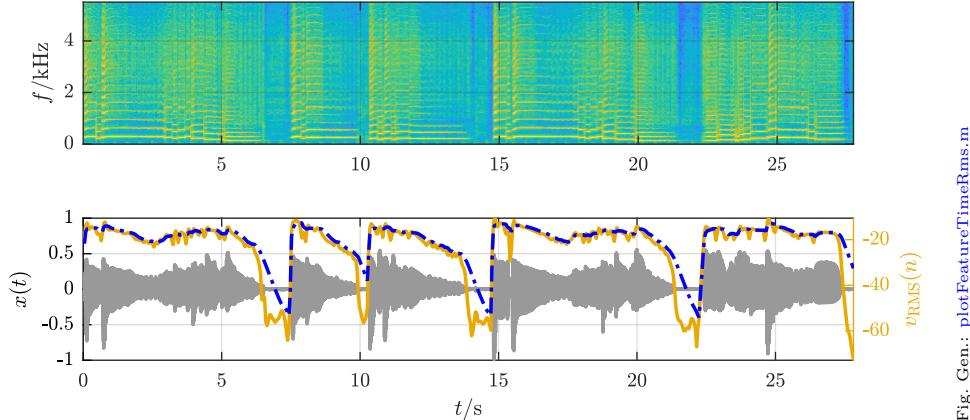


Figure 8.3: Spectrogram (top), waveform (bottom background), and RMS (bottom foreground, gold: standard, blue: single pole) of a saxophone signal.

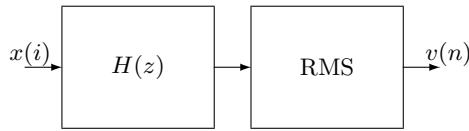


Figure 8.4: Flowchart of the frequency-weighted RMS calculation

The filter coefficient α can be estimated from the integration time (or block length) with Eq. (A.27).

Figure 8.3 shows the RMS of an example signal for the two implementations described by Eqs. (8.4) and (8.7). The RMS is a measure of the power of the signal; the two implementations shown are roughly equivalent except during sudden signal pauses in which the low-pass filtered variant only slowly decreases.

8.3.2 Weighted Root Mean Square

In many scenarios the RMS computation is preceded by a weighting filter to simulate the frequency sensitivity of the human ear. This means the audio signal is simply filtered before computing the RMS as described above (see Fig. 8.4). The transfer function of a weighting filter is usually modeled after an inverse *equal-loudness contour* which measures the level for which a listener perceives equal loudness at different frequencies [FM33]. Thus, the weighting filter amplifies frequency regions in which the human ear is sensitive and attenuates other regions.

Common transfer functions of the weighting filter are:

- *A weighting*: weighting function to be used for sounds at low level [IEC02],
- *C weighting*: weighting function to be used for sounds at medium level [IEC02],
- *RLB weighting*: weighting function according to ITU-R BS.1770 (plus high-frequency emphasis for multichannel signals) [BS.06], and
- *CCIR weighting*: weighting function according to ITU-R BS.468 [BS.86].

Figure 8.5 displays the listed transfer functions on the left; the right plot shows the equal-loudness contours for comparison. Note the different shape of the contours for different levels. When frequency weighted RMS measures are used as models of loudness, the integration time is usually several seconds in order to ignore short-term variations of the signal.

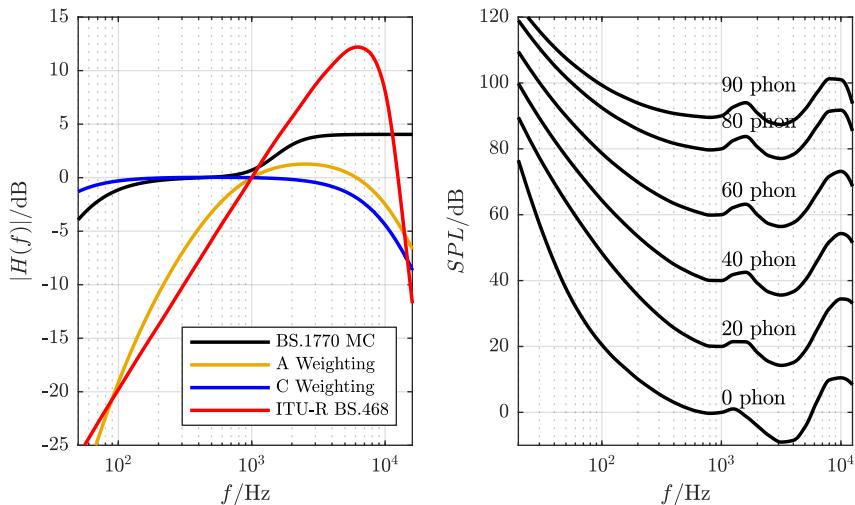
Fig. Gen.: `plotLoudnessWeighting.m`

Figure 8.5: Left: Frequency weighting transfer functions applied before RMS measurement, Right: Equal Loudness Contours [ISO03].

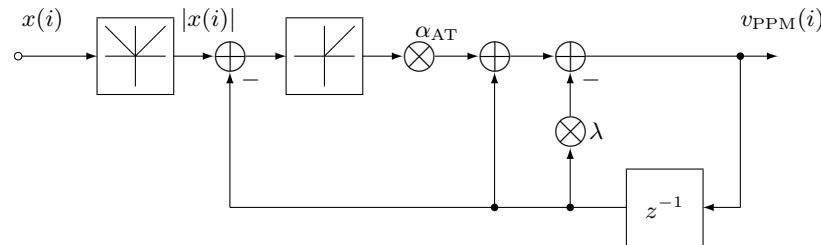


Figure 8.6: Flowchart of a Peak program meter

8.3.3 Peak Envelope

The *peak envelope* of an audio signal can be extracted in different ways. The simplest way of extracting the envelope is to find the absolute maximum per block of audio samples:

$$v_{\text{Peak}}(n) = \max_{i_s(n) \leq i \leq i_e(n)} |x(i)|. \quad (8.9)$$

The envelope can also be extracted on a sample-per-sample basis with a so-called *Peak Program Meter (PPM)*. The PPM, frequently used in recording studio environments, operates with different integration times for attack (*attack time*) and release (*release time*). Typically, the attack time is significantly shorter than the release time (e.g., attack time ≈ 10 ms, release time ≈ 1500 ms) which means that the output reflects an increase in level faster than a decrease. This originates in requirements of recording engineers: the attack time has to be short in order to allow the systems to detect short peaks while the longer release time gives humans more time to actually see those peaks.

The structure of a digital PPM as described by Zölzer is shown in Fig. 8.6 [Zö08]. The filter coefficient representing the attack time is named α_{AT} . The release time coefficient α_{RT} is not always in use and is being represented by λ in the figure. More specifically, λ has the following two states depending on the input magnitude

$$\lambda = \begin{cases} \alpha_{RT}, & \text{if } |x(i)| < v_{PPM}(i-1) \\ 0, & \text{otherwise.} \end{cases} \quad (8.10)$$

The two states may be referred to as *release state* and *attack state*. The corresponding output equations are

- Release state:

$$\begin{aligned} v_{\text{Peak}, \text{PPM}}(i) &= v_{\text{PPM}}(i-1) - \alpha_{\text{RT}} \cdot v_{\text{PPM}}(i-1) \\ &= (1 - \alpha_{\text{RT}}) \cdot v_{\text{PPM}}(i-1), \end{aligned} \quad (8.11)$$

- Attack state:

$$\begin{aligned} v_{\text{Peak}, \text{PPM}}(i) &= \alpha_{\text{AT}} \cdot (|x(i)| - v_{\text{PPM}}(i-1)) + v_{\text{PPM}}(i-1) \\ &= \alpha_{\text{AT}} \cdot |x(i)| + (1 - \alpha_{\text{AT}}) \cdot v_{\text{PPM}}(i-1). \end{aligned} \quad (8.12)$$

The result of both v_{Peak} and $v_{\text{Peak}, \text{PPM}}$ is a value within the range $0 \leq v_{\text{Peak}} \leq 1$. It will equal 0 if the input is silence and approach 1 for certain full-scale signals. Figure 8.7 shows the implementation.

```
% number of results
iNumOfBlocks = floor ((length(x)-iBlockLength)/iHopLength + 1);

% compute time stamps
t = ((0:iNumOfBlocks-1) * iHopLength + (iBlockLength/2))/f_s;

% allocate memory
vppm = zeros(2,iNumOfBlocks);
v_tmp = zeros(1,iBlockLength);

%initialization
alpha = 1 - [exp(-2.2 / (f_s * 0.01)), exp(-2.2 / (f_s * 1.5))];

for (n = 1:iNumOfBlocks)
    i_start = (n-1)*iHopLength + 1;
    i_stop = min(length(x),i_start + iBlockLength - 1);

    % calculate the maximum
    vppm(1,n) = max(abs(x(i_start:i_stop)));

    % calculate the PPM value - take into account block overlaps
    % and discard concerns wrt efficiency
    v_tmp = ppm(x(i_start:i_stop), v_tmp(iHopLength), alpha);
    vppm(2,n) = max(v_tmp);

end

% convert to dB
epsilon = 1e-5; %-100dB

i_eps = find(vppm < epsilon);
vppm(i_eps) = epsilon;
vppm = 20*log10(vppm);
end

function [ppmout] = ppm(x, filterbuf, alpha)

%initialization
alpha_AT = alpha(1);
alpha_RT = alpha(2);

x = abs(x);
for (i = 1: length(x))
    if (filterbuf > x(i))
        % release state
        ppmout(i) = (1-alpha_RT) * filterbuf;
    else
        % attack state
        ppmout(i) = alpha_AT * x(i) + (1-alpha_AT) * filterbuf;
    end
    filterbuf = ppmout(i);
end
(a) Matlab
```

```
# create blocks
xBlocks = pyACA.ToolBlockAudio(x, iBlockLength, iHopLength)

# number of results
iNumOfBlocks = xBlocks.shape[0]

# compute time stamps
t = (np.arange(0, iNumOfBlocks) * iHopLength + (iBlockLength / 2)) / f_s

alpha = 1 - np.array([np.exp(-2.2 / (f_s * 0.01)), np.exp(-2.2 / (f_s * 1.5))])

# allocate memory
vppm = np.zeros([2, iNumOfBlocks])
v_tmp = np.zeros(iBlockLength)

for n, block in enumerate(xBlocks):
    x_block = np.abs(block)

    # detect the maximum per block
    vppm[0, n] = np.max(x_block)

    # calculate the PPM value - take into account block overlaps
    # and discard concerns wrt efficiency
    v_tmp = ppm(x_block, v_tmp[iHopLength - 1], alpha)
    vppm[1, n] = np.max(v_tmp)

# convert to dB
epsilon = 1e-5 # -100dB

vppm[vppm < epsilon] = epsilon
vppm = 20 * np.log10(vppm)

return vppm, t

def ppm(x, filterbuf, alpha):

    # initialization
    ppmout = np.zeros(x.shape[0])

    alpha_AT = alpha[0]
    alpha_RT = alpha[1]

    for i in range(0, x.shape[0]):
        if filterbuf > x[i]:
            # release state
            ppmout[i] = (1 - alpha_RT) * filterbuf
        else:
            # attack state
            ppmout[i] = alpha_AT * x[i] + (1 - alpha_AT) * filterbuf
            filterbuf = ppmout[i]
(b) Python
```

Figure 8.7: Implementation of PeakEnvelope.

Figure 8.8 shows the results of both envelope measures. The comparison with the two RMS results as shown in Fig. 8.3 reveals similar behavior of RMS and peak measures as the calculation is relatively similar. The dotted peak maximum shows faster and more pronounced changes as can be clearly seen during the pauses with the PPM's constant decrease due to the long release time.

8.3.4 Psycho-Acoustic Loudness Features

There exist complex loudness measurements based on either psycho-acoustic properties of human loudness perception or physiological models of the human ear (or both). These are not frequently used for ACA, as

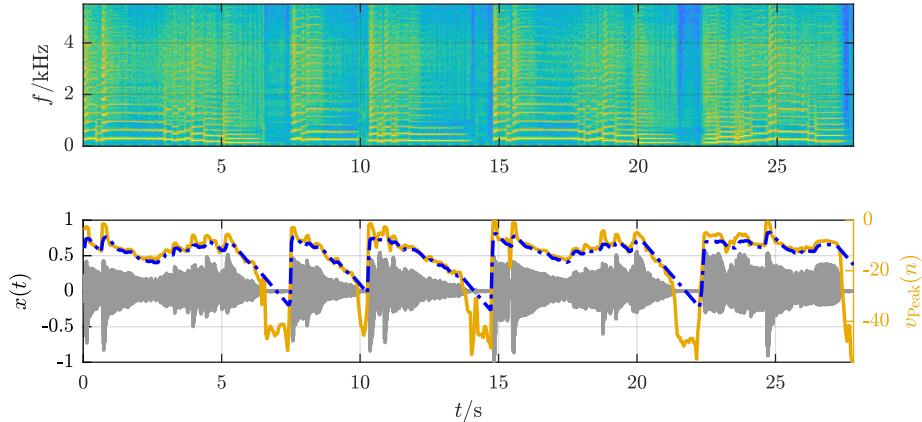


Fig. Gen.: plotFeatureTimePeakEnvelope.m

Figure 8.8: Spectrogram (top), waveform (bottom background), and PPM output compared to the magnitude maximum per block (bottom foreground, blue and gold, respectively) of a saxophone signal.

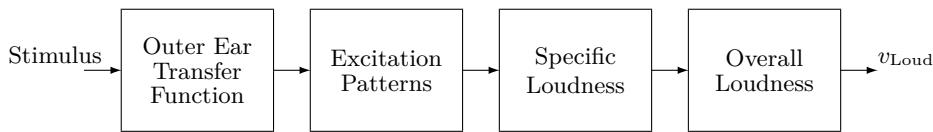


Figure 8.9: Flowchart of Zwicker's model for loudness computation

(i) they are comparably costly to implement and to compute and (ii) there are indications that they are in many cases equally meaningful as simpler loudness approximations such as the algorithm described in *International Telecommunication Union (ITU)* recommendation BS.1770, a weighted RMS solution [SN03].

A widely known and psycho-acoustically motivated loudness measurement has been proposed by Zwicker [45691, ZF99, ISO17]. Figure 8.9 presents the flowchart of this loudness calculation. The signal is transformed into the bark domain (see Sect. 7.1) by a filterbank or a similar frequency transform. The so-called *excitation patterns* are computed from the filterbank outputs by taking into account frequency sensitivity and masking effects. Then, the *specific loudness* is calculated from the excitation patterns per band. The resulting loudness is the sum of the specific loudness over all bands.

One relatively recent recommendation for measurement of the loudness of a program or a file has been published by the *European Broadcasting Union (EBU)* [EBU10]. The loudness itself is measured in compliance to ITU recommendation BS.1770, an RMS measure with an *RLB weighting*. The required block length is 3.0 s with a block overlap ratio of at least 66%. Blocks with very low loudness are discarded with a gating threshold. The EBU recommendation also requires the following additional values to be extracted from the audio signal:

- clipped values according to an up-sampled true peak meter, and
- the loudness range computed from a histogram of all loudness block results as the difference $Q_v(0.95) - Q_v(0.10)$.

8.4 Exercises

8.4.1 Questions

1. Given a signal magnitude of 1, what is its level in dBFS?
2. When implementing the computation of a level in decibel, discuss solutions for dealing with an input signal equaling 0.

8.4.2 Assignments

Chapter 9

Temporal Analysis

The temporal aspects of music signals such as the tempo and the rhythm are important musical properties. For example, a simple sequence of pitches does not make a melody without temporal, structured patterns grouping the pitches [CM63]. A fundamental element for the analysis of such temporal aspects is the onset: the beginning of a musical sound event such as a tone or the stroke on a percussive instrument. The start time of an event is usually considered to be more important than the time of the end of that event, as listeners apparently tend to perceive musical events in terms of Inter-Onset Intervals (IOIs) [LP02].

9.1 Human Perception of Temporal Events

During the process of human perception, the audio stream will be segmented into a series of events. Speaking of segmentation is a simplification because musical meaning and even rhythm can be conveyed by audio streams with no such clear division into distinct events [Wri08]. However, this simplification can be assumed to be sufficiently valid in the context of western music for the majority of possible input signals — other incarnations of temporal information will be ignored for the sake of simplicity.

9.1.1 Onsets

As stated above, an *onset* is the start of a (musical) sound event. The term *onset* is frequently used synonymous to onset time, but it should be more correct to state that its time position (i.e., the onset time) is one —probably the main— property of the onset while it can have other properties such as its strength.

The onset time, however, is not easily pinpointed either. In reality, the start of a musical sound usually is not an exact point in time, but a time span. This time span is referred to as the *attack time* or *initial transient time*, starting with the first instrument-induced oscillation and ending if the maximum amplitude is reached or the sound reaches its quasi-periodic state. An example of an attack phase of a note is shown in Fig. 9.1.

The attack time varies considerably between different musical instruments, groups of instruments, and playing techniques. It ranges from about 5 ms for some percussive instruments to up to 200 ms for woodwind

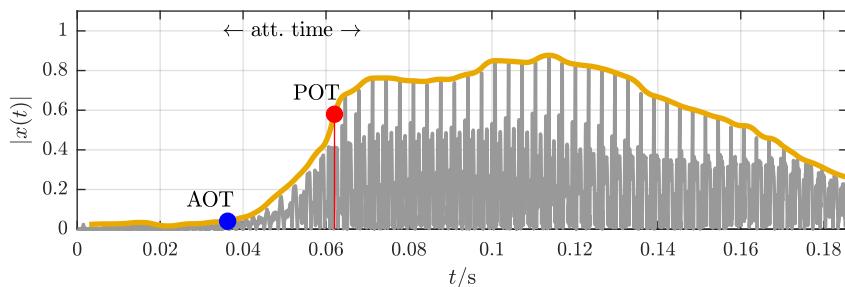


Fig. Gen.: `plotOnset.m`

Figure 9.1: Visualization of an envelope and attack time and one possible location of the perceptual onset time.

instruments (flute) under certain conditions [Reu95].

The terms *onset*, *attack*, and *transient* are often used inconsistently and interchangably, which complicates exact communication. Only the context gives insights into whether these terms have the same meaning or different meanings, or whether they stand for a signal state, a time span, or a point in time. To give an example of a different naming convention than the one used here, Bello et al. propose to use the terms *attack* for our attack time, *transient* as a description of the initial phase of a musical event in which “the signal evolves quickly in some nontrivial or relatively unpredictable way” (the period covered by our attack time), and *onset* as a single instant chosen to mark the temporally extended transient (our onset time) [BDA⁺05].

Repp pointed out that three definitions of onset times can generally be distinguished [Rep96b]:

1. *Note Onset Time (NOT)*: the time when the instrument is triggered to make a sound. In the MIDI domain, the NOT is exactly the time of the Note-On command. Depending on the instrument or sample used for sound generation, this is not necessarily the time when the signal becomes detectable or audible.
2. *Acoustic Onset Time (AOT)*: the first time when a signal or an acoustic event is theoretically measurable. Sometimes the AOT is called *physical onset time*.
3. *Perceptual Onset Time (POT)*: the first time when the event can be perceived by the listener. The POT might also be distinguished from the *Perceptual Attack Time (PAT)* which is the instant of time that is relevant for the perception of rhythmic patterns [Gor84]. While the PAT might occur later than the POT, they will be equal in many cases. For the sake of simplicity, there will be no distinction of POT and PAT in the following.

The POT can never occur before the AOT, which in turn never occurs before the NOT. Due to the “perceptual” definition of the POT, the exact location cannot be determined acoustically but has to be measured in a listening test. Both Gordon and Zwicker found strong location drifts of the PAT depending on the waveform properties during the rise time [Gor84, ZF99]. There are indications of the POT to be correlated with the envelope slope [Gor84].

Given the three definitions above, the following question arises: which of the three onset times is on the one hand detectable in the signal and on the other hand of the utmost interest in automatic onset detection and any rhythm-related task? Due to the symbolic nature of the NOT, it simply cannot be detected from the audio signal. The choice between AOT and POT might be application-dependent; assuming that musicians adapt their timing to their sound perception and that most ACA systems try to analyze the *perceptible* audio content, the POT is most likely the point in time desired as result. This reflection, however, is rather academic since in reality the time accuracy of automatic onset detection systems is usually too poor to differentiate between the different onset times for all but a small class of signal types. Furthermore, the time accuracy of ground-truth data for testing practically limits what accuracy can be tested for.

As these data are annotated by humans, the human ability to locate onset times and to distinguish closely spaced onsets is limiting the expected ground truth accuracy. Furthermore, since most ACA systems aim to be at least as accurate as the human perception, this perceptual detection accuracy also allows useful insights into a reasonable time accuracy of an onset detection system. In the following, a few noteworthy studies on the perception of onset times are summarized to clarify the time ranges under consideration.

Hirsh found that temporal discrimination of two onsets is possible for humans if the onset time difference is as little as 2 ms [Hir59]. However, in order to determine the order of the stimuli, their distance had to be about 20 ms. The measurements were done with synthetic signals with short rise times.

Gordon reported a standard deviation of 12 ms for the accuracy of onset times specified by test listeners, using 16 real-world monophonic sounds of different instruments played in an infinitely long loop pattern with *Inter-Onset Intervals (IOIs)* of 600 ms [Gor84]. Friberg and Sundberg undertook a similar experiment using tone stimuli [FS92]. For IOIs smaller than 240 ms, they reported a just noticeable difference of about 10 ms, and increasing values for larger IOIs.

Repp reported for the manual annotation of onset times by one listener in the context of piano recordings a mean absolute measurement error of about 4.3 ms and a maximum error of about 35 ms [Rep92]. Leveau et al. compared the results of three test subjects annotating the onset times in audio files of various genres and

instrumentations [LDR04]. The results showed a mean absolute measurement error over all test data of about 10 ms; for one piece of classical music, the mean absolute measurement error nearly reached 30 ms.

Rasch evaluated the onset time differences between instruments in three ensemble performances [Ras79]. He found synchronization deviations in a range between 30 and 50 ms between the (string and woodwind) instruments, while the mean onset time differences were in the range of ± 6 ms. However, since the measurement accuracy has not been evaluated in this case, it is unknown how much of the actual time differences can be attributed to the performance itself.

For piano duet performance, Shaffer reported standard deviations within the voices between 14 and 38 ms [Sha84].

It may be concluded that the accuracy of human onset perception depends on the test data and that deviations evoked by motoric abilities seem to be in the same range. The presented results imply that an automatic onset detection system aiming at human detection accuracy (or being evaluated with test data annotated by humans) will have a minimum mean absolute error in the range of 5–10 ms; the error can be expected to be as high as 10 times more for specific instruments and pitches with long rise times.

9.1.2 Tempo and Meter

The *tempo* determines the ‘pace’ of a musical piece [LC96]. It is the rate of occurrence of perceived periodic events (*beats*) at regular intervals [McA10], with these intervals occurring at a moderate and natural rate [DH93]. This perceived tempo is called the *tactus* [LJ83] and is sometimes simply referred to as the *foot tapping rate* [UH03, Set07]. The natural rate of this tactus is usually in the general range of 1–3 Hz, corresponding to 60–180 BPM [Fra78, Par03]. It is important to note that the tactus is a perceptual concept: while it is determined by groups and accents of note events, the pulse of a tactus may or may not fall on a note event, and a note event may or may not fall on a pulse.

For segments of music with constant tempo, the tempo \mathfrak{T} in Beats per Minute (BPM) can be computed using the length of the segment Δt_s in seconds and the number of beats \mathcal{B} in the segment:

$$\mathfrak{T} = \frac{\mathcal{B} \cdot 60 \text{ s}}{\Delta t_s} \text{ [BPM].} \quad (9.1)$$

In the case of a dynamic tempo, the local tempo can be extracted by identifying the event time of every beat t_b and computing the distance between two neighboring beats with indices j and $j + 1$:

$$\mathfrak{T}_{\text{local}}(j) = \frac{60 \text{ s}}{t_b(j+1) - t_b(j)} \text{ [BPM].} \quad (9.2)$$

The inter-beat distance can also be pre-processed by a low-pass filter to smooth out potential inaccuracies in measuring the beat times.

Assigning a single, *overall* tempo to a piece of music is difficult when the tempo is not constant; in this case the mean tempo given in Eq. (9.1) does not necessarily match the *perceived overall tempo* a listener would indicate. This led Gabrielsson to distinguishing between the mean tempo and the *main tempo*, the latter being a measure ignoring slow beginnings or final *ritardandi* [Gab99]. Repp found good correlation of the perceived tempo with the mean value of a logarithmic IOI distribution [Rep94]. Goebel proposed a *mode tempo* which is computed by sweeping a window over the histogram of *Inter-Beat Intervals (IBIs)* and selecting the maximum position as mode tempo [GD01].

To complicate matters further, McKinney and Moelants argued that a single tempo does not sufficiently describe the (listener) group response when presented with a piece of music. They propose a representation of the overall tempo with two BPM values instead of a single one [MM04].

The *meter* is a regular alternation of strong and weak musical elements which are grouped with a length of normally three to seven beats or a length of around 5 s.

The metrical structure of music is typically hierarchical with higher hierarchical levels grouping events in lower hierarchical levels. Figure 9.2 visualizes different groupings depending on the time signature for two bars. Note that different time signatures might lead to different tempi even if the shortest event length, the *tatum*, is identical.

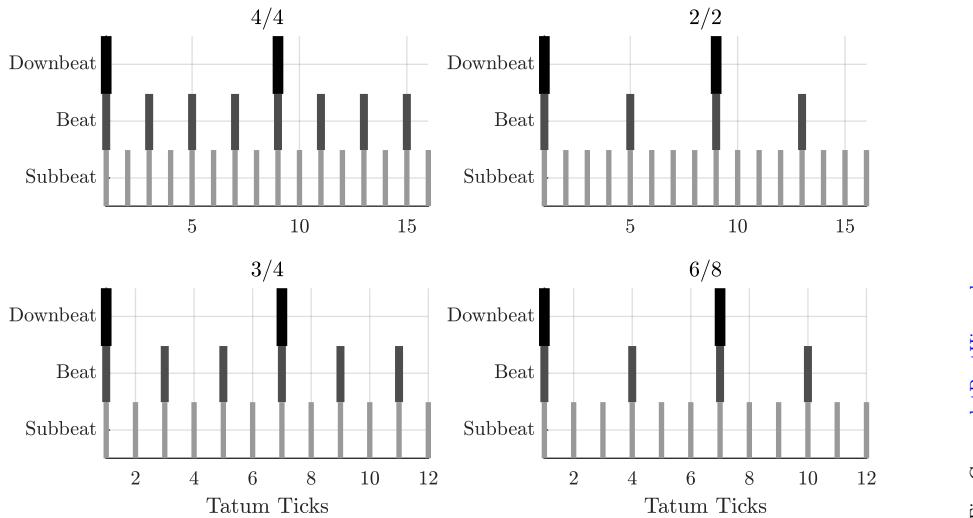


Fig. Gen.: plotBeatHierarchy.m

Figure 9.2: Different hierarchical levels related to tempo, beat, and meter.

9.1.3 Rhythm

The *rhythm* is specified by a pattern of a grouped series of events and durations [McA10]. The grouping properties allow a hierarchical segmentation into smaller subsequences forming different grouping levels. The length of the groups can range from the length of a few notes up to whole parts of the work defining musical form [LJ83].¹ Groups of a length between one beat and the length of the meter are most commonly referred to as rhythm. The rhythm is then defined by its event times, note and rest durations, and accents; if the durations of subsequent intervals relate to simple integer ratios, then the group usually has a closer binding than otherwise [Des92].

The various hierarchical levels of temporal grouping are an important property of many (western) pieces of music. Humans perceive pulses at different levels and with different tempi; at all levels the grouping of strong and weaker events occurs. The basic building block on the lowest (and shortest level) is commonly referred to as tatum [IBWW97], although other terms such as *atomic beat* have been used [HE02]. The tatum specifies the lowest period length or the period of the regular pulse train with the highest frequency represented in the music. Every rhythm is built of the tatus which can be interpreted as a rhythmic grid or a time quantization. The length of the highest level grouping depends on the definition of grouping and could go up to the level defining musical form such as the length of musical phrases or even longer structures which form groups.

9.1.4 Timing

The *timing* of individual notes or temporal events in a music performance does not necessarily exactly reflect the structural properties of the rhythm or meter but shows (minor) systematic temporal deviations from the underlying rhythmic structure [Sea38]. A detailed overview of expressive timing will be given in Chap. 16.

9.2 Representation of Temporal Events in Music

The representation of musical (temporal) events is closely related to the perception of such events for both terms and the musical score.

9.2.1 Tempo and Time Signature

Although the target tempo is sometimes given in a musical score, it is ultimately chosen by the performing artists. Tempo instructions for the performers became more and more explicit over the centuries. While many

¹However, we will use the term *rhythm* only for groups with a length of up to several beats.

$\frac{4}{4}$ $\frac{3}{4}$ $\frac{2}{4}$ $\frac{2}{2}$

Figure 9.3: Frequently used time signatures



Figure 9.4: Note values (top) and corresponding rest values (bottom) with decreasing length in musical score notation.

pieces from the Baroque period do not contain instructions due to the composer’s assumption that the tempo was specified by performance conventions, tempo indications started to occur more frequently in later epochs; Italian terms such as *Largo* (very slow), *Adagio* (slow), *Andante* (walking pace), *Moderato* (moderately), *Allegro* (fast), and *Presto* (very fast) are commonly used. The last century has seen more specific tempo indications in BPM.

The *local tempo* varies throughout a piece of music for nearly all genres. The possibilities to include instructions for such variations in the score are limited besides adding tempo indicators; examples of tempo instructions for sliding tempo changes are *ritardando* (slowing down) and *accelerando* (speeding up).

The *bar* (also called a *measure*) is the score equivalent of the (perceptual) meter. A score marks the beginning of each bar by a vertical line. The first beat of a bar usually has the highest (perceptual) weight and is referred to as *downbeat*.

The *time signature* is a way to convey information on the properties of a bar, namely the number of beats grouped together in one bar (upper part) as well as the note value constituting one beat (lower part of the time signature). The time signatures in Fig. 9.3 group four, three, two, and two beats, respectively. The fourth example differs from the first three in grouping half notes instead of quarter notes. The “denominator” of the time signature thus indicates the note value of one beat while its “numerator” indicates the number of beats per bar.²

9.2.2 Note Value

The *note value* defines the relative length of a note with respect to time signature and tempo. Notational convention requires that the sum of note values and rest values per bar (except a few special cases) must equal the numerator of the time signature. Thus, the absolute onset time of each note is specified by the bar index and the note’s position in the bar, given a specific tempo.

The offset time (also the *note off time*) is determined by the note’s onset time and its note value in the score but is not necessarily as clearly defined in a real-world performance. Sometimes a note value is shortened and a rest is appended to give the performing artists indications of the preferred articulation. In general, however, it is not unusual to place the responsibility for such articulation decisions on the performers rather than the musical score depending on epoch, style, and composer.

Figure 9.4 shows the most common note values (top) starting from a whole note and decreasing the value down to two sixty-fourth notes and the corresponding rests (bottom).

²There are exceptions from this rule such as a time signature $6/8$ with a beat length of three eighth notes.

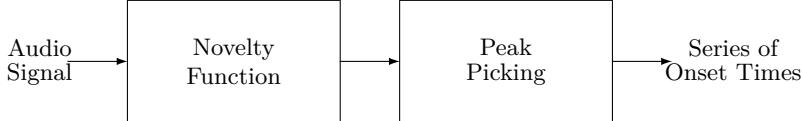


Figure 9.5: General flowchart of an onset detection system

9.3 Onset Detection

The goal of *onset detection* is to segment the audio stream into separate musical events. This supports tasks such as tempo detection or automatic music transcription and enables the analysis of rhythm and timing.

The flowchart of a typical *onset detection* system (also referred to as *onset tracking* system) is shown in Fig. 9.5. First, a so-called novelty function is computed. This novelty function represents a measure of the amount of “new” information in the audio signal for each analysis block and thus indicates the instantaneous likelihood of a musical event starting. Second, a *peak picking* process is applied to the novelty function to identify the locations of the significant maxima which can then be regarded as onset times.

Overview articles for different approaches to detecting onsets have been published by Bello et al. and Dixon [BDA⁺05, Dix06].

9.3.1 Novelty Function

An important property of the beginning of musical sound events is that “something new happens.” Thus, the first step toward automatic onset detection is the computation of a *novelty function* which indicates the amount of audio signal changes over time [Foo00]. Other names of this function are *detection function* [BDA⁺05] or *difference function* [HM03].

The first step in the computation of the *novelty function* is usually the calculation of the difference between current and preceding feature values. The result is then smoothed and negative values are discarded by applying HWR. The latter processing step is usually helpful for onset detection as an (amplitude or energy) increase might be expected at onset times while a decrease should make an onset less likely.

In one of the first publications on onset detection in (percussive) music signals, Schloss presented an algorithm that makes direct use of the audio signal’s envelope slope, extracted in the time domain [Sch85]. He extracts the envelope of the audio signal by computing the maximum of the magnitude of the signal within a block of samples and recommends to adjust the block length to the length of the period of the lowest frequency present. The envelope slope is then computed by using linear regression over several points of the peak amplitude.

As pointed out in Chap. 8, there exist different possibilities to extract the envelope of a signal, including taking the block’s maximum amplitude and low-pass filtering either the signal’s peak magnitudes or its RMS. An envelope-based analysis can work well for simple signals (e.g., drum loops); if the input signal is a mixture of multiple simultaneous sound sources these methods applied to the time-domain signal invariably fail. Although this challenge can be partly addressed by applying the envelope analysis on multiple frequency subbands instead of the broadband audio signal, methods utilizing STFT inputs have shown to be generally more reliable. The reason for this is that the STFT analysis allows to take into account spectral differences such as a pitch change in addition to amplitude and envelope changes. The disadvantage of the STFT-based computation is the comparably poor time resolution which affects the algorithm’s detection accuracy.

Modern systems generally use DNNs to extract the novelty function from a time-frequency input representation. Typical architectures include Recurrent Neural Networks (RNNs) [EBSG10, BAK12] and CNNs [SB14]. Traditional approaches, however, estimate the novelty from the differences between subsequent (overlapping) blocks of audio data, commonly in the frequency domain. This difference can either be computed with each individual spectral bin or with multiple bins grouped into frequency bands.

The frequency resolution of the input representation for onset detection systems increased over time. Early systems used one or a few frequency bands. For example, Scheirer used a filterbank of 6 bands basically

```
% difference spectrum (set first diff to zero)
afDeltaX = diff([X(:,1), X], 1, 2);

% half-wave rectification
afDeltaX(afDeltaX < 0) = 0;

% flux
d_flux = sqrt(sum(afDeltaX.^2))/size(X, 1);

```

(a) Matlab

```
d_flux = FeatureSpectralFlux(X, f_s)
return (d_flux)
```

(b) Python

Figure 9.6: Implementation of the half-wave rectified Flux fix python.

covering a one-octave range [Sch98], Klapuri proposed to use 21 non-overlapping bands with their band width and mid-frequency inspired by Zwicker's critical bands [Kla99], and Duxbury used 5 bands up to 22 kHz with constant Q [DSD02]. Zhou and Reiss proposed to use a filterbank of first-order complex resonators with an overall number of 960 frequency bands and 10 filters covering the range of a semi-tone, respectively [ZR07]. Nowadays, virtually all approaches use a STFT input representation or a similar time-frequency representation.

While spectral domain onset detection systems differ in the number of frequency bands they analyze, their main difference is the difference measure $d(n)$ between consecutive STFTs. The standard way to compute the difference is the spectral flux (see Sect. 3.6.8). However, since only magnitude increases (as opposed to decreases) are of interest, Eq. (3.66) is often modified to discard magnitude decreases by applying a Half-Wave Rectification (HWR) (compare Eq. 3.69) before summation:

$$d(n) = \frac{\sqrt{\sum_{k=0}^{\kappa/2} \text{HWR}(|X(k, n)| - |X(k, n-1)|)^2}}{\kappa/2 + 1} \quad (9.3)$$

Setting all negative intermediate results to 0 ensures that the novelty function does not confuse onsets and offsets. An implementation of this novelty function is shown in Fig. 9.6.

Many other variations and/or extensions of this STFT-based distance measure have been proposed. Some apply non-linear scaling to the magnitude values; Laroche proposed the use of applying a square root function to increase the impact of lower amplitudes [Lar03] and Hainsworth and Macleod calculated a logarithmic distance [HM03]. It is also possible to compute the distance with the cosine distance between two STFT frames as suggested by Foote [Foo00] or to use the distance between complex STFT bins [DBDS03]. Bello et al. pointed out that phase relations may be used for the detection of novelty in an audio stream as well [BMS03]. In particular, they suggest to apply the principles of instantaneous frequency computation (see Sect. B.6) and use the difference of the unwrapped phases.

A common problem of the distance measures above is that they are susceptible to frequency changes that do not mark the start of a musical event such as vibrato or portamento. Goto and Muraoka proposed to address this problem by taking into account not only the differences of magnitudes at the identical bin indices but also differences to neighboring magnitudes in the preceding STFT block [GM95]. This distance strongly depends on the parametrization, more specifically the ratio of STFT size and sample rate as well as the block overlap ratio.

Röbel proposed a transient detection that utilizes the COG of the instantaneous energy per frequency band [Rö05]. The derivation of time reassignment is closely related to frequency reassignment and is a way of virtually improving the time resolution.

Experience shows that the extracted function $d(n)$ often contains too much noise for reliable processing. Therefore, a smoothing filter is commonly applied to the function before further processing. Figure 9.7 shows the implementation of such post-processing with a MA-filter.

9.3.2 Peak Picking

As the goal of onset detection is the detection of events in time, these discrete time locations have to be extracted from the novelty function. Other downstream tasks such as tempo and rhythm extraction might also work on the novelty function itself instead of the extracted series of onsets.

The onsets are estimated by *peak picking* the novelty function. A perfect novelty function would indicate an onset at each local maximum. In reality, however, just detecting the locations of local maxima without

```
% novelty function
d = hNoveltyFunc(X, f_s);

% smooth novelty function
b = ones(iSmoothLpLen,1)/iSmoothLpLen;
d = filtfilt (b,1,d);
d(d<0) = 0;

% compute threshold
iLen = min(iThreshLpLen,floor(length(d)/3));
b = ones(iLen,1)/iLen;
G_T = .4*mean(d(2:end)) + filtfilt (b,1,d);

[fPeaks,iPeaks] = findpeaks(max(0,d-G_T));

```

(a) Matlab

```
# novelty function
d = hNoveltyFunc(X, f_s)

# smooth novelty function
b = np.ones(iLengthLp) / iLengthLp
d = filtfilt(b, 1, d)
d[d < 0] = 0

# compute threshold
G_T = .5 * np.mean(d[np.arange(1, d.shape[0])]) + filtfilt(b, 1, d)

# find local maxima above the threshold
iPeaks = find_peaks(d - G_T, height=0)
```

(b) Python

Figure 9.7: Implementation of Smoothing and Peak Picking the Novelty Function.

applying other constraints will likely cause a large number of falsely detected onsets (*FPs*, see Sect. 6.2.1). To suppress peaks of no interest, a threshold G_d is applied to the novelty function and only local maxima above this threshold are considered as onset time candidates. Practically, all local maxima after subtracting the threshold from the novelty function: $d(n) - G_d$ are considered as onsets.

In the simplest case, the threshold is a fixed threshold:

$$G_{d,c} = \lambda_1. \quad (9.4)$$

A potential problem with the fixed threshold is that the scale and shape of the novelty function can change considerably between different input audio signals. Therefore, an alternative to the fixed threshold is a signal-adaptive threshold. This adaptive threshold could be computed from the smoothed version of the novelty function. A typical smoothing filter is the MA filter:

$$G_{d,ma}(n) = \lambda_2 + \sum_{j=0}^{\mathcal{O}-1} b(j) \cdot d(n + \mathcal{O}/2 - j) \quad (9.5)$$

with $b(j)$ representing a user-defined window function. The parameter λ_2 shifts the threshold for adjusting the algorithm's sensitivity; low values of λ_2 will generally lead to high recall R while high λ_2 values will lead to high precision P . Alternatively, a *median filter* may replace the MA filter; its output $\hat{Q}_d(0.5)$ is an estimate of the median in the block of an appropriate length \mathcal{K} :

$$G_{d,me}(n) = \lambda_2 + \hat{Q}_d(0.5). \quad (9.6)$$

Figure 9.8 displays a simple amplitude-based novelty function, and adaptive threshold and the detected onset times for the first phrase of the jazz standard 'Summertime' played by a solo saxophone.

In addition to the thresholding process, other constraints have been proposed to improve peak picking reliability for onset detection. The confidence for a local maximum being an onset can, for example, be indicated by the "amplitude" distance between the local maximum and the preceding local minimum or the measurement of the novelty function's slope before the local maximum.

A typical problem with the raw output of untweaked onset detection systems is the occurrence of onsets with very close proximity in time. The reason for this can, on the one hand, be a polyphonic input signal with slightly asynchronous onset times supposed to be played synchronously or, on the other hand, simply a detection error of the system. For many tasks, onsets with a proximity of less than a minimum distance, e.g., 100 ms are not desirable, so that it is beneficial to combine two or more closely neighbored onsets into one. Simple ways to do so include (i) choosing the earliest onset, (ii) choosing the onset with the highest weight or confidence, or to (iii) computing some kind of average for the resulting (combined) onset time.

The final result of the onset detection system is a series of estimated onset times $\hat{t}_o(j)$, possibly including additional properties such as strength or confidence. The markers in Fig. 9.8 indicate the estimated onset positions, estimated with the thresholding shown in Fig. 9.7.

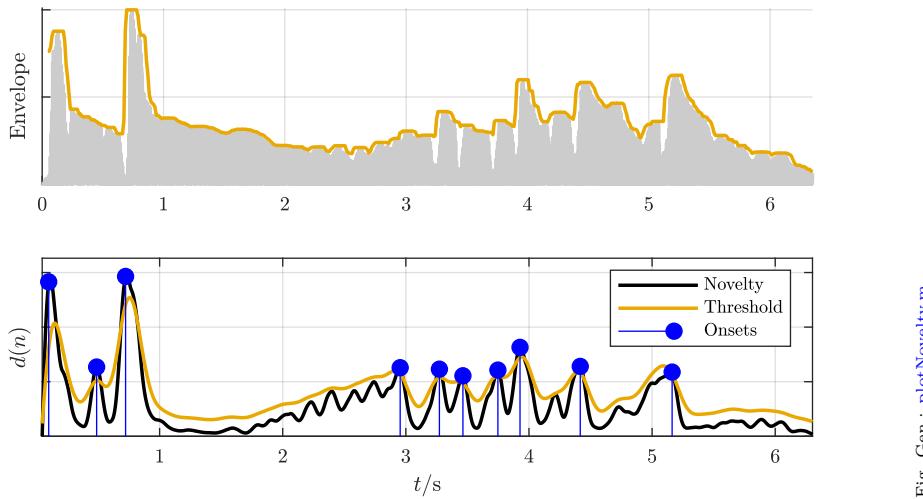


Fig. Gen.: plotNovelty.m

Figure 9.8: Audio signal and extracted envelope (top) and novelty function with example threshold and picked onsets (bottom).

9.3.3 Evaluation

In order to evaluate an onset detection system, a series of ground truth onset times has to be compared to the series of detected onset times. Besides potential issues with the labeling accuracy of annotators, a typical problem impacting both annotation and detection is the potential asynchrony of supposedly synchronous onsets: for example, different voices can be arpeggiated or a melody note might be slightly delayed with respect to the accompaniment for expressive reasons. It is methodically unclear whether these closely spaced onsets should count as individual events or should be combined into one onset, and, if the latter, how that combination should take place.

9.3.3.1 Metrics

The predicted onset times have to be compared with the reference onset times as given by the ground truth. Commonly, classification metrics are used to evaluate onset detection systems. In order to do that, predicted onsets within a tolerance interval around each ground truth interval are considered as correct, while others are incorrect. Typical tolerance intervals range from ± 25 ms to ± 50 ms.

Two possible errors can occur: a FN indicating that no onset is detected at the time of a reference onset, and a FP which is an onset that is wrongly detected where no reference onset is found (compare Sect. 6.2.1). Note that there are also borderline cases that have to be addressed specifically, such as two predicted onsets within one tolerance window. Also note that this task is not exactly a classification task, as the number of “Negatives” is unknown; thus, there is no way to count the TNs without discretizing the time axis. This makes it impossible to compute the accuracy, however, precision, recall, and f-measure can still be computed (compare Sect. 6.2.1) and are considered standard metrics for onset detection [YD07, LE07, EBSG10, SB14].

Several other measurements of detection performance have been proposed in the past utilizing the total number of detections ($TP + FP$) and the total number of reference onsets ($TP + FN$). For example, Liu et al. proposed [LGWM03]:

$$q_{liu} = \frac{\max((TP + FP), (TP + FN)) - (FN + FP)}{\max((TP + FP), (TP + FN))}, \quad (9.7)$$

however, this measure can—at least theoretically—be negative.

Most of the metrics mentioned above give the same weight to both FPs and FNs. This is not necessarily desirable, as sometimes a higher precision can be preferable over a high recall or vice versa. This can be adjusted through the *Receiver Operating Curve (ROC)*, where each data point represents a specific parametrization (e.g., the shift of the onset detection threshold) and the TPR is plotted over the FPR. It is obvious that the TPR can

be expected to increase with the FPR, however, this relation is not necessarily linear. Therefore, identifying the ‘working point’ in the ROC with the desired ratio of TPR to FPR leads back to the desired parametrization.

The evaluation of an onset detection system might also consider the timing accuracy of the system instead of simply matching onsets for evaluation. Such a measure or proximity takes into account the time distance $\Delta t_o(j) = t_o(j) - \hat{t}_o(j)$ between the reference and the detected onset time, which is ideally 0. From the distribution of the resulting time differences, multiple metrics can be computed. First, the arithmetic mean of the time distances

$$d_{\text{mean}} = \sum_{\forall j} \Delta t_o(j) \quad (9.8)$$

indicates the tendency of the system detecting onsets systematically too early or too late. Second, MAE and the MSE, see Eqs. (6.6) and (6.7) are measures of the average time distance between detected and ground truth onset. The standard deviation (compare Eq. (3.20))

$$\sigma_d = \sqrt{\frac{1}{(TP + FP)} \sum_{\forall j} (\Delta t_o(j) - d_{\text{mean}})^2} \quad (9.9)$$

is a measure of how broadly spread out the deviations are, and the absolute maximum deviation

$$d_{\text{max}} = \max_{\forall j} |\Delta t_o(j)| \quad (9.10)$$

gives a worst case estimate for the system. Note that deviation-based metrics can only be computed for TPs, thus they are not meant to replace but only complement the classification-based metrics listed above.

9.3.3.2 Datasets

The common datasets for training and evaluating onset detection performance are usually small, as the manual annotation of all onset times is a tedious and time-consuming task [Rep92, LDR04]. Alternatively, it is possible to use acoustic recordings with a symbolic trigger (such as recordings of the Yamaha Disklavier) or audio data synthesized from symbolic data, with the potential problem that the data might not generalize or not reflect real-world characteristics, respectively.

For these reasons, general datasets are scarce, and many studies fall back on an old dataset [LDR04]. Other datasets are either from monophonic sources [HSL13, CL14], only annotate specific onsets such as drum hits [GR06, MJ13, DG14, SWLH17], or restrict the instrumentation to piano [HSR⁺19].

9.3.3.3 Results

Typical errors of onset detection systems are, for example, FPs that occur due to amplitude or frequency changes without a new musical event (tremolo and vibrato) or FNs occurring due to other voices masking an onset. Results for state of the art systems largely depend on the complexity of the input data and how optimized the system is for specific input audio characteristics. Thus, the results of state-of-the-art systems have F-Measure ranges from roughly 60% – 90%.³

9.4 Beat Histogram

The *beat histogram* or *beat spectrum* is a not an analysis task itself but a representation that captures some rhythmic properties of the signal. However, since it is usually computed from the novelty function introduced in the previous section, it is mentioned in this place.

Similar to the “normal” magnitude spectrum, the x-axis of the beat histogram is a frequency (in this case with the unit BPM) and its y-axis represents some form of magnitude (beat strength). Peaks in the histogram

³https://nema.lis.illinois.edu/nema_out/mirex2018/results/aod/resultsperclass.html, last retrieved on Aug. 10, 2021.

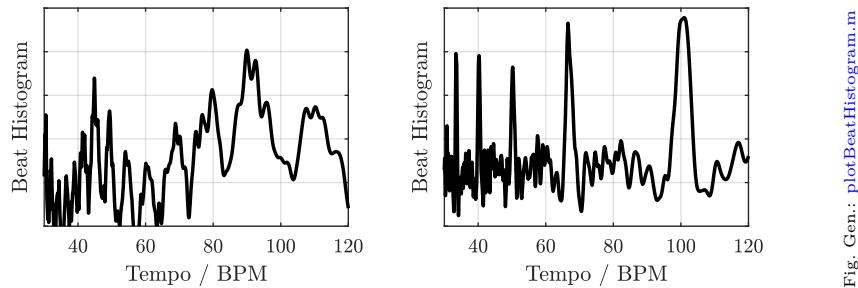


Fig. Gen.: plotBeatHistogram.m

Figure 9.9: Beat histogram of a string quartet performance (left) and of a piece of popular music (right).

should therefore correspond to the main tactus and its integer multipliers and divisors. The beat histogram can be interpreted as a frequency domain representation of the novelty function.

In contrast to a magnitude spectrum computed from the audio signal, a beat histogram focuses only on the rhythmic content. Therefore, it is computed from a novelty function with a much longer time window than an audio transform. While this histogram representation does not visualize the order of events in time, it is a good visualization of the rhythmic periodicities in the audio signal. Each peak indicates a periodic component of the signal. The higher the peaks are, the stronger this periodic component is in the signal, and the strongest peaks have been suggested to indicate the dominant tempo of the audio signal [Tza05].

There are multiple ways of computing such a beat histogram.

Scheirer applied a closely spaced filterbank of comb resonance filters to multiple novelty functions and used the filter's output energy as indication of the *beat strength* [Sch98]. A similar system based on comb resonance filters has also been used by Klapuri [Kla03b].

Foote and Uchihashi proposed to construct a similarity matrix (compare Sect. 9.7.1) from the cosine distance between all pairs of STFTs from the audio file and then derive the beat histogram by summing the similarity matrix along its diagonal [FU01].

Tzanetakis and Cook split the audio signal into four octave bands and extract the envelope per band by applying four processing steps [TC02], namely Full-Wave Rectification (FWR) by computing the absolute value, envelope smoothing by low-pass filtering, down-sampling to reduce the complexity, and DC removal by subtracting the arithmetic mean. An ACF (with special harmonic processing) is then computed in order to identify (rhythmic) envelope regularities. The beat histogram is construed by taking three peaks in the search range and adding their amplitude to the beat histogram, a process applied to each texture window.

These examples indicate that a beat histogram can be computed from various novelty functions and frequency transforms.

Figure 9.9 visualizes the beat histogram of an excerpt of popular music (right) in comparison with the histogram extracted from a string quartet performance (left). Note that for this plot, the beat histogram calculation is based on a very simple novelty function derived in the time domain from the signal's magnitude. After computing the ACF, the amplitude of each individual lag is mapped into the beat domain as the beat strength (this leads to a high resolution at low frequencies and a low resolution at high frequencies). The beat histogram computed from the popular music example has clearly defined peaks at integer ratios of each other; such a pattern is not identifiable in the beat histogram computed from a string quartet recording.

9.4.1 Beat Histogram Features

Similar to an audio magnitude spectrum, the beat histogram can be represented by various features. Examples of such features have been introduced by Tzanetakis and Cook [TC02]:

- the overall sum of the histogram,
- the relative amplitude of the highest peak,
- the relative amplitude of the second highest peak,

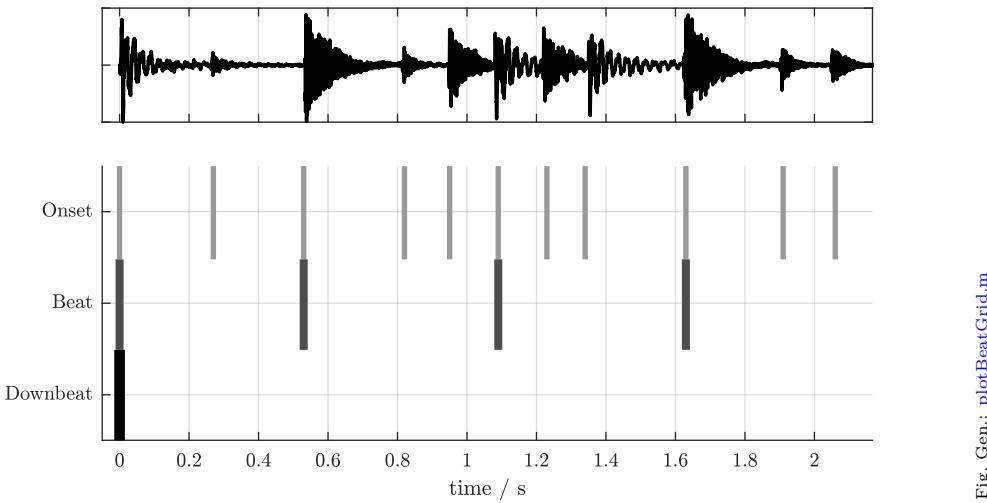


Figure 9.10: Visualization of Onset, Beat, and Downbeat times for a drum loop (top) of 4 beats length.

- the amplitude ratio of second highest to highest peak, and
- the BPM frequencies of the highest and second highest peak.

Burred and Lerch evaluated a feature set including statistical features of the beat histogram such as its arithmetic mean, standard deviation, kurtosis, skewness, and entropy; they additionally used a measure for what they called *rhythmic regularity*. The rhythmic regularity is a measure of how much the computed ACF differs from the linear weighting function of a block-wise ACF [BL04]. A study of all features above plus additional low level features found it difficult to identify single powerful beat histogram features; rather, the combination of features resulted in higher discriminative power when using the features for music genre classification [LL15].

9.5 Detection of Tempo and Beat Phase

Tempo in music is created by an underlying pulse with *beats* at regular intervals as introduced in Sect. 9.1.2. This pulse can be visualized as a beat grid structuring the audio signal. While an onset is a single event that can be detected without much context, the beat only exists as an event in the context of this pulse. Sometimes the terms onset and beat are confused as they both mark a point in time; it is important to note that while an onset may coincide with a beat (and that is likely to happen), there can be both onsets at time positions without beats and beats at positions without onsets. Figure 9.10 visualizes onset, beat, and downbeat positions for an example drum loop (length: 1 bar).

Knowledge of the tempo and the beat grid enables multiple different applications. On the one hand, they can be used as a feature or side information for various ACA tasks. For example, tempo might be one indicator for music genre or emotion, providing important information for the automatic classification. The knowledge of the beat grid has also been successfully used as pre-segmentation info for chord or structure detection. On the other hand, the tempo and beat grid can also be of direct use to a user. To give two examples from the DJ world: the DJ music database is often annotated with tempo labels in order to enable quick identification of songs with compatible tempi. More intelligent software is able to utilize the beat grid to synchronize and mix two or more pieces of music with different musical content (but the same tempo) at their beat positions to generate a so-called *mash-up* [DHYG14, VVDB18]. Without knowledge of the exact beat positions it would not be possible to mix those pieces in a musically meaningful way. A beat grid also allows to align a score representation with an audio file of the same piece of music, allowing, for example, human-computer performances or the comparative analysis of music performance.

While the tempo can be directly inferred from the beat grid as the distance between two neighboring beats (also referred to as *beat period* or *IBI*) can be directly converted into the local tempo (compare Eq. (9.2)), the

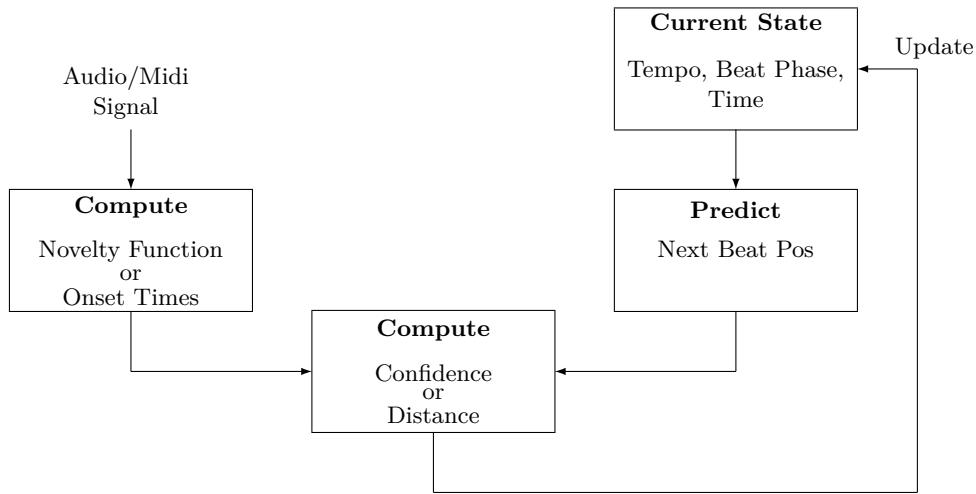


Figure 9.11: Flow chart of a standard beat-tracking system.

absolute grid locations are not strictly necessary to estimate the tempo, only their distance to each other.

Traditional systems for *tempo detection* (also referred to as *tempo induction* or *tempo tracking*) usually compute some kind of novelty function in a first processing step. Under the assumption that musical events are both stronger and more frequent on-beat than off-beat, the distance between beats is detectable by extracting the periodicity of the novelty function at the beat level. Note that the detection of specific beat positions is not necessarily required for detecting the tempo itself as the tempo is based on the *distance* between the beats rather than their absolute position. Typical examples for such periodicity analysis on the novelty function are (i) detecting the maximum of the beat histogram [Sch98, Kla03b, Tza05] or the maximum of an IOI histogram [Dix99], (ii) detecting the position of the ACF maximum as indicator of the inter-beat-distance [GH03], and (iii) finding the highest value of CCFs between the novelty function and a set of quantized template delta pulses for various tempi [Lar03]. Note that both the beat histogram as well as tempo template approaches only yield results for a quantized set of pre-defined tempi.

The tempo indicates the distance of the beats while their absolute position is often referred to as the *beat phase*. Both are needed to create a beat grid. Systems estimating both characteristics simultaneously are often referred to as *beat tracking* systems (although some reserve this phrase for real-time systems only).

Early beat tracking systems focused on extracting the beats from symbolic data such as MIDI files. This allows for accurate onset times as input, avoiding any error propagation from computing the novelty function or the peak picking phase [AD90, Lar95]. The basic approaches, however, are often similar in both MIDI and audio domains.

Many systems for beat tracking have in common an adaptive approach. This recurrent approach is shown in Fig. 9.11. After starting with an initial assumption of tempo and beat phase, the next beat(s) are predicted based on this assumption, followed by the computation of a confidence of the predicted beat location to be correct based on a comparison to either the novelty function or the series of onset times. Finally, the estimated tempo and beat phase are adapted accordingly for the prediction of the next beat. One big advantage of this adaptive approach is that the system should be able to follow moderate tempo changes over time. Large, for example, used an oscillator for generating pulses; the oscillators beat period and beat phase are constantly adapted for a better match to the onset times [Lar95]. The adaption speed is based on the distance between the estimated beat position and the actual onset position.

If the initial estimate of the tempo and tempo and beat phase is not close to the actual values, a recurrent system may fail or take a long time to converge to the correct values. One way of addressing this issue is to run multiple agents with different parametrizations in parallel [RGM94, GM95]. Each agent has its own hypothesis of the tempo and the beat phase and computes its own reliability by measuring the coincidence of the estimated beat positions with the (extracted) onset positions. The agent with the highest reliability is chosen as the one

providing the most likely tempo and beat phase estimate. This approach can also be extended by detecting changes in the tonal components to get additional information on the salience of onsets and estimated beats [GM96, GM99]. Similarly, Dixon chooses between multiple agents by measuring the regularity of the IBIs and the salience of the chosen onsets [Dix00].

In addition to onset times, beat tracking systems have also been shown to benefit from additional side information such as intensity and pitch [DC00, Meu02].

While most of the presented approaches above are (at least theoretically) capable of real-time processing, other approaches attempt to improve the robustness of tempo and beat detection by applying path searching or Dynamic Programming (DP)⁴ methods. For example, Allen and Dannenberg propose a beam search to select the most likely beat phase hypothesis [AD90]. Laroche used the most salient 10 to 15 maxima of the CCF between the novelty function and a set of quantized template delta pulses are used as initial tempo candidates. He then applies *DP* techniques to find the most likely overall path through all the tempo candidates over the whole audio file [Lar03]. A similar approach has been published by Peeters [Pee05]. He adds three different meter templates as possible states to the tempo candidates and finds the most likely tempo path through the audio file with the *Viterbi algorithm*. The non-causal algorithm based on dynamic programming have the advantage of that by looking at the optimal global solution for locally changing tempi allows them to deal with changes in tempo [Ell07].

Most modern, DNN-based approaches are based on recurrent network architectures. There are indications that targeting the prediction of more than one tempo-related parameter improves network performance, such as estimating beats and downbeats jointly [BKW16] or predicting both beats and tempo in a multi-task structure [BDK19].

9.5.1 Evaluation

Given usually one ground truth value per file, the evaluation of a tempo detection systems is straight-forward as long as the tempo stays constant. Some researchers make the case, however, that *perceptual* tempo—as opposed to tempo in music which is defined by the score—is not one value but a distribution with sometimes two salient maxima [MM06]. This would mean that an algorithm has to produce two tempo results which have to be evaluated either separately or in a weighted combination.

The evaluation of beat tracking and the resulting beat grid is somewhat more unclear, and this is reflected by the different custom metrics that have been proposed over time. The ambiguity introduced by the second perceptual tempo mentioned above poses additional questions about what an evaluation should output if, for example, all beats are correct but at half the ground truth rate.

9.5.1.1 Metrics

To evaluate the tempo detection, simple classification evaluation metrics such as the accuracy or the F-measure (see Sect. 6.2.1) can be computed by counting the correct and incorrect estimates given a tolerance around the ground truth (sometimes 4%, sometimes ± 1 BPM). The evaluation might also take into account the octave errors are graceful errors and calculate a second accuracy metrics counting double, triple, half and third tempi as correct as well [GKD⁺06].

The F-measure is also a useful standard metric for evaluating the beat grid, assuming a tolerance window around each beat. This tolerance window can be a fixed time (e.g., Holzapfel suggests ± 70 ms [HDZ⁺12]) or tempo-dependent (e.g., Dixon suggests a fraction of the IBI [Dix07]). The following other metrics have been proposed:

- Cemgil et al. proposed weighting the distance $\Delta t_{R,D}$ between reference and detected time with a Gaussian window function $w(\Delta t)$ [CKDH01]:

$$q_{\text{cemgil},2} = \frac{\sum_{\forall r} \max_{\forall t} w(\Delta t_{R,D})}{((TP + FN) + (TP + FP))/2}. \quad (9.11)$$

⁴Compare the Viterbi algorithm and DTW for examples of DP in this manuscript.

- McKinney et al. propose the P-score, computing a binary match over the quantized time, normalized by the maximum of the number of detected and the number of ground truth beats, and averaging the results over multiple annotators [MMDK07].
- The *Information Gain* of a normalized beat error histogram over a uniform distribution, computed as the Kullback-Leibler Divergence between these two distributions [HDZ⁺12].

Davies and Böck compare these and more metrics with subjective ratings [DB14]. While largely inconclusive, the authors warn against only using the F-measure or the P-score and encourage the use of additional metrics such as the information gain.

9.5.1.2 Datasets

While a beat detection dataset can also serve as data for the evaluation of tempo detection, this is not true the other way around. Most datasets focusing on tempo-only assume constant tempo and provide only one tempo value per song (snippet). Examples for such datasets are Extended Ballroom [MP16] consisting of dance music, GiantSteps [SM18] focusing on electronic dance music, and GTZAN-Tempo [PT14], containing song snippets from a variety of genres. Beat Annotations are available for, e.g., Ballroom [GKD⁺06] and SMC [HDZ⁺12].

9.5.1.3 Results

Due to the hierarchical level of rhythmic and structural properties of music, the most common errors of beat tracking systems are detecting the half or the double tempo, so-called octave errors. Obviously, this also depends on the tempo range that the system is configured to detect; the smaller the target range, the less likely are octave errors.

The variety of testing data leads to large variations of results for different datasets. Tempo detection accuracy of state of the art system tends to vary roughly between 60% and 90%, but tends to be higher than 90% if octave errors are ignored [BDK19].

The biggest challenges to most tempo tracking systems are varying tempi and files without strong and easily identifiable onsets. In these cases, F-Measures around or even below 50% can be considered good.⁵

9.6 Detection of Meter and Downbeat

The relation of *meter* and *downbeat* is very similar to the relation of tempo and beat phase. Just as the tempo is derived from the distance between two neighboring beats, the meter is (usually) the length of a bar while the downbeat marks the start time of a bar. Detecting the length of a musical bar also results in detecting the time signature when combined with beat information.

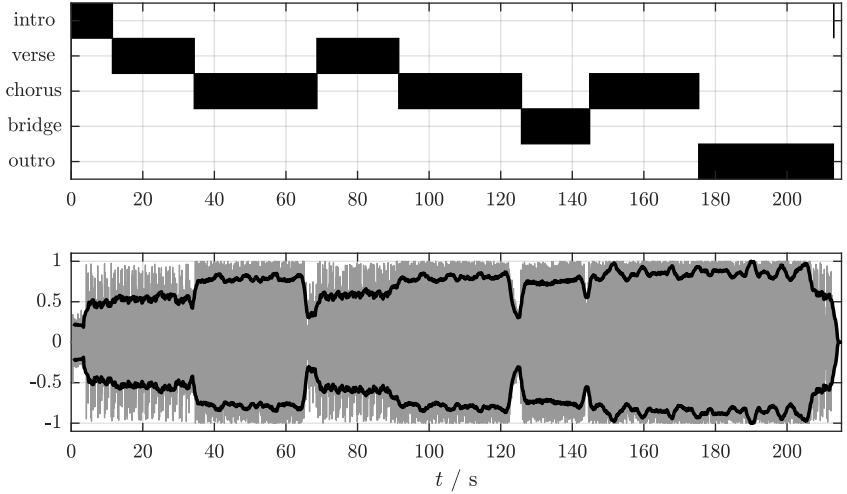
Downbeat and meter information can be helpful for refining beat tracking results. For example, the knowledge of downbeat positions can help to improve beat matching different songs. It also helps as a pre-processing step for tasks like structure detection as a measure is a fundamental building block of musical structure.

The hierarchical metrical structure of music makes the differentiation of detecting the beat and the downbeat basically a question of the hierarchical level to investigate: meter and downbeat detection requires a refocus on long-term periodicities, but the approaches are often similar with differences in the computation of the novelty function and the search range for periodicities.

Brown weighted the series of onsets with their IOI (in order to increase the impact of long notes) and computed the ACF of this series of weighted onsets to detect the meter [Bro93]. Toiviainen and Eerola proposed a similar approach and evaluated different weighting functions for the IOIs [TE06].

Uhle and Herre derived bar length candidates from integer multiples of a previously detected tatum and then computed the CCF of two snippets of the novelty function per frequency band to derive a measure of the likelihood of the individual bar length candidates [UH03].

⁵ compare the Audio Beat Tracking results of MIREX 2019 https://www.music-ir.org/mirex/wiki/2019:MIREX2019_Results, last accessed July 4, 2021

Fig. Gen.: `plotStructure.m`Figure 9.12: Example structure of a popular song (here: Pink's *So What*).

Escalona-Espinosa argued that it is not only the onset pattern itself that is of interest for the estimation of meter and downbeat but other features should be used for computing the novelty function as well [EE08]. More specifically, he assumed that in the western tradition of music (and even more so in the case of popular music) the position of a downbeat increases the likelihood of both (i) a note or harmony change and (ii) the occurrence of a new bass note compared to positions between downbeats. Therefore, he proposed the computation of two novelty functions, one based on the pitch chroma difference and the second on the bass energy increase. The time resolution for this computation is signal adaptive: it is the previously estimated tatum. Using the tatum has the two advantages of a signal-related segmentation and higher computational efficiency of the following processing steps. As an alternative to ACF-based approaches, he constructed two matrices which contain the (self-) similarity between all pairs of samples of the two novelty functions. The matrices are called self-similarity matrices. When averaging the diagonals of each similarity matrix, the result is a measure of periodicity with respect to the distance from the main diagonal. Depending on the similarity (or distance) measure used for computing the similarity matrix, this function can be closely related to the ACF. The lag of the main peak within a pre-defined search range is then the detected bar length. In combination with a tempo estimate, the result allows to derive the time signature of the piece of music. The most likely downbeat position is then estimated with the extracted bar length by computing the CCF of each novelty function with a delta pulse spaced with the bar length period. The lag of the CCF's maximum indicates the downbeat position.

9.7 Structure Detection

Similar to how a text book is has form on different levels (words are grouped into sentences, sentences are grouped into paragraphs, paragraphs are grouped into sections, and sections are grouped into chapters), music has structural groupings; it has been argued that form in music even plays a more pronounced role than in literature [Mac15]. We have already looked into different hierarchical levels of grouping above where tatum, tactus, and meter have been introduced. But there are many more large-scale structural elements in music. In Western classical music, bars are grouped into phrases, phrases are grouped into sentences or periods [Pro97].⁶ The highest grouping levels are usually those of a movement and, ultimately, the complete piece, for example, the symphony or sonata. Western popular music has similar fundamental building blocks (phrases) and combines them into segments such as chorus and verse. Figure 9.12 shows an example structure of a popular song. Studying the structure and form of music is a common task in Western music theory, but while the general structural elements are clear, the actual result of a structural analysis is subjective even if experts often consent. The potential ambiguity of musical structure stems from the fact that many different factors inform the listener

⁶Note that there are alternative terminologies commonly used as well.

of structure and structural boundaries. There is the perceptual grouping of note events based on rhythmic and melodic patterns, a harmony progression with cadences potentially marking structural boundaries, as well changes in key, tempo, or instrumentation marking new segments. Higher level characteristics like tension and relaxation, which combine rhythmic, harmonic, dynamic, melodic, and timbral elements can also impact the perception of musical structure [SH13]. Finally, there is also the performers' rendition of the piece which can reinforce or even shape the perception of structure in music through accents and tempo variations [Pal97]. Structure can be inferred from the audio signal by looking at three main characteristics [PMK10], (i) novelty and contrast, meaning something new and possibly unexpected happens, (ii) homogeneity, meaning that similar parts tend to be grouped together, and (iii) repetition, meaning that the recognition of a repeated segment indicates a structural segment. All of these characteristics can be represented through a variety of musical elements including but not limited to harmony, rhythm, melody, instrumentation, tempo, and dynamics. A structural boundary could, for example, be indicated by a instantaneous change of rhythmic patterns, dynamics, or instrumentation (high novelty). A segment in itself is often more homogeneous by itself than in contrast to other segments; this might be, for example, with respect to tonality, harmony, or tempo. The repetition of a sequence of bars indicates a potentially important segment.

Possible applications of structure detection can be found in visualization and navigation within a song; for example, a DJ may want to jump to specific segments in a song during a performance. Structural information can also be used for large-scale musicological analysis and might support other ACA tasks such as downbeat detection or even audio classification.

The objective of automatic structure detection is to reveal structural properties and relationships in a way that a list of musically meaningful, non-overlapping segment boundaries can be extracted. This list should also indicate segment repetitions and musical relations between segments. Note that the goal is not necessarily to assign labels such as 'verse' or 'chorus' to the detected parts, although some systems attempt that. An overview article on structure analysis was published by Nieto et al. [NMW⁺20]. Task related to structure detection are chorus detection [Got06] and audio thumbnail extraction [AS02].

9.7.1 Self Similarity Matrix

A common input representation of a structure detection system is the *Self Similarity Matrix (SSM)*. The SSM visualizes the similarity between different instants of a piece of music in an intuitive way. Given a feature sequence $v(n)$, the elements of similarity matrix \mathbf{S} can be computed by

$$S(n_A, n_B) = s(v(n_A), v(n_B)) \quad (9.12)$$

with s being a measure of similarity, e.g., the cosine similarity.

Both axes of the resulting SSM indicate time. The SSM is symmetric, as the similarity between (n_A, n_B) is identical to the similarity between (n_B, n_A) (for symmetric similarity measures). Figure 9.13 shows the SSM for a popular song as an example. Note that while structural details can be seen for this real-world recording, analyzing it with the simple approaches presented in this chapter will only have limited success. The red diagonal indicates the maximum of self-similarity ($n_A = n_B$). Blocks of constant red color indicate areas of high homogeneity such a held chord or a short repeated pattern, and blue vertical or horizontal lines indicate low similarity to all other points (often rests and pauses). Lines parallel to the diagonal indicate repetitions (the distance to the diagonal is the distance of the repeated segment). Analyzing Fig. 9.13 more closely, the following structural indicators can be identified visually: the intro stops at about 19 s, the bridge at 60–69 s is followed by a chorus (69–86 s) and the same constellation is repeated (as indicated by the line parallel to the diagonal) at 120 s, the high homogeneity of the instrumental is indicated by the red box from 145–170 s, and the songs ends with four repetitions of the chorus starting at 180 s visualized by the high similarity with diagonal structure in the end. Note that the above-mentioned lines parallel to the diagonal cannot be observed for recordings with varying tempo. While repetitions which replicate tempo exactly will result in such parallel lines, repetitions with different tempo will "bend" these lines slightly up or down.

The choice of the input feature sequence depends on the musical characteristic being evaluated for similarity. The example in Fig. 9.13 has been computed with the pitch chroma, but the self-similarity can be computed

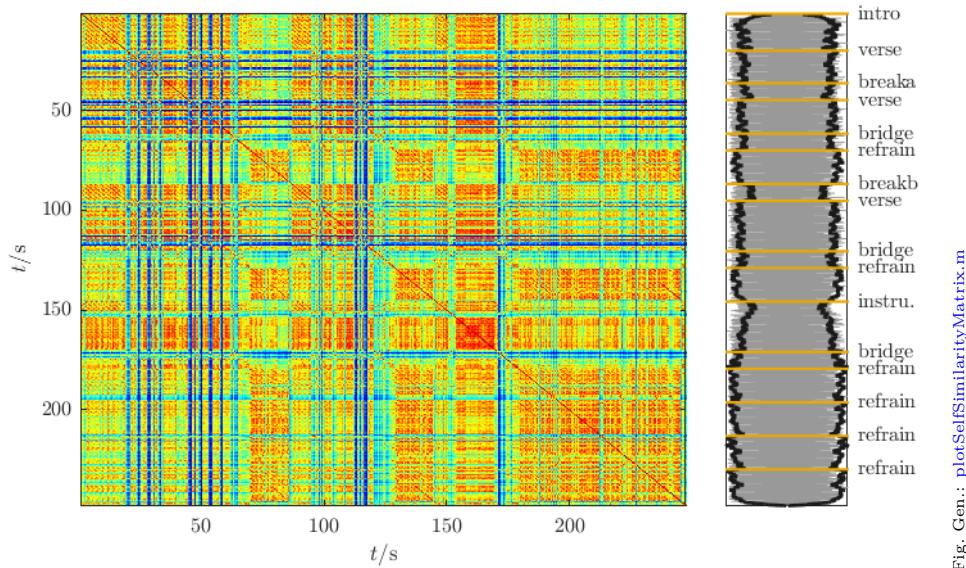


Figure 9.13: Self-Similarity Matrix of Michael Jackson’s *Bad*.

with any meaningful musical feature describing pitch and tonal content, rhythmic content, timbre content, or dynamics. Figure 9.14 displays the SSMs for the same piece but computed from different input representations, namely (from left to right) pitch chroma, RMS, MFCCs, and the Mel-spectrogram. Pre-processing aggregating the feature values over musically meaningful windows like beats can also be used to improve results. Overall, these three matrices look similar when we discard the skewed distributions of the similarities as indicated by the different average color. A closer look, however, reveals that some structural elements are more or less clearly highlighted by specific features. Ultimately, this is not surprising as, for example, verse and chorus might have the same level but different tonal content. It is a good reminder that there is no general objective measure of an overall music similarity; rather, the similarity has to be computed for specific, clearly defined characteristics.

The parametrization, especially the time resolution of the SSM can impact the usefulness of the matrix considerably. It is possible to compute multiple matrices with different time resolutions to find a balance between accurate timing and good overall detection, however, the timing accuracy of boundary detection might also be improved by utilizing other ACA tasks such as beat detection in a post-processing step.

The raw SSM is often subjected to post-processing. Common examples are (i) filtering, such as the application of a low pass smoothing filter or a high-pass edge detection filter, (ii) applying a nonlinear function to the matrix values, e.g., to decrease lower values or increase higher values, or (iii) quantizing the matrix values to a binary matrix by thresholding, thus enabling subsequent image processing techniques such as erosion and dilation [Soi04].

9.7.2 Approaches to Structure Detection

There are several ways to use the SSM to algorithmically infer the musical structure. As pointed out above, the main approaches revolve around the detection of novelty, homogeneity, and repetition.

9.7.2.1 Novelty analysis

The principles behind extraction of novelty from music signals have been introduced in the context of onset detection above (see Sect. 9.3). The differences of this structural analysis is that, on the one hand, significantly longer time windows have to be considered and that, on the other hand, many other low or high level features can be helpful for extracting the novelty.

A simple way of extracting a novelty function from the SSM is computed by sliding a checker board kernel as visualized in Fig. 9.15 along the matrix diagonal [Foo00]. The resulting novelty function is displayed in

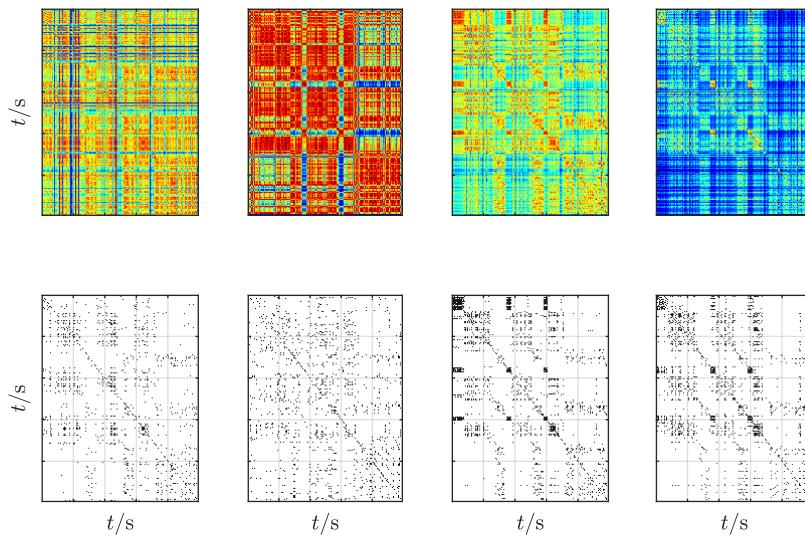


Figure 9.14: Self-Similarity Matrices of the same song based on different features (from left to right: pitch chroma, RMS, MFCCs, and Mel-spectrogram).

Fig. Gen.: plotSsmFeatures.m

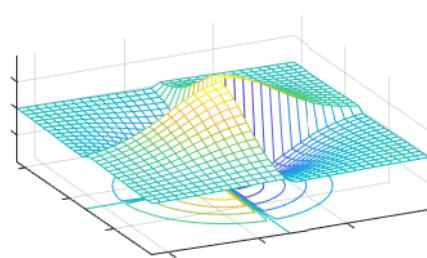


Fig. Gen.: plotCheckerBoard.m

Figure 9.15: Checker Board Filter Kernel with high-pass characteristics to detect novelty in a SSM.

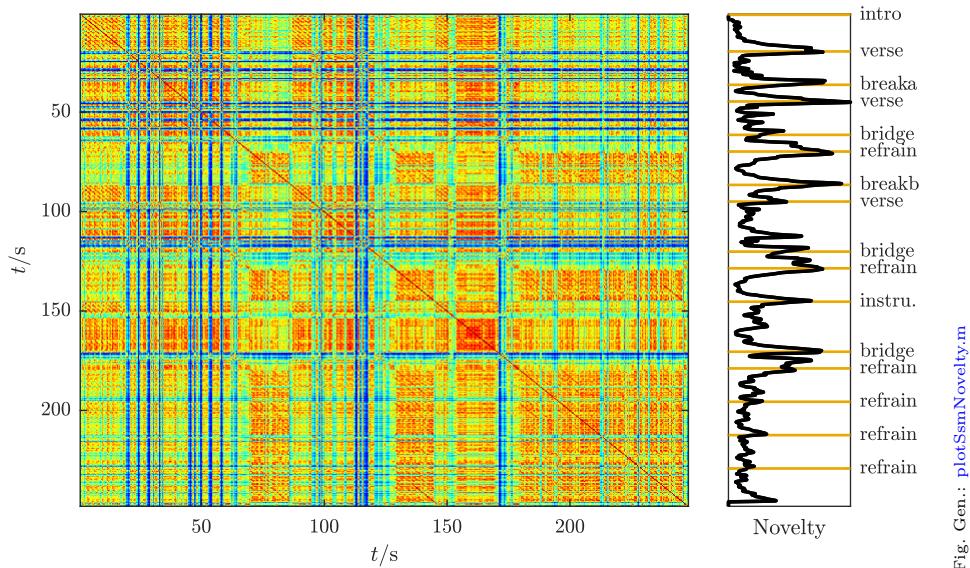
Fig. Gen.: [plotSsmNovelty.m](#)

Figure 9.16: Self Similarity Matrix (left) and extracted Novelty Function.

Fig. 9.16.

9.7.2.2 Homogeneity analysis

Detecting homogeneous regions in the SSM has led to a multitude of different proposals in the past. A naive approach might try to identify regions of small change by either segmenting the novelty function according to regions close to zero or detect plateaus in a low-pass filtered diagonal. The result of such a low-pass filter is shown in Fig. 9.17. As can be seen, this naive approach does not give convincing results with the chosen features and parameters.

Other approaches have successfully used clustering methods for both, detecting neighboring SSM entries [LS08] as well as identifying other segments that are repetitions [ME14].

9.7.2.3 Repetition analysis

Repetitions, indicated by lines parallel to the diagonal, might be found with edge detection techniques on the SSM [DG08]. A common approach to simplify the edge detection is to rotate half of the SSM in a way that one axis shows the distance from the diagonal as lag. All parallels to the diagonal of the original SSM will then be shown as vertical lines. Figure 9.18 shows such a rotated matrix (left) and the accompanying ground truth labels (right). Note that the rotated matrix in this example has not been extensively post-processed so the structure is not obvious in this example. The ground truth labels use different colors for different parts, and parsing the lag 0 from top to bottom visualizes directly the song structure. Parts which are not repeated (such as the intro or the instrumental) lack any vertical lines at lags larger than 0. Parts that are repeated, however, show vertical lines at the lag corresponding to the distance in time between the repetitions. Note that there is a certain redundancy to this visualization: for example, the four repetitions of the refrain at the end of the song appear as repetitions of the first refrain (at ≈ 70 s) with a lag of 120 s, but also as repetitions of the second refrain (at ≈ 130 s) with a lag of 60 s.

9.7.3 Evaluation

The formal evaluation of structure detection systems is more complicated than it seems at first glance, as human annotators—even if in agreement about the structure—tend to annotate different structural levels. For instance, what might just be the two segment structure (A) (B) for one annotator could easily be annotated

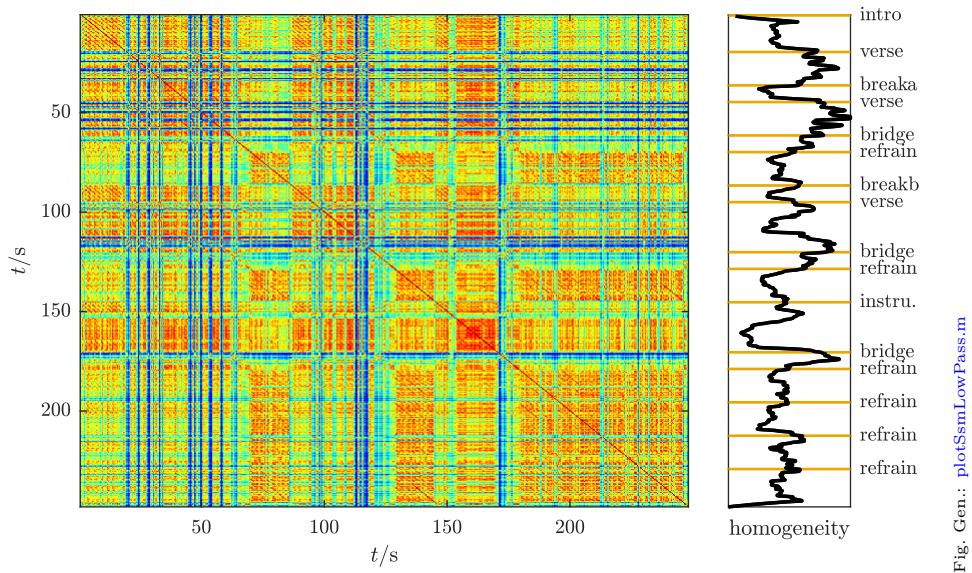


Figure 9.17: Self Similarity Matrix (left) and Low-pass filtered Diagonal as an estimate of the Homogeneity Function.

Fig. Gen.: `plotSsmLowPass.m`

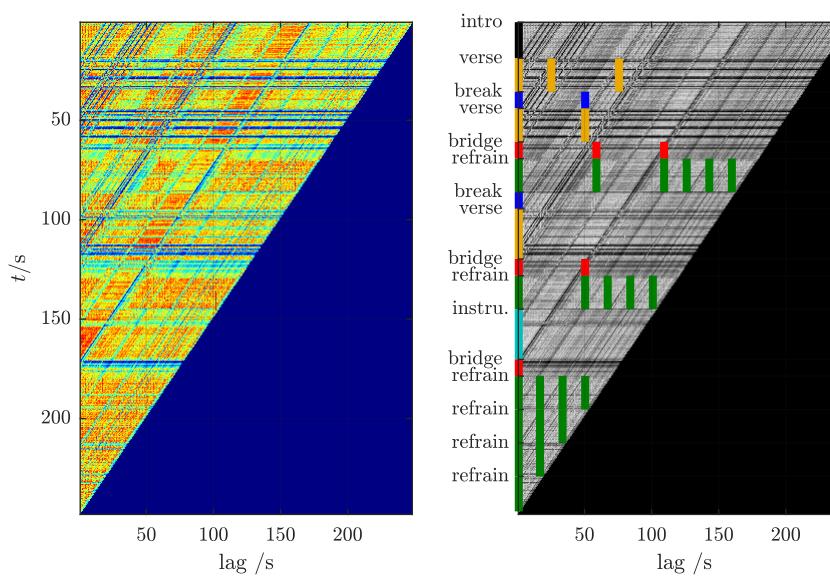


Figure 9.18: Rotated Self Similarity Matrix (left) and accompanying Ground truth (right) for detection of repetitions.

Fig. Gen.: `plotSsmRotated.m`

intro	A				A		outro		
intro	verse	chorus		verse	chorus		outro		
intro	V ₁	V ₂	C ₁	C ₂	V ₁	V ₂	C ₁	C ₂	outro

Table 9.1: Hypothetical example for annotator disagreements on musical structure

by another as (a b) (c d), which, given that both annotations are correct and the system might detect either one, makes an evaluation by simply matching the ground truth often impractical. Despite repeated proposals of addressing this problem [Kin16, MNFB17, KM21], there remain open questions how to best deal with this hierarchy problem [NMW⁺20].

To complicate matters more, the structure itself may be ambiguous. The various indicators of structural boundaries in a piece of music do not always necessarily lead to the same conclusion by listeners. There is also a simpler form of subjectivity involved: if the repetition of a chorus is modified in some way, is it still labeled as the chorus or not?

9.7.3.1 Metrics

The ground-truth complications with ambiguous, non-matching labels, confusion about hierarchical level, and annotator disagreement are impacting the choices of metrics for structure detection.

To evaluate the predicted boundaries, most commonly the F-measure, Eq. (6.5), is used. A predicted boundary is a TP if a ground truth boundary is within a certain tolerance (usually 0.5–3 s), a FP otherwise. Confusingly, this F-measure is often referred to as the *Hit Rate* [TLPG07].

If blockwise processing allows each block to be assigned to a segments, then the validity of these assigned labels can be evaluated with *Pairwise Clustering*. Again, this is an F-measure, with the difference that a TP is now counted when a pair of blocks with the same estimated labels matches is verified to have identical labels in the ground truth as well [LS08].

In addition, metrics based on conditional entropies have been proposed to measure over- and under-segmentation [Luk08].

9.7.3.2 Datasets

While there is a number of datasets available for structure detection, the availability of audio files is —similar to other tasks— often limited. The only datasets with publicly available audio data with segment annotations are the RWC dataset [Got02] and a small dataset called Sargon.⁷

For the remaining datasets, audio data have to be acquired in different ways. Note that even the same song may be published in different versions and masters, so all annotation data has to be re-verified by the researchers for that specific audio file. Frequently used datasets for segmentation include the Harmonix Set, the INRIA data [BSD⁺14], Isophonics [MCD⁺09], SPAM [NPB16], and SALAMI [SBF⁺11].

9.7.3.3 Results

Recent results are not easily available or comparable. Results from previous MIREX years indicate that the pair-wise F-Measure will —dependent on the evaluation data— be roughly in the range of 50 – 70% with large error margins.⁸

9.8 Exercises

9.8.1 Questions

1. Sort the following instruments according to their expected attack time: flute, harpsichord, trumpet.

⁷<https://github.com/urinieto/msaf-data/tree/master/Sargon/audio>, last retrieved Aug 12, 2021

⁸https://www.music-ir.org/mirex/wiki/2014:MIREX2014_Results, last access date Aug 12, 2021

2. What is the relation of tempo and the distance between two neighboring beat markers?
3. Assuming a constant tempo: if a piece of music has 20 beats in 10 secs, what is the tempo in BPM?
4. Given a time signature of 5/8, what is the number of sixteenth notes you could fit into this measure?
5. You have a rock song and a string quartet recording. Which of the signals do you expect to lead to stronger peaks in the beat histogram.
6. A beat spectrum shows a maximum at 3 Hz. What is the most likely tempo in BPM?
7. Describe the flowchart and processing steps of a prototypical onset detection system.
8. Consider an IOI histogram with a maximum at 400 ms. What is the most likely tempo in BPM?

9.8.2 Assignments

Chapter 10

Alignment

The alignment of two sequences results in a direct mapping of every time instant in one sequence to at least one time instant in the other sequence by minimizing some predefined alignment cost. Computing the alignment of two sequences makes most sense if the two sequences are similar yet not identical, such as two music performances of the same piece with varying tempo. The target mapping between two sequences could be visualized like in Fig. 10.1.

The results of the alignment can be used for various ACA tasks. They can, for example, be used for estimating the overall similarity between two audio sequences, or to generate an onset-synchronized score visualization after aligning an audio file with its musical score.

While sequence alignment algorithms are usually independent of the application domain, this chapter looks at music alignment, specifically audio-to-audio and audio-to-score alignment algorithms. This chapter first introduces a standard approach to sequence alignment (or path finding), the DTW algorithm, and then provides an overview of standard approaches to audio-to-audio alignment and audio-to-score alignment. The Viterbi algorithm (see Sect. 7.6.2) is a closely-related approach to path finding.

10.1 Dynamic Time Warping

The objective of *Dynamic Time Warping (DTW)* is to align or to synchronize two sequences of different length [SC78]. The algorithm assumes that both sequences start at their first index and end at their last index, which means it is not suitable for subsequence alignment in its default form. Furthermore, it assumes that a mapping sequence—the *path*—can not jump over multiple indices and that the mapped indices can never decrease. This section will first attempt to introduce DTW in an abstract way before providing a step-by-step example that should clarify the practical application.

Given two sequences $A(n_A)$ with $n_A \in [0; N_A - 1]$ and $B(n_B)$ with $n_B \in [0; N_B - 1]$, the first step is the computation of a pairwise distance between $A(n_A)$ and $B(n_B)$. This results in the *distance matrix* $D_{AB}(n_A, n_B)$.

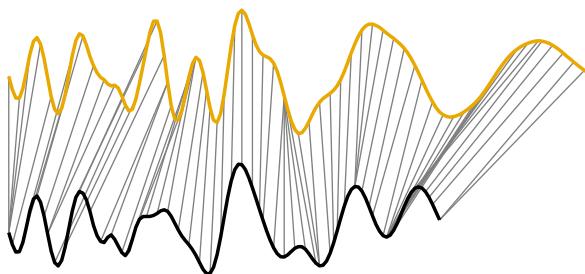


Fig. Gen.: plotSequenceAlignment.m

Figure 10.1: Visualization of the mapping of two similar sequences to each other after alignment.

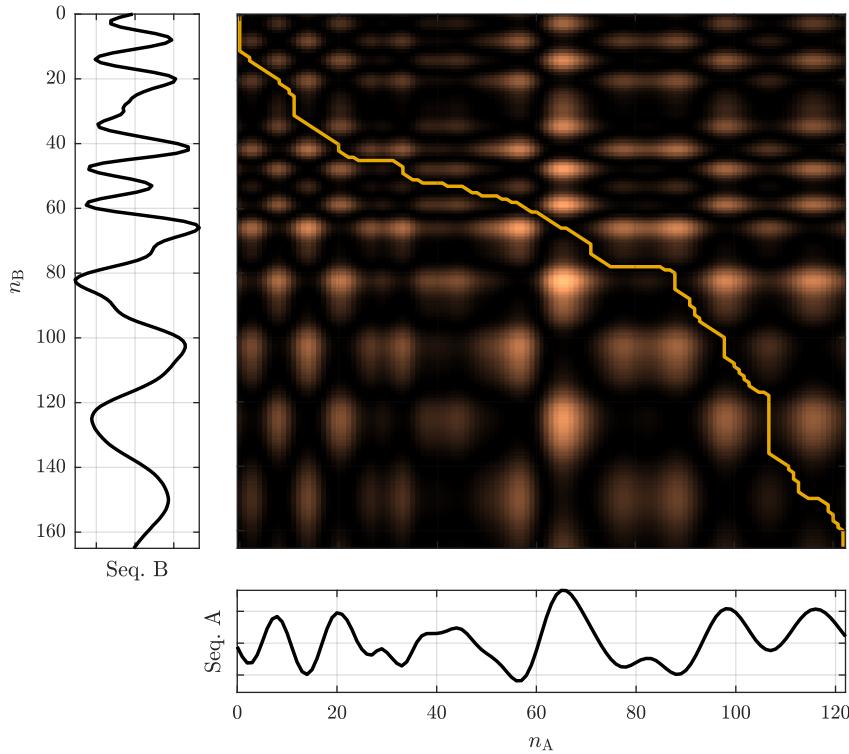
Fig. Gen.: `plotDtwPath.m`

Figure 10.2: Distance matrix and alignment path for two example sequences; dark entries indicate a small distance.

If the two sequences are identical, the result is very similar (albeit inverted) to the SSM introduce in Sect. 9.7.1.

The goal of the algorithm is to find the specific *alignment path* from $\mathbf{D}_{AB}(0, 0)$ (the start of both sequences) to $\mathbf{D}_{AB}(\mathcal{N}_A - 1, \mathcal{N}_B - 1)$ (the end of both sequences) which minimizes the overall accumulated distance. Minimizing the accumulated distance is thus the most likely alignment path (also *warping path*) between the two sequences. Figure 10.2 shows two example sequences, the corresponding distance matrix, and the resulting alignment path between the two sequences computed with standard DTW as described below.

The alignment path will be referred to as $\mathbf{p}(n_P)$ and $n_P \in [0; \mathcal{N}_P - 1]$; it is a direct measure of how one sequence has to be warped (scaled in time) to give the best fit to the other sequence. Each path entry is a matrix index in the range $([0; \mathcal{N}_A - 1], [0; \mathcal{N}_B - 1])$.

As indicated above, the following constraints apply to the path:

- *Boundaries*: the path has to start at the first index of both sequences and has to end at the end of both sequences, meaning that it covers both entire sequences from beginning to end:

$$\mathbf{p}(0) = [0, 0], \quad (10.1)$$

$$\mathbf{p}(\mathcal{N}_P - 1) = [\mathcal{N}_A - 1, \mathcal{N}_B - 1]. \quad (10.2)$$

- *Causality*: the path can only move forward through both sequences, meaning that it is not allowed to “go back in time”:

$$n_A|_{\mathbf{p}(n_P)} \leq n_A|_{\mathbf{p}(n_P+1)}, \quad (10.3)$$

$$n_B|_{\mathbf{p}(n_P)} \leq n_B|_{\mathbf{p}(n_P+1)}. \quad (10.4)$$

- *Continuity*: no index n_A or n_B can be omitted, meaning that the path is not allowed to jump through

either sequence:

$$n_A|_{\mathbf{p}(n_P+1)} \leq (n_A + 1)|_{\mathbf{p}(n_P)}, \quad (10.5)$$

$$n_B|_{\mathbf{p}(n_P+1)} \leq (n_B + 1)|_{\mathbf{p}(n_P)}. \quad (10.6)$$

These path restrictions result in a theoretical maximum path length of

$$\mathcal{N}_{P,\max} = \mathcal{N}_A + \mathcal{N}_B - 2 \quad (10.7)$$

when the path runs along the edges of the distance matrix¹ and a minimum path length of

$$\mathcal{N}_{P,\min} = \max(\mathcal{N}_A, \mathcal{N}_B) \quad (10.8)$$

when the path runs on the matrix diagonal for as long as possible.

In order to find the optimal alignment path the concept of “cost” is used. The cost of a path \mathbf{p}_j is determined by accumulating the values of the distance matrix at all path points:

$$\mathfrak{C}_{AB}(j) = \sum_{n_P=0}^{\mathcal{N}_P-1} \mathbf{D}(\mathbf{p}_j(n_P)). \quad (10.9)$$

The goal is, therefore, to identify the optimal alignment path which is minimizing the overall cost:

$$\mathfrak{C}_{AB,min} = \min_{\forall j} (\mathfrak{C}_{AB}(j)), \quad (10.10)$$

$$j_{opt} = \operatorname{argmin}_{\forall j} (\mathfrak{C}_{AB}(j)). \quad (10.11)$$

The optimal path can thus be found by computing all possible paths through the matrix \mathbf{D} and determining the path with the lowest overall cost, however, this can be very inefficient for long sequences.

DP techniques allow to avoid this brute force approach as the best global solution can be computed more efficiently. As an intermediate result, the *cost matrix* \mathbf{C}_{AB} is introduced; it has the same dimensions as the distance matrix, but each matrix element contains the accumulated overall cost of the best path to this specific matrix element.

The cost matrix can be computed iteratively by

$$\mathbf{C}_{AB}(n_A, n_B) = \mathbf{D}_{AB}(n_A, n_B) + \min \left\{ \begin{array}{l} \mathbf{C}_{AB}(n_A - 1, n_B - 1) \\ \mathbf{C}_{AB}(n_A - 1, n_B) \\ \mathbf{C}_{AB}(n_A, n_B - 1) \end{array} \right. . \quad (10.12)$$

Figure 10.3 plots both the distance matrix and the corresponding cost matrix for the example sequences from Fig. 10.2.

The first entry of the cost matrix is initialized with

$$\mathbf{C}_{AB}(0, 0) = \mathbf{D}_{AB}(0, 0). \quad (10.13)$$

Due to the alignment path restrictions given above the computation of both the first row and the first column of the cost matrix is trivial:

$$\mathbf{C}_{AB}(n_A, 0) = \mathbf{D}_{AB}(n_A, 0) + \mathbf{C}_{AB}(n_A - 1, 0), \quad (10.14)$$

$$\mathbf{C}_{AB}(0, n_B) = \mathbf{D}_{AB}(0, n_B) + \mathbf{C}_{AB}(0, n_B - 1). \quad (10.15)$$

This can also be interpreted as initializing the distance matrix for indices smaller than 0 with

$$\mathbf{C}_{AB}(n_A, -1) = \infty,$$

$$\mathbf{C}_{AB}(-1, n_B) = \infty,$$

¹The -2 originates in the automatic avoidance of the corner of the distance matrix.

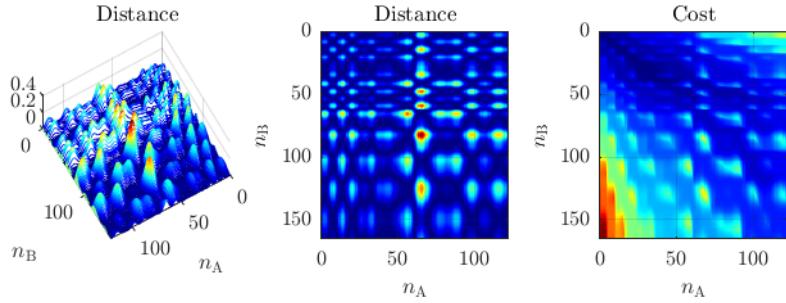


Fig. Gen.: plotDtwCost.m

Figure 10.3: Distance matrix (left and middle in two different visualizations) and corresponding cost matrix (right).

and applying Eq. (10.12).

Each cost matrix element contains the minimum cost to reach that element. But computing the cost alone is not sufficient. During the calculation of the cost matrix, the indices of the preceding cell that has been selected as the minimum cost for each matrix element have to be remembered. The optimal path can then be traced back from the any matrix element to the beginning.

The complete iterative algorithm can thus formally be summarized by

- *Initialization:*

$$\mathbf{C}_{AB}(0,0) = \mathbf{D}_{AB}(0,0), \quad (10.16)$$

$$\mathbf{C}_{AB}(n_A, -1) = \infty, \quad (10.17)$$

$$\mathbf{C}_{AB}(-1, n_B) = \infty. \quad (10.18)$$

- *Recursion:*

$$\mathbf{C}_{AB}(n_A, n_B) = \mathbf{D}_{AB}(n_A, n_B) + \min \begin{cases} \mathbf{C}_{AB}(n_A - 1, n_B - 1) \\ \mathbf{C}_{AB}(n_A - 1, n_B) \\ \mathbf{C}_{AB}(n_A, n_B - 1) \end{cases}, \quad (10.19)$$

$$j = \operatorname{argmin} \begin{cases} \mathbf{C}_{AB}(n_A - 1, n_B - 1) \\ \mathbf{C}_{AB}(n_A - 1, n_B) \\ \mathbf{C}_{AB}(n_A, n_B - 1) \end{cases}, \quad (10.20)$$

$$\Delta \mathbf{p}(n_A, n_B) = \begin{cases} [-1, -1] & \text{if } j = 0 \\ [-1, 0] & \text{if } j = 1 \\ [0, -1] & \text{if } j = 2 \end{cases}. \quad (10.21)$$

- *Termination:*

$$n_A = \mathcal{N}_A - 1 \wedge n_B = \mathcal{N}_B - 1. \quad (10.22)$$

- *Path backtracking:*

$$\mathbf{p}(n_P) = \mathbf{p}(n_P + 1) + \Delta \mathbf{p}(\mathbf{p}(n_P + 1)), \quad n_P = \mathcal{N}_P - 2, \mathcal{N}_P - 3, \dots, 0. \quad (10.23)$$

Figure 10.4 shows an example implementation of Eqs. 10.16 to 10.23.

It can happen that a matrix has more than one optimal path in the case the cost is identical for two different routes through the matrix. Note that the distance matrix is frequently be replaced by a *similarity matrix*; the algorithm will remain the same although the cost matrix (as introduced below) will have to be replaced by a *likelihood matrix* and some algorithmic details will have to be modified in other places as well, for instance, replacing the min operation by a max operation.

```
% init directions for back-tracking [diag, vert, hori]
iDec = [-1 -1; -1 0; 0 -1];

% cost initialization
C = zeros(size(D));
C(1,:) = cumsum(D(1,:));
C(:,1) = cumsum(D(:,1));

% traceback initialization
DeltaP = zeros(size(D));
DeltaP(1,2:end) = 3; % (0,-1)
DeltaP(2:end,1) = 2; % (-1,0)

% recursion
for (n_A = 2:size(D,1))
    for (n_B = 2:size(D,2))
        % find preceding min (diag, column, row)
        [fC_min, DeltaP(n_A,n_B)] = min([C(n_A-1,n_B-1), C(n_A-1,n_B), C(n_A,n_B-1)]);
        C(n_A, n_B) = D(n_A, n_B) + fC_min;
    end
end

% traceback
p = size(D); % start with the last element
n = [size(D,1), size(D,2)]; % [n_A, n_B];
while ((n(1) > 1) || (n(2) > 1))
    n = n + iDec(DeltaP(n(1),n(2)),:);

    % update path (final length unknown)
    p = [n; p];
end

```

(a) Matlab

```
# init directions for back-tracking [diag, vert, hori]
iDec = np.array([[-1, -1], [-1, 0], [0, -1]])

# cost initialization
C = np.zeros(D.shape)
C[0, :] = np.cumsum(D[0, :])
C[:, 0] = np.cumsum(D[:, 0])

# traceback initialization
DeltaP = np.zeros(D.shape, dtype=int)
DeltaP[0, :] = 2 # (0,-1)
DeltaP[:, 0] = 1 # (-1,0)
DeltaP[0, 0] = 0 # (-1,-1)

# recursion
for n_A in range(1, D.shape[0]):
    for n_B in range(1, D.shape[1]):
        # find preceding min (diag, column, row)
        DeltaP[n_A, n_B] = int(np.argmin([C[n_A-1, n_B-1], C[n_A-1, n_B], C[n_A, n_B-1]]))
        C[n_A, n_B] = D[n_A, n_B] + C[prevC_index]

# traceback init
p = np.asarray(D.shape, dtype=int) - 1 # start with last element
n = p

while (n[0] >= 0) or (n[1] >= 0):
    n = n + iDec[DeltaP[n[0], n[1]], :]

    # update path
    tmp = np.vstack([n, p])
    p = tmp

```

(b) Python

Figure 10.4: Implementation of DTW (left: Matlab, right: Python).

10.1.1 Example

A short example is presented here in order to allow a better understanding of the initially rather abstract concept of DTW. The two (one-dimensional) input sequences are

$$\begin{aligned} A &= [1, 2, 3, 0], \\ B &= [1, 0, 2, 3, 1], \end{aligned}$$

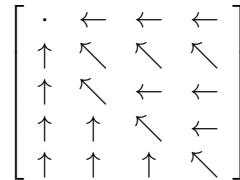
and we will simply use the magnitude of the difference as the distance measure. The distance matrix is then

$$D_{AB} = \begin{bmatrix} 0 & 1 & 2 & 1 \\ 1 & 2 & 3 & 0 \\ 1 & 0 & 1 & 2 \\ 2 & 1 & 0 & 3 \\ 0 & 1 & 2 & 1 \end{bmatrix}, \quad (10.24)$$

and the corresponding cost matrix is

$$C_{AB} = \begin{bmatrix} 0 & 1 & 3 & 4 \\ 1 & 2 & 4 & 3 \\ 2 & 1 & 2 & 4 \\ 4 & 1 & 1 & 4 \\ 4 & 3 & 3 & 2 \end{bmatrix}. \quad (10.25)$$

During the calculation of the cost matrix, we also memorized the direction of lowest cost for each matrix element:



in order to be able to backtrack the optimal path through the distance matrix:

$$\mathbf{D}_{AB} = \begin{bmatrix} 0 & 1 & 2 & 1 \\ 1 & 2 & 3 & 0 \\ 1 & 0 & 1 & 2 \\ 2 & 1 & 0 & 3 \\ 0 & 1 & 2 & 1 \end{bmatrix}. \quad (10.26)$$

10.1.2 Common Variants

The standard DTW algorithm as described above is frequently modified; the reasons for this modification can be either the adaption to specific use cases or the optimization of its workload requirements. Several such modifications are described in detail by Müller [Mü07].

In order to favor vertical, horizontal, or diagonal path movement, weighting factors can be applied to Eq. (10.12)

$$C_{AB}(n_A, n_B) = \min \begin{cases} C_{AB}(n_A - 1, n_B - 1) + \lambda_d \cdot D_{AB}(n_A, n_B) \\ C_{AB}(n_A - 1, n_B) + \lambda_v \cdot D_{AB}(n_A, n_B) \\ C_{AB}(n_A, n_B - 1) + \lambda_h \cdot D_{AB}(n_A, n_B) \end{cases}. \quad (10.27)$$

In the default DTW approach all these weights had been set to 1. Another typical set of weights is

$$\begin{aligned} \lambda_d &= 2, \\ \lambda_v &= 1, \\ \lambda_h &= 1 \end{aligned}$$

to prevent the implicit preference of the diagonal since one diagonal step corresponds to one horizontal plus one vertical step.

Instead of forcing the algorithm to only increment each index by one, it is conceivable to loosen this constraint to allow for larger step sizes or jumps. The arguments of the minimum operation of Eq. (10.12) could, for example, be replaced by

$$\min \begin{cases} C_{AB}(n_A - 1, n_B - 1) \\ C_{AB}(n_A - 2, n_B - 1) \\ C_{AB}(n_A - 1, n_B - 2) \end{cases} \quad (10.28)$$

which constrains the slope of the warping path to avoid the path containing many consecutive horizontal or vertical steps. This modification will only work if the sequence lengths N_A and N_B differ not by more than a factor of 2. Another alternative enforcing the alignment of all elements of both sequences is to replace the arguments of the minimum operation of Eq. (10.12) by

$$\min \begin{cases} C_{AB}(n_A - 1, n_B - 1) \\ C_{AB}(n_A - 2, n_B - 1) + D_{AB}(n_A - 1, n_B) \\ C_{AB}(n_A - 3, n_B - 1) + D_{AB}(n_A - 1, n_B) + D_{AB}(n_A - 2, n_B) \\ C_{AB}(n_A - 1, n_B - 2) + D_{AB}(n_A, n_B - 1) \\ C_{AB}(n_A - 1, n_B - 3) + D_{AB}(n_A, n_B - 1) + D_{AB}(n_A, n_B - 2) \end{cases}. \quad (10.29)$$

10.1.3 Optimizations

If the two sequences to be aligned are long, the size of the distance matrix increases drastically as the number of matrix elements is the multiplication of the length of the sequences $N_A \cdot N_B$. This results in both high memory requirements and a large overall number of operations. The effects can be alleviated by using different approaches to optimization.

One simple approach to reduce the amount of memory without changing the (standard DTW) algorithm or its results is to replace the cost matrix with two vectors, a row and a column vector of cost entries, as the cost

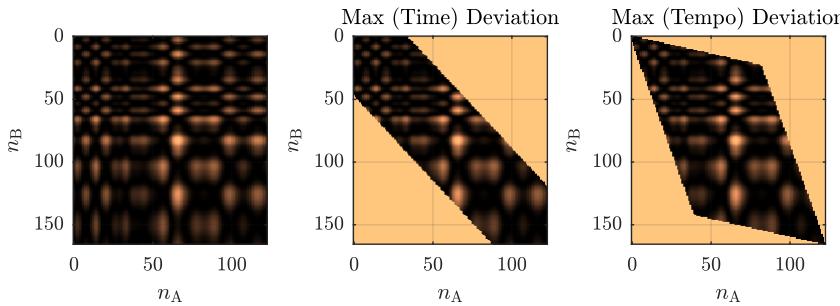
Fig. Gen.: `plotDtwConstraints.m`

Figure 10.5: Path restrictions for performance optimizations of DTW: original distance matrix (left), matrix with maximum time deviation constraint (middle) and matrix with maximum tempo deviation (right).

of distant previous elements is irrelevant. However, while this reduces the memory footprint somewhat, it does not reduce the workload.

Under the assumption that the timing of the two sequences to be aligned does not deviate more than a certain maximum deviation T_{\max} , neither the distances nor the cost has to be computed for matrix entries outside a band with the width of $2T_{\max}$ centered around the diagonal from start $(0, 0)$ to stop $(N_A - 1, N_B - 1)$. This optimization has first been proposed by Sakoe and Chiba [SC78] and is visualized in Fig. 10.5 (middle).

If the slope of the alignment path is constrained or, in other words, the tempo difference between the two sequences is limited then matrix entries outside of a trapezoid spanned by this tempo relationship do not have to be computed. This optimization has first been proposed by Itakura [Ita75] and is visualized in Fig. 10.5 (right).

If start and end points of the two sequences are not necessarily a perfect match, for example, in the case of silence frames at the beginning and end of a sequence, this approach should be combined with the optimization assuming a maximum time deviation (see above) to allow horizontal and vertical movement along the matrix edges at start and end. Otherwise the path restrictions might become too narrow if the sequences do not start and stop at exactly the same point. This is a problem of the path shown in Fig. 10.5 (right) at both start and end.

The DTW algorithm is not a real-time algorithm as it requires the complete sequences for processing. It is possible to use DTW in a pseudo-real-time context when the alignment path is only computed within a sliding local window [DW05]. This results in a high algorithmic latency. Outside this window the path will not be updated anymore. The number of operations depends on the window length and is independent of the sequence length.

Multi-scale DTW is based on the idea of processing the input sequences at different time resolution. More specifically, the time resolution of the input sequences can be reduced by both, using larger block sizes for computing the distances or low-pass filtering and down-sampling the existing sequences. The extracted path through this down-sampled distance matrix can then be used to define a sliding window to be used for a second matrix with finer resolution. This process can be done in two stages [TE03] or iteratively in multiple stages [MMK06, Mü07].

10.2 Audio-to-Audio Alignment

Audio-to-audio alignment describes the process of retrieving corresponding points in time in between two audio signals with the same or a similar content. It requires an analysis of the audio files enabling the mapping of points in time in one signal to points in time in the other signal. The knowledge of those synchronization points enables a variety of different use cases such as

- quick browsing for certain parts in recordings in order to easily compare parts auditorily [DW05, MMK06],
- adjusting the timing of one recording to that of a second by means of a dynamic time stretching algorithm. This is similar to beat-matching but potentially at a higher time resolution. In a music production

environment, the different voices of a homophonic arrangement (e.g., the backing vocals in a pop song) can, for instance, be automatically synchronized to the lead voice. The same applies for different instruments playing in unison or at least in the same rhythm,

- automated synchronization of a (dubbed) studio recording with the original, possibly distorted, recording, and
- musicological analysis of the timing information contained in the alignment path of several performances of the same score in order to compare the tempo and timing to a given reference music performance.

Standard audio-to-audio alignment approaches often use DTW as introduced above. They mainly vary in terms of the features extracted from the audio signal and the subsequent computation of the distance matrix. Hu and Dannenberg compute a simple pitch chroma as input feature [HDT03]. The distance is the Euclidean distance between all pitch chroma pairs of the two “audio” sequences. Turetsky and Ellis use several STFT-based features such as the power and the first order difference in time and frequency. They then use the cosine distance as similarity measure [TE03]. Dixon and Widmer argue that the main objective of audio-to-audio alignment is the synchronization of the onset times and propose to use a spectral flux-based feature subjected to HWR; the feature is computed in semi-tone bands [DW05]. The distance measure is the Euclidean distance. In order to achieve pseudo-real-time alignment, they use a modified DTW approach that estimates the optimal local path within a sliding window (see Sect. 10.1.3). Müller et al. compute the pitch chroma via the energy outputs of a filterbank. They use multi-scale DTW as described in Sect. 10.1.3 to efficiently compute the alignment path [MMK06]. Kirchhoff and Lerch pointed out that the type of features used for audio-to-audio alignment depends on the similarity to be evaluated between the signals; the distances can be computed, e.g., in the pitch domain, timbre domain, rhythm domain, or envelope [KL11]. In their study they proposed to train a two-class LDA classifier (*on path* vs. *not on path*) instead of a vector distance; this combines feature weighting and distance computation automatically.

Add distance matrix with paths for different features for two example alignments.

10.3 Audio-to-Score Alignment

Systems for the synchronization of an audio signal with a musical score (frequently in MIDI format) are sometimes categorized with respect to their real-time capabilities. There seems to be some consensus to refer to real-time systems as *score following* systems, and to call non-real-time (or offline) implementations *audio-to-score alignment* systems.

Possible applications of such alignment systems (compare [SRS03]) include

- linking notation and music performance in applications for musicologists to enable working on a symbolic notation while listening to a real performance,
- using the alignment cost as a distance measure for finding the best matching document in a database (i.e., retrieve an audio signal for a score query or vice versa),
- the musicological comparison of different performances of the same musical score,
- automatic accompaniment systems,
- the construction of an extended score describing a selected performance by adding information on dynamics, mix information, or lyrics, and
- musical tutoring or coaching systems for which the timing of a recorded performance is evaluated per note.

The core challenge of systems synchronizing audio and score is the proper handling of the different input modalities. While in the features for audio-to-audio alignment are extracted from audio signals, audio-to-score alignment systems need to compare features extracted from audio with features extracted from the score. This can lead to challenges in computing reliable distance measures.

10.3.1 Real-Time Systems

Historically, the research on matching a pre-defined score automatically with a music performance goes back to the year 1984. At that time, Dannenberg and Vercoe independently presented systems for the automatic (computer-based) accompaniment of a monophonic input source in real time [Dan84, Ver84]. In the following years, these systems were improved for higher accuracy by learning from real-world performances [VP85] or enhanced by allowing polyphonic input sources, increasing their robustness against musical ornaments, and by using multiple agent systems [BD85, DM88].

Baird et al. proposed a score following system working with MIDI input (for the performance) based on the concept of musical segments as opposed to single musical events; the tracking algorithm itself is not described in detail [BBZ90, BBZ93]. Heijink and Desain et al. presented a score following system that takes into account structural information as well. It uses a combination of DP and strict pitch matching between performance and score [Hei96, DHH97].

While many of studies presented above focus on the score following part rather than audio processing itself, Puckette and Lippe worked on systems with audio-only input with monophonic input signals such as clarinet, flute, or vocals [PL92, Puc95]. Vantomme proposed a monophonic score following system that uses temporal patterns from the performer as its primary information [Van95]. From a local tempo estimate the next event's onset time is predicted and the distance between expected onset time and extracted onset time is evaluated. In the case of an “emergency,” he falls back to the use of pitch information. Grubb and Dannenberg presented a system following a monophonic vocal performance. It uses the fundamental frequency, spectral features, as well as amplitude changes as features for the tracking process [GD97, GD98]. The estimated score position is calculated based on a PDF conditioned on a distance computed from the previous score event, from the current observation, and from a local tempo estimate. Raphael published several approaches to score following implementing probabilistic modeling and machine learning approaches [Rap99, Rap01, Rap04]. Cano et al. presented a real-time score following system for monophonic signals based on an HMM [CLB99]. They used the features zero crossing rate, energy, and its derivative plus three features computed from the fundamental frequency. Orio et al. introduced a score following system for polyphonic music which utilizes a two-level HMM modeling each event as a state in one level, and modeling the signal with attack, sustain, and rest phase in a lower level [OD01, OLS03]. They use a so-called *Peak Structure Distance (PSD)* that represents the energy sum of band-pass filter outputs with the filters centered around the harmonic series of the pitch of the score event under consideration. Cont proposed a polyphonic score following system using hierarchical HMMs using previously learned pitch templates for multiple fundamental frequency matching [Con06]. Dorfer et al. introduced Reinforcement Learning into the task of score following and showed that accuracy could be improved with an end-to-end system using images of sheet music [DJA⁺18].

10.3.2 Non-Real-Time Systems

While the publications presented above deal with score following as a real-time application, the following publications deal with the closely related topic of non-real-time audio-to-score alignment.

The importance of reliable pattern matching methods has already been recognized in early publications on score following and alignment; in most cases DP approaches have been used [Lar93]; see, for example, Dannenberg's publications on score following mentioned above. Orio and Schwarz presented an alignment algorithm for polyphonic music based on DTW which combined several local distances (similarity measures) [OS01]. It uses the PSD as described above and a *delta of PSD* (ΔPSD) modeling a kind of onset probability; it also uses a silence model for low-energy frames [OD01]. Meron and Hirose proposed a similar approach with several easy-to-compute audio features and suggested post-processing of the alignment results to improve the alignment robustness [MH01]. The system by Arifi et al. attempts to segment the audio signal into (polyphonic) pitches and performs DP to align MIDI file to the extracted data [Ari02, ACKM04]. The algorithm has been tuned for polyphonic piano music. Turetsky and Ellis avoided the problems of calculating a spectral similarity measure between score and audio by generating an audio file from the (reference) MIDI file and aligning the two audio sequences with audio-to-audio alignment [TE03]. For the alignment itself a DP approach is used. Similarly, Dannenberg and Hu generated an audio file from the MIDI file to align two audio sequences [DH03, HDT03].

They calculate the distance measure based on a 12-dimensional pitch chroma. The alignment path is then calculated by a DP approach. Shalev-Shwartz et al. presented a non-real-time system for audio-to-score alignment utilizing DP [SSKS04]. Their algorithm features a training stage for the weighting the features for the distance measure. They derived a confidence measure from audio and MIDI similarity data and trained a weight vector for these features to optimize the alignment accuracy over the training set. The audio feature set contains simple pitch-style features extracted by band-pass filtering, derivatives in spectral bands to measure onset probability, and a time deviation from the local tempo estimate. An alignment system of Müller et al. is also based on DP [MKR04]. While targeted at piano music, they claim genre independence. For the pitch feature extraction, they used a (zero phase) filterbank with each band-pass' center frequency located at a pitch of the equally tempered scale; the filter outputs are used to extract onset times per pitch. Devaney proposed a post-processing step for fine-alignment of onsets marked as simultaneous in the score by applying a HMM to a previously computed DTW result [Dev14].

In summary, the standard approach to audio-to-score alignment consists of three major processing steps: first, the audio feature extraction which in most cases approximates a pitch-like representation with onset information; timbre and loudness are often too performance-specific to be used for audio-to-score alignment. Second, a similarity or distance measure between audio and symbolic (score) features is computed. Finally, the actual alignment or path finding algorithm that is either based on DP, DTW, or on HMMs is applied. To allow for a useful distance matrix, a meaningful audio-to-score distance has to be computed. On the one hand, the score can be transformed into a more audio-like representation (ultimately by directly rendering the MIDI signal into an audio signal) that allows audio features to be extracted from the score; on the other hand the audio signal can be converted into a more score-like symbolic format, ultimately by transcribing the signal completely, and a score feature can be extracted from the audio. Alternatively, a distance measure between audio and score can be learned by data-driven methods [JER13].

10.4 Evaluation

The objective in evaluating alignment systems is to compare two sequences of time-stamps and how well they match each other. Therefore, a set of pairs of audio files (or one audio and one time-annotated score file) with clearly defined synchronization points is required as ground truth to evaluate the accuracy of an estimated alignment path. Similar to other tasks, properly annotated data is scarce for the evaluation of both audio-to-audio and audio-to-score alignment. Another important consideration is the time resolution of the ground truth labels. Common variants for audio-to-score alignment include note-level annotations, beat-level annotations, and bar-level annotations. The necessity of higher or lower resolutions is usually defined by the use-case. For score visualization, for example, matching audio at a bar level might be sufficient. For synchronization to trigger accompanying audio events, a much higher time resolution will be necessary. Due to the missing score information, audio-to-audio alignment is generally evaluated against the time-stamps of short blocks after blocking the signals.

Pauses or rests and held notes pose a problem to the evaluation of alignment systems, as there is no ground truth for “correctly” progressing through a rest or a fermata. Such pauses generally have to be treated differently or ignored completely in the evaluation. Similarly, long held notes can become a problem as they do not contain events to align.

10.4.1 Metrics

Cont et al. introduce the following metrics for audio-to-score alignment: (i) the *missed note rate*, i.e. the percentage of missed notes, (ii) the *misalign rate*, i.e., the percentage of notes with a time deviation to the references greater than a tolerance, (iii) the *piece completion*, i.e., the percentage of events until only misaligned events follow towards the end of the piece, (iv) the *average imprecision*, i.e., the average absolute deviation between ground truth and estimated event (compare Eq. 6.6) and (v) the *variance of error*, i.e., the variance of the deviations between ground truth event and corresponding estimated event.

For audio-to-audio alignment, the following metrics have been proposed [KL11]: (i) the *mean deviation*, i.e.,

the average offset between the estimated and the ground truth time, (ii) the *mean absolute deviation* similar to the “average imprecision” mentioned above, (iii) the *maximum deviation*, i.e., the largest deviation per file to have a worst-case estimate, and (iv) the *relative number of matching path points* between estimated times and reference times.

10.4.2 Data

There exist several ways of generating a ground truth data set for audio-to-audio alignment systems. Corresponding points in time can be (manually or semi-automatically) annotated, onset times can be monitored and stored during a recording of a computer-monitored instrument, or MIDI files can be rendered to audio by means of a sample player.

As pointed out in Sect. 9.3.3.2 the manual annotation of points in time is a rather arduous process and thus generally only a few points per test file can be labeled. However, Gadermaier et al. argue that modern alignment systems are sufficiently (human-level) reliable if only one of multiple performances of one piece is being annotated carefully while the others can be annotated automatically [GW19]. Therefore, automatic annotation of audio data followed by manual verification of the labels might lead to an unfair evaluation advantage of the system used to create the labels compared to others [CSSR07]. Still, it is a valid option to measure algorithmic improvements.

The drawback of using piano-generated data is its restriction to solo piano music; furthermore, confining the evaluation to note onsets might be sufficient for the case of solo piano music, however, other kinds of music may also require the synchronization during the time span in between onsets. Still, the MAESTRO dataset is a good resource for evaluating aligning piano music [HSR⁺19]. The disadvantage of using synthesized samples is that their properties might differ from “real-world” signals in terms of sound quality and signal complexity.

Alternatively, it is possible to artificially generate modified pairs of audio signals. In the MIDI domain, this can be achieved by varying the tempo in an editor; in the audio domain this can be done with a dynamic time stretching algorithm, i.e., an audio processing algorithm able to change the tempo without changing the pitch of an audio signal [Eva02]. This allows for high accuracy of the data set while allowing to test with a wide range of musical styles and instrumentations. In this case, however, the validity of the data set depends (i) on the realistic dynamic use of stretch factors and (ii) on the audio quality of the stretching engine. Furthermore, as the test data originates from the same audio signal the algorithm’s performance might be overestimated.

10.5 Exercises

10.5.1 Questions

1. Given two time series v_1 of length 4 and v_2 of length 5, what are the minimum and maximum possible DTW alignment path lengths?
2. Given two time series

$$\begin{aligned} v_1 &= [0 \ 2 \ 3 \ 1] \\ v_2 &= [0 \ 1 \ 2 \ 0 \ 1], \end{aligned}$$

- (a) compute the distance matrix D_{AB} using the absolute difference,
- (b) compute the cost matrix C_{AB} , and
- (c) compute the correct path indices with DTW.

10.5.2 Assignments

Part III

Music Identification, Classification, and Assessment

Chapter 11

Audio Fingerprinting

Fingerprinting aims at identifying audio recordings in a previously assembled database. More specifically, each recording is represented by a fingerprint, a unique and compact digest summarizing the (perceptually) relevant aspects of the recording. The fingerprint of an unknown recording is then the query to identify this recording in a database containing previously extracted fingerprints and to retrieve associated meta-data. In contrast to many other tasks presented in the book, fingerprinting does *not* attempt to extract musical properties from the audio signal but aims at identifying a specific recording (as opposed to a specific song). Different music performances (or recordings) of the same song should therefore have different fingerprints. However, a recording still has to be identified when subjected to quality degradation such as perceptual audio coding, added noise, distortions, and other typical signal manipulations.

There are two main areas of application, (i) *broadcast monitoring* to determine which songs have been played at which frequency to allow rights holders the estimation of royalties and track possible license violations, and (ii) end consumer services which enable the user to easily identify audio in order to provide meta data such as artist name, song title, album art, or other added value services. A detailed overview of other applications for fingerprinting has been given by Cano et al. [CBG⁺05].

Fingerprinting is not to be confused with *watermarking*. Although both are used for identifying recordings, watermarking embeds a perceptually unnoticeable data block directly in the audio data, utilizing methods similar to perceptual audio coding while fingerprinting does not modify the audio signal. Watermarking thus enables the content provider to embed different watermarks in the same audio content. It allows, for example, to embed a user-specific watermark in the specific copy of the recording in order to identify this specific user copy of the recording later. This is not possible with fingerprinting, which identifies a recording by its audio content itself, not by its embedded data. As watermarking can embed meta data directly it can also give the user direct access to this additional data (e.g., song title or artist name) without the need for a database connection. The major advantage of fingerprinting over watermarking is that the audio signal does not have to be modified. Therefore, fingerprinting can cover legacy audio recordings distributed previously or through other distribution channels. Furthermore, it avoids the risk that audio signal modifications can possibly degrade the audio quality (with similar quality degradation as caused by perceptual encoders). Table 11.1 compares the main properties of fingerprinting and watermarking.

A fingerprinting system consists of two basic building blocks, the fingerprint extraction and a database of

Table 11.1: Main properties of fingerprinting and watermarking in comparison

<i>Property</i>	<i>Fingerprinting</i>	<i>Watermarking</i>
Allows Legacy Content Indexing	+	-
Allows Embedded (Meta) Data	-	+
Leaves Signal Unchanged	+	-
Identification of	Recording	User or Interaction

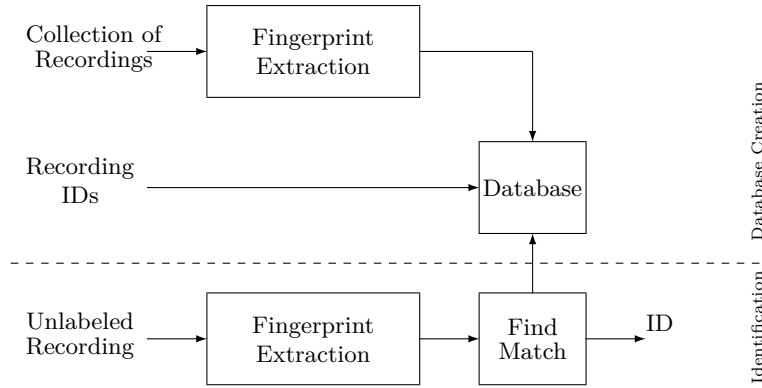


Figure 11.1: General framework for audio fingerprinting. The upper part visualizes the training phase and the lower part the query phase

previously extracted fingerprints coupled with unique identifiers or additional meta data about the piece of music. Figure 11.1 visualizes these blocks; the upper part of the graph shows the process of adding new entries to the database (done by the service provider) and the lower part shows the query for a recording by a client.

The requirements on a general fingerprinting system have been summarized by Cano et al. [CBKH05] as:

- *Accuracy & reliability*: high number of correct identifications (TPs) compared to the number of missed identifications (FNs) and wrong identifications (FPs).
- *Robustness & security*: high accuracy even in case of a heavily distorted signal. Possible distortions include lossy compression, added noise, equalization, interference, and non-linearities of the transmission path. Sophisticated systems should also be robust against changes in tempo and pitch.
- *Granularity*: the shorter the length of an excerpt required for its identification the better (modern systems require a length of a few seconds).
- *Versatility*: independence of detection from the file format and the file origin as well as an application-independent implementation.
- *Scalability*: good performance on very large databases and a large number of simultaneous identification queries.
- *Complexity*: low computational cost of both extracting a fingerprint and finding this fingerprint in the database.

11.1 Fingerprint Extraction

The fingerprint should be a unique identifier of a recording that allows unambiguous identification of a recording while being robust against quality degradations from, e.g., equalization, bandwidth restriction, audio coding, or background noise. For these reasons, it is sometimes referred to as *perceptual hash*. At the same time, it should be as compact as possible.

Since the fingerprint should be robust against bandwidth restrictions and audio format, the two most common pre-processing steps are down-mixing to a single mono channel and down-sampling to a lower sample rate (usually 5–20 kHz). Applying a high-pass filter discards frequency components below the lowest transmittable frequency of phones in an optional pre-processing step.

A system that can be seen as an early predecessor of today's fingerprinting systems targeted the detection of advertisements in broadcast streams. Lourens used the advertisement's energy envelope and selected a "unique" section serving as fingerprint [Lou90]. In most contemporary systems, the features are extracted in the frequency domain. These features include MFCCs [BMG02, CBMN02, RK06], a spectral flatness measure and

a spectral crest factor per frequency band [HAH⁺01], a spectral centroid per subband [SJL⁺05], band energies [KKKM01] or (the sign of) energy band differences [HK02], carefully selected spectral peaks [Yan01, Wan03], statistical moments of subbands [KY07], and modulation frequency features [SA02].

Frequently, the feature extraction process yields multiple values per processing block (often represented with a word length of 32 bits), leading to too much data to serve as fingerprint directly. In order to receive a more compact information and to decrease the memory footprint, many of these features (or the feature derivatives) are quantized into a binary or ternary representation. There have also been attempts at learning a fingerprint extraction algorithm automatically from the audio [CLP⁺21].

The resulting fingerprint then contains a unique series of quantized feature values or feature vectors.

11.2 Fingerprint Matching

The extracted fingerprint, representing the unknown recording, has to be compared against all previously stored fingerprints in the database. Often, the extracted fingerprint only consists of a segment of the song; after all, a user does not necessarily want to wait to record the whole song before identifying it. The database, however, has to contain all potential fingerprints of the whole recording to ensure a recording can be identified regardless of the position in the recording. As the database can be large, the similarity (or distance) measure has to be computationally efficient. Common measures include a correlation measure [SK04, MVSP11], the Euclidean distance [KKKM01, BPJ02, SJL⁺05], and the Manhattan distance (which in the case of binary input equals the *Hamming distance*) [RVKH00, HK02], but there exist many possible alternatives (see, e.g., [BMG02]).

Even with a fast-to-compute similarity measure, it might not be possible to compare every query fingerprint against all stored fingerprints in a large database due to workload and response time constraints. Various ways for performance improvement have been proposed. One simple way would be to pre-sort the database entries by their “popularity” in order to reduce search time for songs with frequent queries. It is also possible to pre-compute distances between the stored fingerprints in order to find different entry points for the query [KKKM01] or to use different similarity measures, an efficient one to discard many database entries in a first run and a second more accurate similarity measure to be computed on a selected small subset [KAHH02].

The decision whether a computed distance is considered a match or not requires careful tuning. It depends, as Cano et al. note, on the used fingerprint model, the discriminative information of the query, the similarity of the fingerprints in the database, and the database size [CBKH02].

11.3 Fingerprinting System: Example

To allow a better understanding of the process of audio fingerprinting, a widespread and frequently referenced system will be explained in detail in the following: the Philips fingerprinting system as published by Haitsma et al. [HK02].

After the signal is down-mixed to one channel and down-sampled to a sample rate of 5 kHz, it is subjected to a (von-Hann-windowed) STFT. The block length is 0.37 s and the hop size is 11.6 ms. The large block overlap ratio increases the system’s robustness against time-shift operations.

Figure 11.2 shows an overview of the subfingerprint extraction. The magnitude spectrum is divided into 33 non-overlapping bands in the range 300–2000 Hz. The frequency bandwidth increases logarithmically to take into account the non-linear frequency resolution of the human ear (see Sect. 7.1). The energy E per band k is then used to derive a binary result by using both the time and frequency derivative:

$$v_{\text{FP}}(k, n) = \begin{cases} 1 & \text{if } (\Delta E(k, n) - \Delta E(k, n - 1)) > 0 \\ 0 & \text{otherwise} \end{cases} \quad (11.1)$$

with

$$\Delta E(k, n) = E(k, n) - E(k + 1, n). \quad (11.2)$$

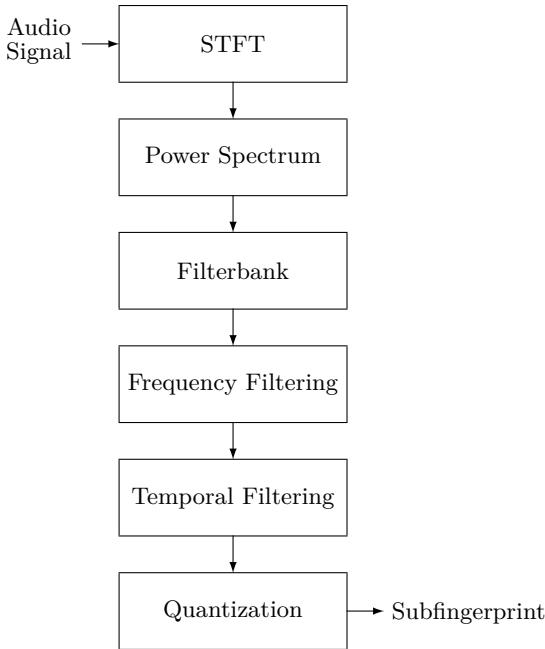


Figure 11.2: Flowchart of the extraction process of subfingerprints in the Philips system

This results in a 32-bit word per STFT; Haitsma et al. refer to this word as *subfingerprint*. One complete fingerprint consists of 256 subsequent subfingerprints and thus spans a time of 3 s. It has a size of $256 \cdot 4 \text{ Byte} = 1 \text{ kByte}$. Two example fingerprints are shown in Fig. 11.3, one of the original file (left) and one of the same file but encoded and decoded with MPEG-1 Layer 3 (MP3) (center). Each row is one subfingerprint. The differences between the two fingerprints are highlighted in red on the right.

Figure 11.4 shows example code for the extraction of this fingerprint.

The distance measure for the database search is the Manhattan distance; since the fingerprints are binary, the Manhattan distance equals the *Hamming distance*. The length of 3 s appears to be sufficient for the identification of a song from the database. The database has to contain the series of all subfingerprints of each complete recording. Thus, if the database contains one million songs of approximately 5 min length, it holds more than 25 billion subfingerprints. Even in the case of a highly compressed subfingerprint format and the use of the computationally efficient Hamming distance, this amount of subfingerprints rules out the brute force approach of searching the whole database for each query.

Haitsma et al. suggested two methods to improve computational efficiency of the database search, a simple and a more refined method. First, a lookup table is added to the database. This table contains all possible 32-bit subfingerprints which leads to a maximum number of 2^{32} table entries. Each table entry points to a list of occurrences in the database. The lookup table can also be replaced by a hash table for efficiency.

The first proposed method is based on the assumption that at least one of the 256 extracted subfingerprints has an exact match at the correct position in the database. Therefore, only the database entries listed under one of the 256 subfingerprints of the current query have to be evaluated as possible matches.

While this approach reduces the workload dramatically, the assumption that there is at least one subfingerprint without a bit error is not necessarily true for audio with more than minor degradations. A large number of bit errors can be expected for highly distorted signals, which means that the correct database entry might never be checked for a match. However, considering bit errors in the subfingerprints has the disadvantage of increasing the workload again: if *one* bit error is expected per subfingerprint, the number of database queries and thus the computational workload might increase as much as by a factor of 33. In order to reduce this additional workload while still taking into account possible bit errors, the concept of the *reliability* of a bit error is introduced in the second proposed method. Since the bits of a subfingerprint are computed by energy differences, the likelihood of a bit being flipped (a bit error) is high for small energy differences and low for

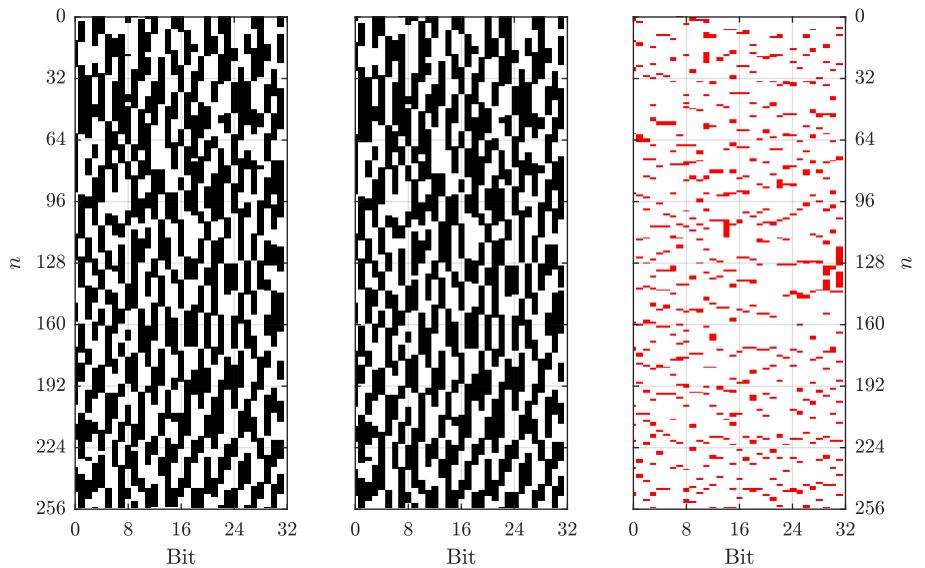


Fig. Gen.: plotFingerprint.m

Figure 11.3: Two Fingerprints and their difference. Left: original, center: mp3 encoded, right: bit differences between the two fingerprints.

```
% power spectrum
X = abs(X)*2/iBlockLength;
X([1 end],:) = X([1 end],:)/sqrt(2); %let's be pedantic about normalization
X = abs(X).^2;

% group spectral bins in bands
E = H*X;

% extract fingerprint through diff (both time and freq)
SubFingerprint = diff(diff(E,1,1),1,2);
tf = tf(1:end-1) + iHopLength/2*target_fs;

% quantize fingerprint
SubFingerprint(SubFingerprint<0) = 0;
SubFingerprint(SubFingerprint>0) = 1;
```

(a) Matlab

```
% power spectrum
X = abs(X)*2/iBlockLength;
X([1 end],:) = X([1 end],:)/sqrt(2); %let's be pedantic about normalization
X = abs(X).^2;

% group spectral bins in bands
E = H*X;

% extract fingerprint through diff (both time and freq)
SubFingerprint = diff(diff(E,1,1),1,2);
tf = tf(1:end-1) + iHopLength/2*target_fs;

% quantize fingerprint
SubFingerprint(SubFingerprint<0) = 0;
SubFingerprint(SubFingerprint>0) = 1;
```

(b) Python

Figure 11.4: Code example for fingerprint extraction.
[implement python](#)

large differences. Thus, the bits can be ranked by their reliability and only the unreliable bits have to be flipped for the database search.

11.4 Evaluation

In contrast to other tasks presented in this book, the evaluation of audio fingerprinting algorithms does not require musical annotations. Therefore, the size of the available datasets is not restricted by a time-consuming manual annotation step. However, the performance of a fingerprinting system is dramatically being impacted by the scale of the dataset used for evaluation; obviously, it is easier to identify a song correctly from a set of 1,000 recordings than from a set of 1,000,000 recordings. A dataset of that scale and with complete songs, however, is not publicly available.

Given these reasons, the evaluation section of existing audio fingerprinting studies is often not as rigorous as for other tasks, impacted by the lack of benchmark results and generally the availability of a dataset. Often, evaluations are carried out on small proprietary datasets of thousands of songs with limited replicability and validity.

Since most audio fingerprinting systems are required to work in scenarios with quality-degraded and distorted audio, experiments with time-shifted audio, added noise, different bandwidths, dynamics processing, and other linear or non-linear distortions can verify the robustness of the tested algorithm. Robustness against adversarial attacks such as time-stretching and pitch-shifting algorithm can also be an important consideration if the algorithm is intended for detecting copyright infringements.

The most common metrics are the TPR (also referred to as hit rate) and the FPR. These metrics have been introduced in Sect. 6.2.1. Note that some studies compute the hit rate with a margin: if the correct song is ranked in the list of top N predictions, it is considered a correct detection. Another interesting point of information is conveyed by the FNR (also referred to as false rejection rate). This is the ratio of negative responses for recordings that are in the database.

11.5 Exercise

11.5.1 Questions

11.5.2 Assignments

Chapter 12

Musical Genre Classification

The automatic recognition of the musical genre or the *musical style* from an audio signal is one of the oldest subjects of ACA and can be regarded as one of the seminal research topics shaping today's research field MIR.

The goal of *musical genre classification* is assigning a category from a pre-defined set of genre categories to piece of music based on the stylistic properties of the piece. The task is a classic machine learning task — suitable features are extracted from the audio signal and with these features a classification system is trained and used for classification.

Applications for this technology can be found mainly in the annotation, sorting, and retrieval of (similar) audio files from large databases or the Internet.

At first glance the meaning of the term *musical genre* appears to be self-explanatory and its intuitive meaning obvious. On a more thorough investigation, however, it has to be concluded that an objective definition of the term is hardly possible. First, it should be noted that this text uses “genre” and “style” interchangeably, although it can be argued these terms have separate meaning [Moo01]. Musical genres are labels assigned to categories grouping songs together based on some similarity, tradition, and set of conventions. The similarity dimensions used to cluster music into these categories can include a large number of high-level and low-level musical characteristics, such as musical form (e.g., Blues), instrumentation (e.g., String Quartet), timbre and effects (e.g., distortion in Rock), rhythm (e.g., Reggae), and performance techniques (e.g., Rap), but it might also include sociological, text-based, or other measures of similarity such as epoch as well as lyrics content and style [Moo01, Dan10]. Pachet and Cazaly [PC00] and later Scaringella et al. [SZM06] summarize some issues with existing genre categorizations:

- *Scope of the genre label*: Can an individual song be classified into a genre or does the context of album and performing artist influence or overrule the classification decision? Or may even different parts of a piece of music have different genres?
- *Non-agreement of taxonomies*: The number and definition of genre labels can strongly vary; in the year 2000, Pachet and Cazaly compared genre taxonomies from the three web sites Allmusic, Amazon, and MP3.com. The overall number of genres varied from 430 (MP3.com) to 531 (Allmusic) to 719 (Amazon), and these labels had only 70 terms in common. Furthermore, the hierarchy of the taxonomies differed. In some practical applications of ACA the number of genres has to be restricted due to technical limitations of the classification systems used, but the underlying problems of defining a taxonomy remain the same. Figure 12.1 shows two taxonomies defined in the context of early musical genre classification systems.
- *Ill-defined genre labels*: There is a semantic confusion between genre labels. They can be geographically defined (*Indian music*), related to an epoch in music history (*baroque*), refer to technical requirements (*barbershop*), the instrumentation (*symphonic music*), or usage (*Christmas songs*).
- *Scalability of genre taxonomies*: A specific genre may be split into a variety of subgenres (e.g., *Hip Hop* into *Gangsta Rap*, etc.). The number of subgenres might evolve over the years.
- *Non-orthogonality of genre categories*: One piece of music can possibly be sorted into multiple genre categories at the same time.

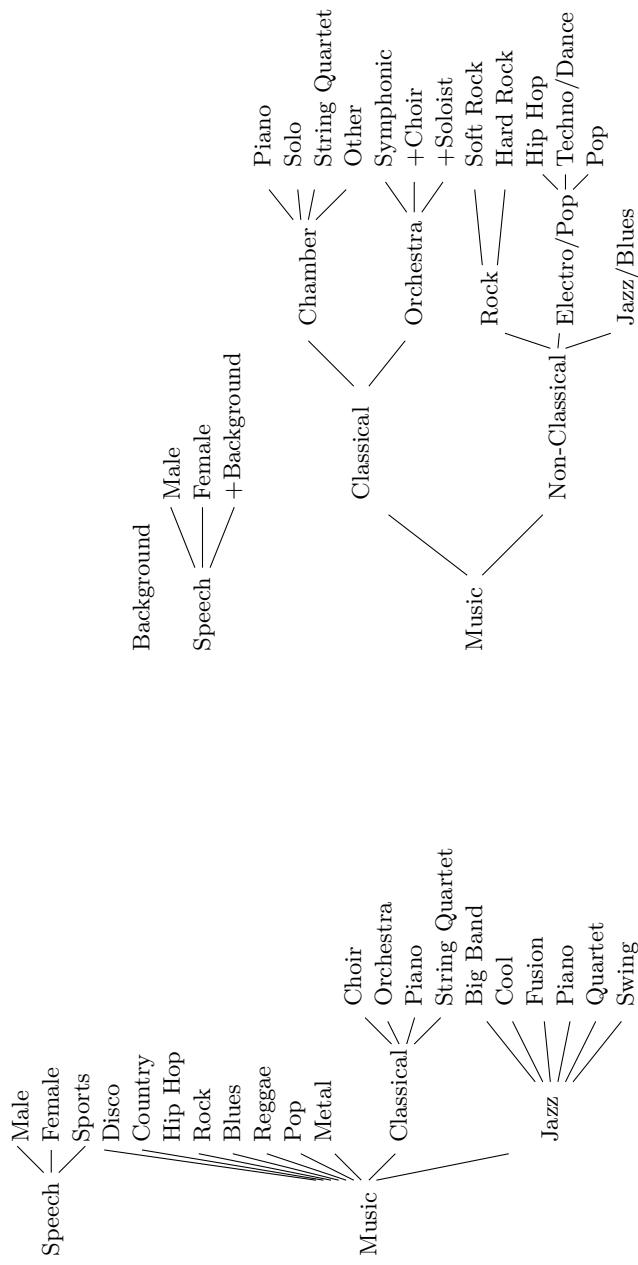


Figure 12.1: Two examples for author-defined genre taxonomies. Left: Tzanetakis and Cook [TC02], Right: Burred and Lerch [BL04]

- *Incompleteness of genre categories:* A piece might fall between or outside of pre-defined genre categories.

Given these observations it becomes clear that deriving a complete and musicologically consistent taxonomy is practically impossible [BL04]. Without a clear definition of the term *musical genre* it also comes as no surprise that the performance of humans annotating pieces with genre labels is far from perfect [Sol97, LMMT04, SWK10]. Since ACA systems are trained with human-labeled data, automatic systems can be hardly expected to outperform humans at this task.

12.1 Approaches to Musical Genre Classification

The aforementioned issues in defining genre unambiguously and objectively pose fundamental challenges to the task of musical genre classification. For these reasons, researchers have even suggested abandoning all attempts and focusing on other tasks [MF06]. Nevertheless, genre classification remains a popular research topic in the field of MIR because of its appealing and intuitive task definition and the general availability of labeled data.

The first publications on the automatic recognition of musical style appeared in the 1990s; at that time, the focus was mainly on the discrimination of speech and music signals [Sau96, SS97, WE99, CPLT99, EMKPK00], although studies aiming at classifying a broader range of signals can be found during that decade as well [WBKW96, Foo97]. Tzanetakis and Cook introduced what has been for many years considered a standard baseline approach in the year 2002 [TC02]: low-level feature extraction and aggregation followed by multi-class classification.

General surveys of approaches to musical genre classification have been published by Scaringella et al. and Fu et al. [SZM06, FLTZ11].

From a perceptual point of view, musical genre categorization may be influenced by

- *temporal characteristics* such as the tempo, the time signature, and rhythmic patterns,
- *dynamic characteristics* such as the loudness range, the change of loudness over time, accents,
- *tonal characteristics* such as melodic properties, the complexity of harmony (progression), and prominent pitch classes,
- *production-related characteristics* such as a specific sound quality and volume relations between instruments, effects and stereo panning properties, and
- *instrumentational characteristics* such as the number and type of instruments used.

Ultimately, this reflects all musical content categories in audio signals as introduced in Sect. 2.1.

Early audio classification systems utilized only a restricted set of features, starting with simple features such as the zero crossing rate (see Sect. 3.6.13) and intensity-related features (see Chap. 8) [Sau96, SS97, LKS⁺98, LWC98], although timbre-related spectral features quickly got added to the default set of features [WBKW96, Foo97, SSWW98]. Figure 12.2 plots a simple (and very noisy) feature space representation of the GTZAN dataset spanned by the standard deviation of the spectral centroid (x-axis) and standard deviation of the RMS (y-axis); while we see how some of the ten music classes form loose clusters, these two features alone obviously do not allow for a good separation of the classes. Pitch-related features which included properties of the fundamental frequency variation could also be found in the early speech/music classification systems [DTW97, Zha98, CPLT99] but are usually not used for systems targeting polyphonic audio input.

Over the years the number of features grew more and more, eventually covering more or less all features presented in Chap. 3.6 plus more custom-designed instantaneous features [Li00, MB03].

The most common features remained related to intensity and timbre but, although the classification results with these features alone seem to be surprisingly good, other feature dimensions got added to the set of features. These additional features include temporal and rhythmic features derived from a beat histogram (see Sect. 9.4) [TC02, DPW03, BL04], simple tonal features such as pitch histogram features [TEC02], and stereo panning features [TJM07].

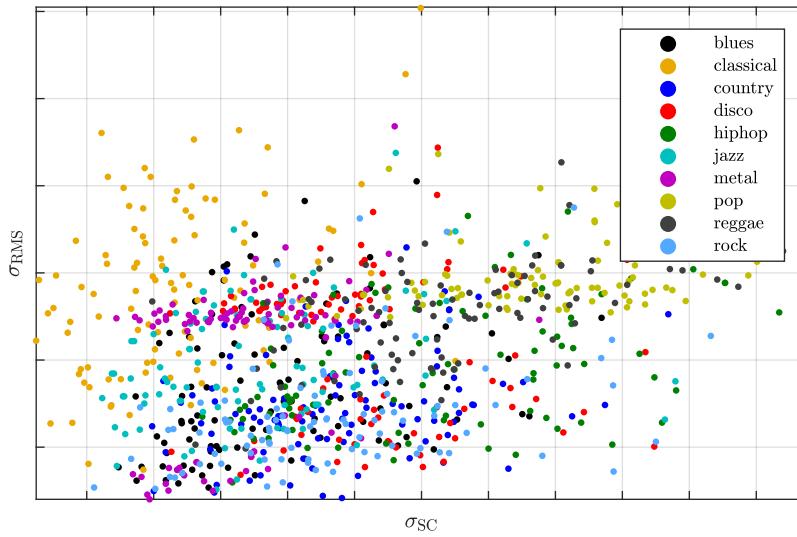
Fig. Gen.: [plotFeatureScatter.m](#)

Figure 12.2: Scatterplot of the feature space spanned by the RMS and the Zero Crossing Rate for the 10 music classes of the GTZAN dataset.

Nowadays, all these features are usually replaced by learned representations as introduced in Sect. 3.6.14. This replacement can either happen through training a neural network, e.g., with a Mel-Spectrogram input or with the help of transfer learning. While such transferred features can be learned from music [CFSC17, PS19], it is interesting to note that features learned from general audio classification such as VGGish [HCE⁺17] or OpenL3 [CWSB19] can perform well on music-related tasks [GCKT20].

Since we usually have the genre annotations per audio file or song, one could assume that the features mentioned above are simply aggregated (compare Sect. 3.7.3) over the whole audio file. However, the aggregation window (or texture window) can be significantly shorter: Tzanetakis and Cook found that the classification accuracy does not improve after a texture window length of 1 s, Burred and Lerch identified a texture window of approximately 15 s to yield satisfactory results, and Ahrendt et al. found a window length of 5 s to be sufficient in most cases. No solid conclusions can be drawn from these results except that the optimal texture window size depends strongly either on the features and subfeatures used or on the test data set and its categories. It is worth noting that humans seem to excel at quickly identifying the musical genre; in a data set with 10 genres test subjects were able to identify the genre reliably after listening to audio snippets significantly shorter than 1 s [GP08]. Texture windows can also be applied hierarchically: it is possible to aggregate features over a short window and then in turn aggregate the aggregated results over the file [TC00].

Traditionally, typical classifiers range from kNN [WBKW96] over GMM [TC02, BL04] to SVM [GL03], the latter nowadays often considered as the default baseline approach. A short description of these classifiers can be found in Sect. 4.1.

While Artificial Neural Networks (ANNs) have been proposed for musical genre classification as early as the late 1990s [SSWW98], only the performance of more modern deep architectures proved superior to the classifiers mentioned above. A common deep learning architecture is a CNN with a mel-spectrogram input [SD14].

12.2 Evaluation

As one of the historic core tasks in MIR, music genre classification systems are among the most evaluated tasks [Stu14]. As it is usually considered to be a straight-forward multi-class classification task, the evaluation usually follows standard classification methodology. The biggest issues, as pointed out above, stem from the acquisition of data and their annotations and will be summarized below.

12.2.1 Metrics

The most common metrics are, unsurprisingly, the standard classification metrics from Sect. 6.2.1: mean accuracy, precision, recall, and F-measure. The confusion matrix is also a common tool to analyze typical between-class confusions. It is also important to take into account potential class imbalance in the data. It has been argued, that the ground truth ambiguity inherent to this task should lead to adjustments in evaluation methodology and metrics [Stu13].

12.2.2 Data

The long history of the task combined with the intuitive appeal of automating musical genre annotations resulted in a comparably large number of publicly available datasets. They are, however, not necessarily compatible as they vary considerably in their definition of genre taxonomy, e.g., in their labels, the number of genre classes, and their hierarchy. For example, while most data assume only one label to be true, researchers have been making a case for music genre classification being better modeled as a multi-label scenario [LADG09, SZ11]. Many datasets also show strong preference for or focus exclusively on Western genres. Besides the genre label assignment itself, datasets have been criticized for including multiple songs of the same album or artist and thus possibly confounding the task with artist identification [PFW05, Fle07, FS10]. Typical problems with genre datasets have been highlighted by an in-depth analysis of the frequently-used GTZAN dataset (see below) by Sturm [Stu12].

The most commonly used datasets with available audio data are (listed alphabetically) the Ballroom dataset with 8 classes [GKD⁺06], the FMA datasets with up to 161 classes [DBVB17], the GTZAN dataset with 10 classes [TC02], the Homburg dataset with 9 classes [HB11], the ISMIR2004 genre dataset with 6 classes, the RWC dataset with 10 classes [Got02], and the Seyerlehner datasets with up to 19 classes [SWK08].

12.2.3 Results

Parsing the MIREX results over the past few years,¹ we see accuracies roughly in the range of 50–70% for tasks with 7 to 10 classes. While these numbers are mostly meaningless without detailed information on the audio collection and annotation process, at least they can provide a first rough estimate what to expect when working with genre data.

12.3 Exercises

[write me](#)

12.3.1 Questions

12.3.2 Assignments

¹https://www.music-ir.org/mirex/wiki/MIREX_HOME, last retrieved Aug 15, 2021

Chapter 13

Music Similarity Detection

A measure of *music similarity* allows the retrieval of similar musical pieces and enables applications such as intuitive browsing large catalogs of music or automatic playlist generation. It should be noted, however, that there are indications that content-based similarity measures for, e.g., recommendation, are consistently outperformed by systems building on contextual meta-data [Sla11].

In the previous chapter it was pointed out that music genre could be seen as a similarity cluster, formed by their acoustic and musical properties. In this sense music similarity could be derived from inter-genre distances. Reducing music similarity to genre similarity, however, is too simplifying (despite the complex nature of defining genre itself). While the task of describing music similarity is at least as subjective as assigning genre labels to music, genre similarity is only one facet of a large variety of factors influencing the perception of music similarity. The understanding of what music is considered similar often depends on the point of view. Two songs might be seen as similar because they are of the same musical genre when they are compared to songs of different genres, but the same songs might be considered completely dissimilar when compared to a cover version of one song. A DJ might want to look for songs with similar tempo or key in her database, but might actually appreciate if they are different in other aspects. Thus, the definition of music similarity depends on both the individual user and the task at hand.

To give a better impression of the multi-faceted and multi-dimensional nature of music similarity beyond genre similarity, a few aspects of similarity are listed here with a few literature examples as entry points: rhythmic similarity [PM87, HE02], structural similarity [Big90], melodic similarity [Bar88, EJLT01, AFL⁺06], harmonic similarity [LD01], lyrics similarity, timbral similarity, and similarity of performance characteristics [LD01, Tim05]. This list is far from exhaustive. Some of the listed characteristics will also impact genre similarity, but others might not. McAdams et al. propose to group all types of similarity into three clusters, surface and texture, figural, and structural [MM01].

Note that the subjective nature of music similarity applies also to these aspects by themselves. While rhythmic similarity is quite clearly a part of general music similarity, defining a model of rhythmic similarity is a difficult task in itself. Furthermore, there are indications that subjective music similarity ratings depend amongst other things on the familiarity of the test subject with the music [NMK07] and possibly on the listener's expert level as well.

Last but not least, the associative nature of memory cannot be neglected in real-world applications. Editorial data and context will also influence a human's rating of similarity. Editorial data refers to data that cannot be extracted from the audio signal such as recording date and studio, artists and producers who participated in other albums, the label, etc.

13.1 Approaches to Music Similarity Computation

The knowledge of music similarity is insufficient to build reliable quantitative models, however, even the existing knowledge of music psychologists and empirical musicologists is rarely utilized for the design of systems modeling audio similarity. For example, despite the wide range of similarity dimensions mentioned above, the majority of work on measuring similarity of musical audio has been focusing on timbre similarity. The features used are

therefore closely related to those for musical genre classification. They include feature sets such as MFCCs [Foo97, LS01b, FC01, AP02], loudness-related and rhythm-related features [PRM02, PDW03]. Nowadays, basically all low-level features and mid-level features introduced in the preceding chapters [LO04, BSWH09]. Other, less common features models of the spectral envelope, for example through K -means clustering [LS01b] or GMMs [AP02]. Similarly, Pampalk et al. proposed a histogram containing level classes per frequency band [PDW03].

The mapping of a music recording into the resulting, potentially high-dimensional, feature space is then used to model similarity by computing distance measures between this recording and others. Common pairwise distance measures are the Euclidean distance and the cosine distance (compare Sect. 13). The choice of features and their relative weighting to each other, therefore, can be seen as an ad-hoc definition of music similarity used by this system.

13.2 Evaluation

Researchers noted very early the challenges in evaluating any attempts at music similarity estimation. Even when focusing on timbre similarity specifically, the unclear definition of the task impedes the generation of reliable ground truth for evaluation. Aucouturier and Pachet suggested three ways of evaluating a system [AP02], (i) qualitative evaluation, i.e., looking and analyzing results for various pairs of songs, a method that Pampalk scaled-up by visualizing similarity distance through a Self-Organizing Map (SOM) [PDW03], (ii) evaluation through annotations for a related task such as the genre, the artist, the album, or other tags [LS01b, PDW03], and (iii) listening survey results that indicate the degree of similarity [LEB03]. Flexer and Grill, however, point out that there is huge variance between test subjects ranking similarity and thus the ground truth may only be of limited use [FG16].

While the result of listening studies is ultimately the only way of evaluating music similarity systems, the fundamental issues in the definition of this task and the accompanying challenges defining a proper methodology and setup of listening studies have ultimately led to slow progress in establishing ground truth data and evaluation methodology. One of the few publicly available datasets has only been released recently [LBS⁺20].

13.3 Exercises

13.3.1 Questions

13.3.2 Assignments

Chapter 14

Mood Recognition

The mood of a piece of music is one of the cues a user often finds helpful in finding and browsing music [Vig04]. This has lead to a considerable amount of research targeted at automatically recognizing the emotional characteristics of recordings of music. Terms for this task include (audio) *mood recognition* and *music emotion recognition*. This task aims at predicting the mood of a specific musical audio snippet. Albeit somewhat outdated, two overview articles provide a good introduction into the literature on this task [KSM⁺10, YC12].

The synonymous use of the terms *emotion* and *mood* is a first indicator that this task definition faces similar or worse challenges as the musical genre task introduced in Sect. 12. There is no general consensus on how emotion and mood are differentiated across researchers working on this task. It turns out that this is not only true for ACA researchers: Kleinginna and Kleinginna, for example, reviewed more than a hundred scientific definitions of emotion [KK81] without being able to identify a consensus. They conclude that some definitions might be better or worse but ultimately it is just not possible to prove the correctness of an individual definition.

Weld described the difference between emotion and mood by characterizing emotion as temporary and evanescent in contrast to mood which is more permanent and stable [Wel12]; mood has also been described as a diffuse affect state which can emerge without apparent cause [Sch05]. However, in a musical context it remains arguable if and when the term *emotion* should be preferred over the term *mood*.

Researching the relation of emotion and music exposes some of the typical problems in psychological research; one problem is, for example, the process of verbalization which confines the description of subtle and varied emotional states to the standardized words used to denote them [Mey56]. Meyer points out that descriptions of emotions are usually apocryphal and misleading since emotions are named and distinguished largely in terms of the external circumstances in which the response takes place; music itself, however, presents no external circumstances [Mey56]. Scherer criticizes the tendency to assume that music evokes basic emotions such as anger, fear, etc. [Sch03a]. This led him to propose the differentiation between *utilitarian* emotions, which are the emotions usually studied in emotion research (anger, fear, joy, disgust, sadness, shame, guilt, etc.), and *aesthetic* emotions which are not driven by external influences or personal goals but rather by the appreciation of the intrinsic qualities of a work of art [Sch04a]. Zentner et al. found that negative emotions such as guilt, shame, disgust, anger, fear, etc. are practically never aroused by music [ZGS08]. They also find that the “description of musical emotions requires a more nuanced affect vocabulary and taxonomy than is provided by current scales and models of emotion” [ZGS08].

The exact musical characteristics that lead to defining the mood of a piece of music continue being explored. There is strong evidence of a relation between moods and both the tempo and the loudness of the performance, as reported by Juslin [Jus00], Kantor [KM06], Sloboda and Lehmann [SL01], Schubert [Sch04b], and Timmers et al. [TMCV06]. The mode (major vs. minor) has also been reported to have influence on the mood [HTS02].

It is of importance to distinguish between the emotion aroused in the listener and the conveyed emotion perceived by a listener without particularly feeling it [Mey56]. Ratings of *perceived* emotion differ significantly from ratings of *felt* emotion [ZGS08]. A performance projecting ‘happiness,’ for example, does not necessarily make the listener happy even though they perceive the projected emotion.

To complicate matters further, it might be of interest to differentiate between score-inherent emotions and performance-inherent emotions [LAPG20]. To study this, however, seems to be only possible in very

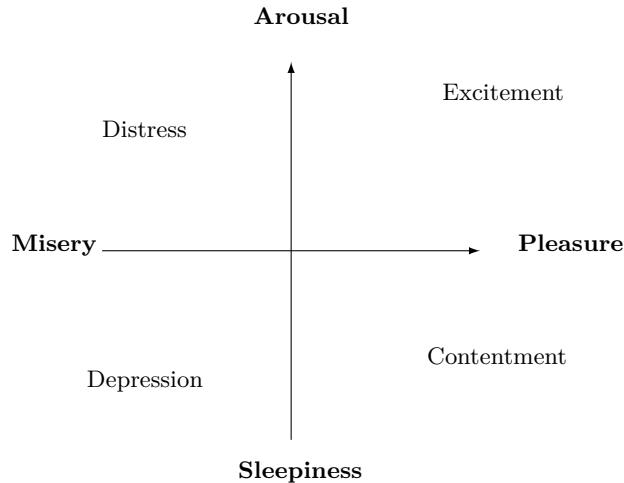


Figure 14.1: Russel’s two-dimensional model of mood

controlled environments [Jus00, Dil01, Dil03]; a real-world scenario does not allow for easy differentiation as the performance is an integral part of the “music” (see also Chap. 16). Also, the differentiation between score and music performance is less distinct for popular music than for non-popular or classical music.

14.1 Approaches to Mood Recognition

Building a model predicting mood requires either a categorical or a quantitative representation of mood. A categorical representation assumes a finite set of distinct mood classes. Based on preceding research on measuring emotional response to music, Schubert grouped mood labels into nine clusters as shown in Table 14.1 [Sch03b]. Deriving mood clusters by statistical analysis from large publicly available meta data collections has been done by Hu and Downie [HD07]. The resulting five clusters as shown in Table 14.2 are also used in the MIREX automatic mood recognition task. These clusters can then be used for labeling the training data of a classifier.

Alternatively, musical mood is frequently represented as a model with two dimensions or parameters instead of distinct label categories. Russel’s two-dimensional emotion space is one example of such models [Rus80]. It describes mood by the two dimensions *pleasure-misery* (horizontal) and *arousal-sleepiness* (vertical) and is displayed in Fig. 14.1. There have been other similar two-dimensional mood models presented [WT85, Tha89]; while they are not identical, they have some correspondence [FBR98, YRB99, BR99]. Extensions sometimes call for adding a third dimension (for example, *dominance* or *interest*) to this model, but the usefulness of this dimension is less obvious than for the first two dimensions [LVDV⁺05, MOCC07]. Using such a model, each mood will be represented by an two-dimensional (or even three-dimensional) coordinate in the mood space in the training data and can be predicted by a regression model.

The features extraction and classification algorithms used for mood recognition are very similar to the ones used in musical genre classification and music similarity. Some mood-specific features which are not common in musical genre classification are articulation-based features estimating the smoothness of “note transitions” [MDP08]. Mood recognition systems also use tempo and rhythm-related features more frequently than systems for musical genre classification [FZP03, LLZ06, YSLC07]. Since the measurable characteristics conveying the mood in music remain fuzzy and intangible, research is often exploratory in the sense that all conceivable features are investigated for the power to estimate musical mood.

The definition and number of mood classes varies across studies. Feng, for example, classifies music into the four classes *happiness*, *anger*, *sadness*, and *fear* [FZP03], while Li uses the three dimensions *cheerful-depressing*, *relaxing-exciting* and *comforting-disturbing* [LO04]. Russell’s two-dimensional arousal/valence-model as shown in Fig. 14.1 is probably the most frequently used model [Rus80]. This model avoids a categorical taxonomy

Table 14.1: Mood Clusters as presented by Schubert

<i>Cluster 1</i>	<i>Cluster 2</i>	<i>Cluster 3</i>	<i>Cluster 4</i>	<i>Cluster 5</i>	<i>Cluster 6</i>	<i>Cluster 7</i>	<i>Cluster</i>
Bright	Humorous	Calm	Dreamy	Dark	Heavy	Tragic	Agitate
Cheerful	Light	Delicate	Sentiment	Depressing	Majestic	Yearning	Angry
Happy	Lyrical	Graceful		Gloomy	Sacred		Restless
Joyous	Merry	Quiet		Melancholy	Serious		Tense
	Playful	Relaxed		Mournful	Spiritual		
		Serene		Sad	Vigorous		
		Soothing		Solemn			
		Tender					
		Tranquil					

Table 14.2: Mood clusters derived from meta data and used in MIREX

<i>Cluster 1</i>	<i>Cluster 2</i>	<i>Cluster 3</i>	<i>Cluster 4</i>	<i>Cluster 5</i>
Rowdy	Amiable/Good Natured	Literate	Witty	Volatile
Rousing	Sweet	Wistful	Humorous	Fiery
Confident	Fun	Bittersweet	Whimsical	Visceral
Boisterous	Rollicking	Autumnal	Wry	Aggressive
Passionate	Cheerful	Brooding	Campy	Tense/Anxious
		Poignant	Quirky	Intense
			Silly	

by using continuous values for arousal and valence and each piece of music can be represented by a point in the two-dimensional space [YLSC08, HBR10], which can then be predicted by regression algorithms. Some researchers, however, convert this model back to a categorical model by using each quadrant as a category or cluster (*contentment, depression, exuberance, anxious/frantic*) [LLZ06, YLC06]. Other options are using a fuzzy classification approach that assigns probabilities to each class [YLC06] and a multi-class or multi-label classification system which assigns a group of labels to a test sample [LO04, LLZ06, TTKV08].

Note that most approaches mentioned above assume the mood label to be stationary for the whole input audio recording. It has been pointed out that this is not necessarily the case, triggering research trying to predict a temporal evolution of mood [LLZ06, HRJH10]. The subjectivity of musical affect can also be taken into account by personalizing the model of emotional responses of individuals or groups [YSLC07].

14.2 Evaluation

While the issues in task subjectivity complicate the collection of trustworthy ground truth data, once these data have been accepted as ground truth, the system evaluation can be carried out mostly without further complications.

When mood recognition is interpreted as a classification task, the common metrics are accuracy and F-measure, precision, and recall. When it is interpreted as a regression task, the coefficient of determination R^2 is most commonly applied, but other metrics such as the MAE have been used [HBR10]. To evaluate both valence and arousal predictions together, researchers have also used the Euclidean distance [CYWC15].

Some of the available datasets include the AMG1608 [CYWC15] without audio files. The available datasets including audio data include the DEAM dataset [AYS17], the Soundtracks dataset [EV10] and the EmoMusic dataset [SCS⁺13] with annotations for valence and arousal, and the TAFFC dataset [PMP20] with quadrant annotation.

Recent results suggest that the accuracy range for classifying mood in the four quadrants or Russel's model is about 60 – 80% [KD21].

14.3 Exercises

14.3.1 Questions

14.3.2 Assignments

Chapter 15

Instrument Recognition

notes:

- task definition:
 - single note
 - melody
 - one instrument playing
 - pre-dominant instrument
 - polyphonic and polystimbral mixture
 - also refer to time resolution (start/end point vs. general presence in snippet)
 - mention specialized tasks singing voice detection, drum transcription
- challenges
 - general polyphonic mixture, etc
 - variety of timbre within one instrument (playing techniques etc.)
 - genre-dependent instrumentation
 - ill-defined instrument classes (synth, is a section, e.g., 1st violin one instrument or multiple?)
 -

Old section:

An algorithm for *instrument recognition* attempts to identify the musical instruments which compose a sound or are present in a musical recording. In contrast to the classification into genres or moods it is straightforward to find ground truth data for training and evaluation even if the problem of defining instrument taxonomies cannot be considered to be ultimately solved either [Kar90].

The two basic forms of instrument recognition can be distinguished by their type of input signal; some systems require a single note with no other pitches or instruments present; the signal has to be properly edited to eliminate preceding or succeeding notes or noises. Other systems work on a complex mixture of different instruments and estimate the (number and) type of the instruments present in the mixture.

Since the latter case is obviously algorithmically harder to handle, it comes as no surprise that most of the early publications on instrument recognition work on monophonic snippets of sound containing only one note. Herrera et al. give a good survey on the literature on monophonic instrument recognition [HABS00].

The restriction to short monophonic input signal snippets allows the definition and usage of a new specialized feature set that extends the set of features introduced in Chap. 3.6. More specifically, it allows the system to use features that cannot be used in the context of polyphonic music until it will be possible to separate the sources into individual monophonic subsignals. The additional features can be structured in two categories:

- *Temporal envelope features* are features describing the temporal evaluation of the sound. Simple examples are the sound's duration, its temporal centroid, and the (logarithmic) attack time.
- *Pitch-based features* utilize the fundamental frequency of the sound. Examples include the energy ratio of even and odd harmonics, the inharmonicity of the harmonics, and the onset asynchrony between harmonics.

Kaminskyj et al. presented one of the first systems for automatic instrument recognition [KM95, KV96]. It used a short time RMS and some harmonicity and spectral onset asynchrony features; the classification is done with either an ANN or a nearest-neighbor classifier. Similar to the systems for music similarity and musical genre classification, cepstral coefficients and MFCCs are frequently used for automatic instrument recognition, for instance, by the systems presented by Brown [Bro99] and Marques and Moreno [MM99]. With time, the number of features and the diversity of instrument classes increased, while as classification approaches basically the same systems KNN, ANN, GMM, and SVM are used [EK00, KC01, Ero01, BHM01].

The next level in the history of instrument recognition was reached when the input signals did not have to be individual notes anymore; while the input still needed to be monophonic, it could now contain phrases and whole melodies [ALP03, KS03, LR04, ERD04, ERD06, DSC08, JER09].

There are only a few systems estimating the instrumentation of polyphonic signals. Eggink and Brown presented a system based purely on spectral features with a GMM-based classification that automatically masks out temporary “unreliable” features [EB03a, EB03b, EB04]. Eisenberg proposed a system for detecting instruments using a so-called *harmonic peak spectrum* by modeling instrument sounds with harmonic sinusoidal peaks [Eis08]. It extracts the most salient component in the input signal so that it usually detects only the most prominent instrument; according to Eisenberg the system is able to detect accompanying instruments during pauses of the solo instrument. Heittola et al. attempt to decompose the signal into a sum of spectral bases and detect the individual sound sources [HKV09]. The classification is done with GMMs on MFCCs.

15.1 Drum Transcription

15.2 Exercises

15.2.1 Questions

15.2.2 Assignments

Chapter 16

Music Performance Assessment

Many of the ACA tasks introduced above focus on transcribing the musical audio signal into a score-like representation. Music performance analysis and its important subtask *music performance assessment* instead focus on the musical performance and its parameters. Typical research questions target, for example, the differences of two performances of the same piece of music, the identification of the most impactful performance parameters, and the affect of specific performance parameter variations on the listener.

Beyond the musicological and psychological interest in answering these questions, music performance assessment targets a set of applications for music instrument education: the ability to automatically assess a student performance, determine its quality, and understanding which characteristics are impacting an assessment enables artificially intelligent software tutors. Such tutoring systems have the potential to transform instrument practice and didactics of music education through low cost, individualized adaptive training sessions.

16.1 Music Performance

Music is a performing art. Therefore, a composer cannot “communicate” directly with a listener but requires a performer or group of performers who, in a traditional setting, “self-consciously enacts music for an audience” [Slo85]. The composition is a musical “blueprint” that is rendered by the performers into an acoustic realization [Hil02, Cla02b]. This acoustic signal is then received by the listeners. This process can be represented by the chain of music communication as shown in Fig. 16.1 [KC90, LAPG20].

The *composition*, referred to above as musical blueprint can, for example, be represented by a partitura in the case of Western classical music, a lead sheet for jazz, or some other genre-dependent representation describing the compositional content of a piece of music. *Scores* of the Western tradition always contain information on pitch and (relative) duration of each note and often general instructions on musical dynamics (compare Sects. 8.2, 7.2, and 9.2). Additional instructions, for instance, on character, quality, or specific ways to perform may also be found in the score. Some of the contained information is available only implicitly (for example, information on the musical structure) or might be ambiguous or hidden, complicating its description and quantification as pointed out by many musicologists and music psychologists [Dor42, Mey56, Pal97, BM99]. It can be observed that modern scores tend to be more explicit in terms of performance instructions than historic scores, indicating the composers’ intention to eliminate the unspecified or ambiguous information in the score [Dor42]. This may be due to the increasing awareness of the fact that scores often take into account

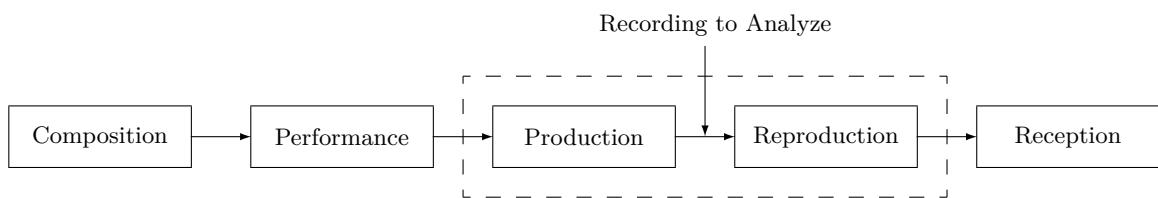


Figure 16.1: Chain of musical communication

performance rules that may seem “natural” at the time of composition but may change over decades and centuries, possibly leading to “unintended” performances.

In the music performance, the compositional information is then used by the performers for an acoustic realization of the score. All this information is subject to the performers’ interpretation — they detect and evaluate implicit information, try to understand and explain performance instructions, and identify ways to convey their understanding of musical ideas to the listener. By doing so, the performers create (either intentionally or unintentionally, either in preparation or on-the-fly) a performance plan for which the score information might be interpreted, modified, added to, and dismissed [Slo82, Gab99]. The decisions for the performance plan are based, for example, on (i) general interpretative rules describing norms which every performance follows to avoid being perceived as uncommon or unnatural, (ii) the expressive strategy, covering the intended communication of musical structure, the addition of unexpectedness, specific stylistic and cultural rules, the intended conveyance of mood, (iii) the performers’ personal, social, and cultural background, (iv) physical influences such as the physical and cognitive abilities of the performer, (v) rehearsal experience, and (vi) immediate influences such as the performers’ emotional state, room acoustics, and instrument feedback. For more details, compare [Dor42, Slo82, Slo85, Tod93, Pal97, TH02, Wal02, Cla02b, Cla02a, Par03, Jus03a, Jus03b, Ler09].

In a live performance setting, no recording, production, or acoustic reproduction takes place and the acoustic rendition is exactly what the listeners will hear depending on the acoustic properties of the venue and their location. The objective quantitative analysis of a performance, however, has to be based on a recording of a signal. Recorded performances differ significantly from the live performance, even in the case of so-called live recordings [Cla02a, Joh02]. Mechanical and technological restrictions enforce differences between an original and reproduced performance. Furthermore, choices and interventions by the production team will have impact on the recording. Maempel discusses as main influences in the context of classical music the sound engineer (dynamics, timbre, panorama, depth) and the editor (splicing of different tracks, tempo and timing, pitch) [Mae11]. A performance (plan) can also evolve during a recording session. Katz points out that in such a session, performers will listen to the recording of themselves and adjust “aspects of style and interpretation” [Kat04]. All these manipulations and influences can be part of the audio recording to be analyzed and cannot be separated anymore from the performers’ creation. As the release of a recording usually has to be pre-approved by the performers, it is assumed that the performers’ intent has not been distorted.

As hinted at above, the chain of musical communication as pictured in Fig. 16.1 is a simplified visualization. All stages between performance and listener may feed back information to the performer and subsequently reshape the performance plan.

16.2 Music Performance Analysis

Music Performance Analysis (MPA) focuses on the observation, extraction, description, interpretation, and modeling of music performance parameters as well as the analysis of attributes and characteristics of the generation and perception of music performance. Formally, performance parameters to be analyzed can be grouped into the same basic categories that we use to describe musical audio in general: tempo and timing, dynamics, pitch, and timbre. Although the change of performance parameters can have a major impact on a listener’s perception of the music [Cla02a], the nature of the performance parameter variation is often subtle, and must be evaluated in reference to something; typically either the same performer and piece at a different time, different performances of the same piece, or deviations from a perfectly quantized performance rendered from a symbolic score representation (e.g., MIDI) without tempo or dynamics variations. Such deviations and variations have often been described as representing musical expressivity, even if it is unlikely that all deviations can be considered as expressive [Dev16].

As Clarke points out, “musical analysis is not an exact science and cannot be relied upon to provide an unequivocal basis for distinguishing between errors and intentions” [Cla04], emphasizing the challenge of meaningful interpretation of extracted performance data. A related difficulty that MPA has to deal with is to distinguish between inherent performance attributes and individual performance attributes. In the context of musical accents, for example, Parncutt [Par03] distinguishes between *immanent accents* which are assumed to be apparent from the score (structural, harmonic, melodic, metrical, dynamic, instrumental) and *performed*

accents “added” to the score by the performer.

The performance data for MPA is either acquired through sensor-equipped instruments (e.g., Yamaha Disklavier), manual annotation (e.g., annotating onset times in a wave editor software), or by extracting parameters directly from the audio signal with methods described in the previous chapters. Sensors can give very accurate information, however, excludes the analysis of materials recorded outside of the lab setting. Manual annotation is typically regarded as time-consuming and tedious. While audio content analysis approaches allow access to an enormous and continuously growing heritage of recordings, including outstanding and legendary performances recorded throughout the last century and until now, it is questionable whether state-of-the-art algorithms can provide the detail and accuracy required for many MPA tasks [LAPG19].

Close relationships were observed between musical phrase structure and deviations in tempo and timing [Pov77, Sha84, Pal97] and dynamics [Rep96a, Ler09]. For example, tempo changes in the form of *ritardandi* tend to occur at phrase boundaries.

However, simply investigating deviations does not necessarily give an understanding of music performance as only the listener is the ultimately judge who ultimately consumes, processes, interprets, and assesses the music. The perceptual and subjective meaning of various performance characteristics has, however, often resulted in conflicting insights. For example, on the one hand musical tension has been reported unaffected by flat dynamics, constant tempo or both [Kru96, FU13], on the other expressive timing has been found as the most important parameter to convey musical tension [GPG⁺16].

16.3 Approaches to Music Performance Assessment

Instead of simply extracting and modeling individual performance parameters, the goal of *music performance assessment* is the estimation of overall ratings for a performance, taking into account parameters spanning the domains pitch (vibrato rates, intonation, tuning and temperament), timing (tempo, groove and micro-timing), dynamics (accents, tension), and timbre (playing techniques, instrument settings). This is a task of considerable commercial interest as it enables musically intelligent tutoring software. An automatic system for assessment of performances would provide objective, reliable, and repeatable feedback to the student during practice sessions and increase accessibility of affordable instrument education.

Performance assessment is a ubiquitous aspect of music pedagogy: regular feedback from teachers improves the students’ playing and auditions are used to place students in ensembles. It is, however, seen as a highly subjective and aesthetically challenging task with considerable disagreement on assessments between educators [TW03, WWE16]. With increasing musical proficiency, the assessment tends to become increasingly influenced by aesthetic considerations and “technical” skills such as the control of pitch intonation, musical dynamics, and tempo become a minimum requirement. Furthermore, it is important to note that there exists no true, absolute, or optional interpretation; music is a living art and constant re-interpretation is the artistic breath giving music life [Ler09].

Generally, the structure of a performance assessment system resembles the basic structure of an audio content analysis system: features describing the performance are extracted and then used in the inference stage to estimate one or more ratings. Current systems vary mostly with respect to the features used to describe the music performance, the need for a symbolic reference such as the musical score, and the type and granularity of the estimated assessment labels, and the musical instrument they focus on assessing.

While standard instantaneous features as introduced in previous chapters have been proposed for music performance assessment [KUF11], the majority of systems use hand-designed features informed by expert knowledge. Of the four assessment categories introduced above, intonation and timing are most frequently looked at in the literature. Especially features related to pitch deviation and pitch stability [NGH06, Luo15, WGL⁺16, VGW⁺17] and tempo and timing stability [AHG⁺14, WL18c] are frequently used. The analysis of tone quality in music performances is much less common [KUF11, RPRD⁺15, NR17]. A score-like reference to be practiced is commonly available in the educational setting, so many researchers incorporate information from the musical score into feature computation [MBL09, DMEF11, BBY17, VGW⁺17, HHP⁺20]. In many cases, such score information is simply taken into account by computing deviations from a “robotic” performance of the score.

The assessment inference can be seen as classification problem in the case that performances are simply categorized as 'good' or 'bad' [KUF11, NGH06], but the prediction via a regression of a rating on a scale, possibly for multiple different performance criteria, is more common [VGW⁺17, PGL18].

The general trend from feature design towards feature learning can be observed for music performance assessment, albeit at a slower pace than in other fields. Besides the general availability of training data, one of the possible reasons for this reluctance is that an educational setting normally requires not only an accurate assessment but also an explanation of the reasons for that assessment (and possibly a strategy to improve the performance). Many neural network architectures, however, do not allow simple interpretation of the learned features necessary for assessment [EMNS20]. Furthermore, most existing neural models for performance assessment estimate a single holistic rating for the whole performance and thus do not allow to pin-point specific playing mistakes.

The success rate of tools for modern automatic performance assessment still leaves considerable room for improvement. Most of the presented systems either work well only for very select data [KUF11] or have comparably low prediction accuracies [VGW⁺17, WGL⁺16], rendering them unusable in most practical teaching scenarios.

Appendices

Appendix A

Fundamentals

This chapter re-introduces and summarizes some of the important characteristics of digital audio signals and tools used in signal processing and ACA. Furthermore, it will introduce some definitions used in the following chapters. In order to keep this chapter short, some of the definitions may lack derivation since it is not possible to give an extensive introduction to digital audio signal processing in this context. The interested reader may refer to the books by Oppenheim and Schafer [OS99], Ohm and Lüke [OL07], and Smith [Smi07] for more complete introductions.

A.1 Sampling and Quantization

In the digital domain, we cannot deal with continuous signals. Therefore the analogue, continuous signal $x(t)$ has to be discretized in both amplitude and time, where *quantization* refers to the discretization of amplitudes and *sampling* refers to the discretization in time. The resulting signal is a series of quantized amplitude values $x(i)$.

A.1.1 Sampling

The discretization in time is done by *sampling* the signal at specific times. The distance T_S in seconds between sampling points is equidistant in the context of audio signals and can be derived directly from the *sample rate* f_S by

$$T_S = \frac{1}{f_S}. \quad (\text{A.1})$$

Figure A.1 visualizes the process of sampling by plotting a continuous signal (top) and a sampled representation of it (bottom) sampled at a sample rate of $f_S = 700$ Hz.

An important property of sampled signals is that the sampled representation is ambiguous as exemplified in Fig. A.2. This example shows that different input signals may lead to the same series of samples.

The signal can only be reconstructed unambiguously when certain requirements for audio signal and sample rates are met as defined by the *sampling theorem*:

Sampling Theorem

A sampled signal can only be reconstructed without loss of information if the sample rate f_S is higher than twice the bandwidth f_{\max} of the audio signal.

$$f_S > 2 \cdot f_{\max} \quad (\text{A.2})$$

Historically the sampling theorem is attributed to Kotelnikov [Kot03] and Shannon [Sha98] and is based on work by Whittaker [Whi29] and Nyquist [Nyg28].

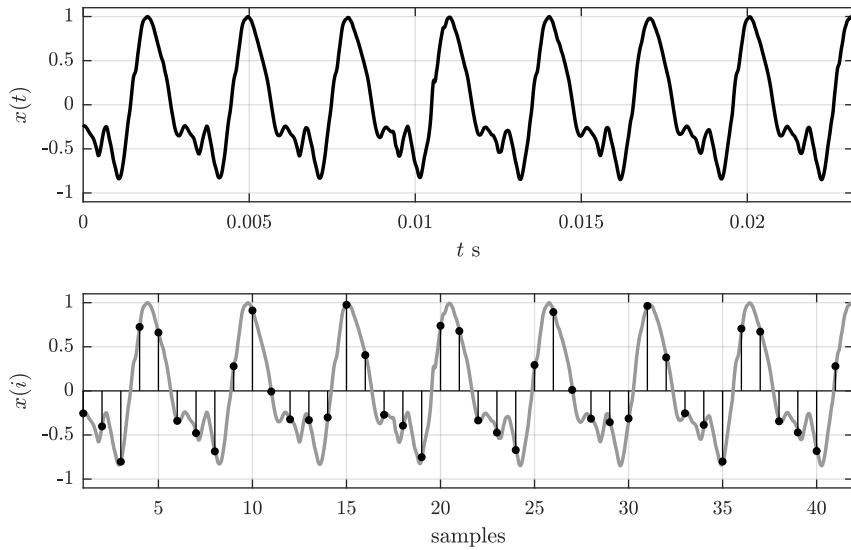


Figure A.1: Continuous audio signal (top) and corresponding sample values at a sample rate of $f_s = 700$ Hz).

Fig. Gen.: [plotSampling01.m](#)

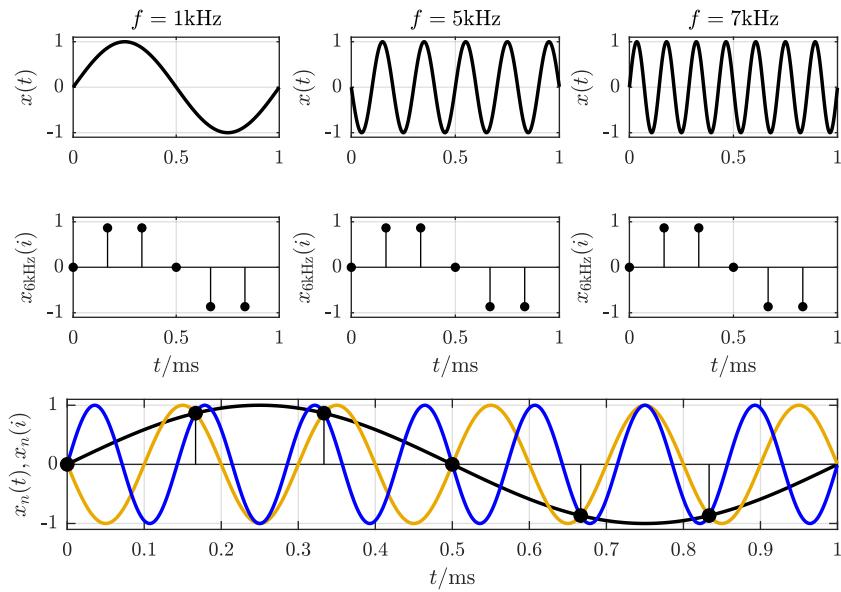


Figure A.2: Continuous (top) and sampled (below) sinusoidal signals with the frequencies 1 kHz (left), 5 kHz (mid), 7 kHz (right). The sample rate is $f_s = 6$ kHz.

Fig. Gen.: [plotSampling02.m](#)

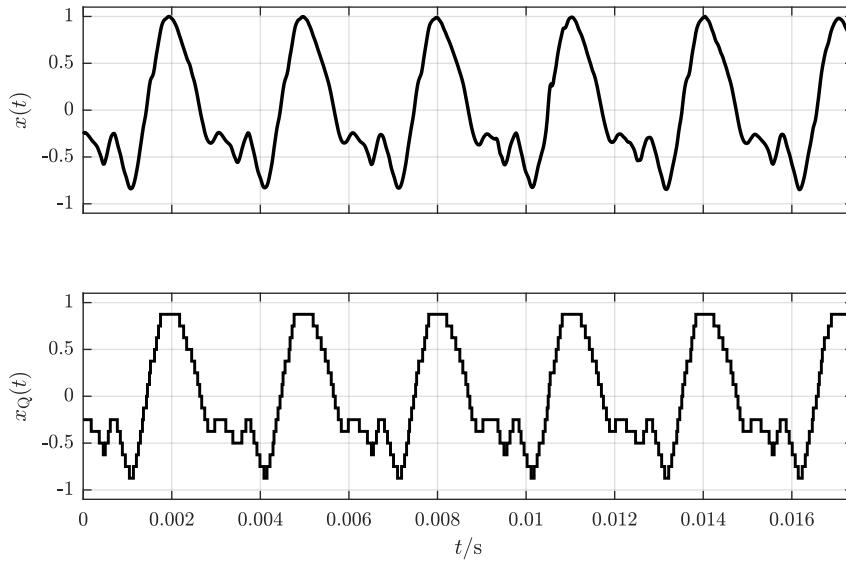
Fig. Gen.: [plotQuantization.m](#)

Figure A.3: Unquantized input signal (top) and quantized signal at a word length of 4 Bit (bottom).

If the signal contains frequency components higher than half the sample rate, these components will be mirrored back and *aliasing* occurs. See also Appendix B.3 for further explanation of this artifact.

A.1.2 Quantization

In order to discretize the signal amplitudes the signal is quantized. *Quantization* means that each amplitude of the signal is rounded to a pre-defined set of allowed amplitude values. Figure A.3 shows a signal in its continuous and quantized form. Usually, the input amplitude range between the maximum and the minimum amplitude is assumed to be symmetric around 0 and is divided into \mathcal{M} steps. If \mathcal{M} is a power of 2, the amplitude of each quantized sample can easily be represented with a binary code of *word length*

$$w = \log_2(\mathcal{M}). \quad (\text{A.3})$$

Since \mathcal{M} is an even number, it is impossible to both represent zero amplitude as a quantization step and maintain a symmetric number of quantization steps above and below zero. The usual approach is to have one step less for positive values. Typical word lengths for audio signals are 16, 24, and 32 bits. Figure A.4 shows the characteristic line of a quantizer and the corresponding *quantization error* in dependence of the input amplitude.

The distance Δ_Q between two neighboring quantization steps is usually constant for all steps. That means that the quantization error e_Q is always in the range $[-\Delta_Q/2; \Delta_Q/2]$ if the maximum input signal amplitude is within the range spanned by the maximum allowed quantized amplitudes (no *clipping*). The quantization error is the difference between original signal x and quantized signal x_Q :

$$e_Q(i) = x(i) - x_Q(i). \quad (\text{A.4})$$

If the input signal amplitudes are much larger than the step size Δ_Q but in the range of allowed amplitudes, the distribution of quantization error amplitudes can be assumed to be rectangular. The power of the quantization error P_Q can then be estimated with its *PDF* $p_q(e_Q)$ (compare Sect. 3.1.3):

$$P_Q = \int_{-\infty}^{\infty} e_Q^2 \cdot p_q(e_Q) de_Q = \frac{1}{\Delta_Q} \int_{-\Delta_Q/2}^{\Delta_Q/2} e_Q^2 de_Q = \frac{\Delta_Q^2}{12}. \quad (\text{A.5})$$

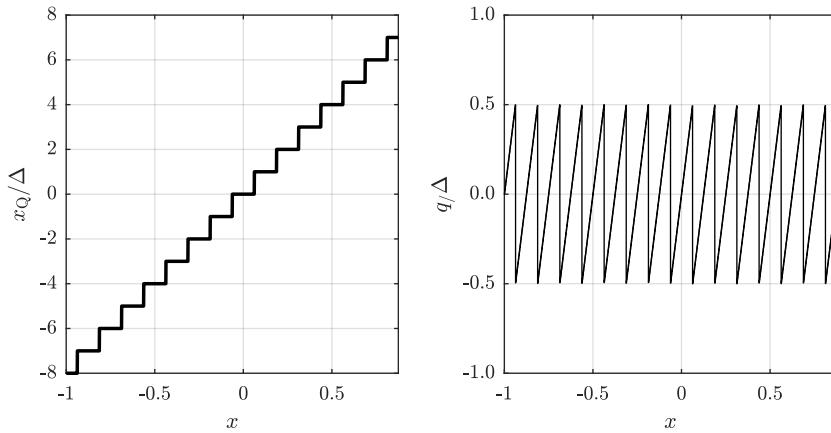


Fig. Gen.: plotQuantizationError.m

Figure A.4: Characteristic line of a quantizer with word length $w = 4$ bits showing the quantized output amplitude for a normalized input amplitude (left) and the quantization error with respect to the input amplitude (right).

It is obvious that the power of the quantization error will decrease with decreasing step size Δ_Q and with increasing word length w , respectively. The *Signal-to-Noise Ratio (SNR)* in decibels is a good criterion of the quality of a quantization system. The SNR can be calculated from the ratio of the signal power P_S and the quantization error power P_Q by

$$\text{SNR} = 10 \log_{10} \left(\frac{P_S}{P_Q} \right) [\text{dB}]. \quad (\text{A.6})$$

A high SNR thus indicates good quantization quality. Using Eqs. (A.5) and (A.6) it follows that increasing the word length by one bit gains approximately 6 dB:

Signal-to-Noise Ratio (SNR)

$$\text{SNR} = 6.02 \cdot w + c_S [\text{dB}] \quad (\text{A.7})$$

The constant c_S depends on the signal's amplification and PDF:

- square wave (full scale): $c_S = 10.80$ dB
- sinusoidal wave (full scale): $c_S = 1.76$ dB
- rectangular PDF (full scale): $c_S = 0$ dB
- Gaussian PDF (full scale = $4\sigma_g$)¹: $c_S = -7.27$ dB

Assuming that a real-world audio signal is closer to the PDF of a noise with Gaussian PDF than to that of a sine wave, the typical SNR is approximately 8 dB lower than commonly stated. Furthermore, the above reference values are given for signals that are leveled more or less optimally; if the input signal is scaled, the SNR will decrease accordingly (by 6.02 dB per scaling factor $1/2$).

The term *full scale* refers to the highest quantization step. A full-scale sine wave will thus have an amplitude equaling the highest quantization step (compare also page 155).

These considerations are often only of limited use to the signal processing algorithm designer, as the quantized signal is usually converted to a signal in floating point format, effectively resulting in a non-linear characteristic line of the quantizer. The floating point stores the number's mantissa and exponent separately

¹Using the approximation that no values are clipped.

Convolution Properties

- Identity for convolution with the delta function $\delta(i)$:

$$x(i) = x(i) * \delta(i) \quad (\text{A.8})$$

- Commutativity:

$$y(i) = x(i) * h(i) = h(i) * x(i) \quad (\text{A.9})$$

- Associativity:

$$g(i) * h(i) * x(i) = (g(i) * h(i)) * x(i) = g(i) * (h(i) * x(i)) \quad (\text{A.10})$$

- Distributivity:

$$g(i) * (h(i) + x(i)) = (g(i) * h(i)) + (g(i) * x(i)) \quad (\text{A.11})$$

- Circularity: If $h(i)$ is periodic, then the convolution result $y(i) = h(i) * x(i)$ will also be periodic.

so that the quantization step size Δ_Q basically increases with the input amplitude. In practice the signals in floating point format are just used as if they had continuous amplitude values, although the quantization error still persists. The choice of a maximum amplitude is arbitrary in floating point format; the most common way is to map full scale to the range $[-1; 1]$ to make the processing independent of the quantizer's word length.

A.2 Convolution

Every linear, time-invariant system can be completely described by its *impulse response* $h(i)$, which is the output of a system for an impulse as an input signal. Examples of such systems (or systems which can be approximated as such systems) are filters, speakers, microphones, rooms, etc. The output $y(i)$ of a discrete linear and time-invariant system can be computed from input signal $x(i)$ and the system's impulse response $h(i)$ of length \mathcal{J} by the *convolution* operation:

$$y(i) = x(i) * h(i) = \sum_{j=0}^{\mathcal{J}-1} h(j) \cdot x(i-j). \quad (\text{A.12})$$

The most important properties of the convolution operation are summarized in Eqs. (A.8)–(A.11). clean up structural mess

A.2.1 Identity

The result of a convolution with a delta function is the signal itself:

$$x(i) = \delta(i) * x(i). \quad (\text{A.13})$$

The result for each individual sample can be computed by the sum of the sample itself (weighted by 1) and all other samples weighted by 0, i.e., the sample value itself [compare Eq. (B.29)].

A.2.2 Commutativity

Changing the order of operands does not change the result of the convolution operation. That means that the distinction between impulse response and signal is of no mathematical consequence in the context of convolution:

$$h(i) * x(i) = x(i) * h(i). \quad (\text{A.14})$$

This can be shown by substituting $j' = i - j$:

$$\begin{aligned} x(i) * h(i) &= \sum_{j=-\infty}^{\infty} h(j) \cdot x(i-j) \\ &= \sum_{j'=-\infty}^{\infty} h(i-j') \cdot x(j') \\ &= \sum_{j'=-\infty}^{\infty} x(j') \cdot h(i-j'). \end{aligned} \tag{A.15}$$

A.2.3 Associativity

The associative property of convolution means that changing the order of subsequent convolution operations does not change the overall result. When applying two or more filters to a signal, the output will be identical for every order of filters.² This means that

$$(g(i) * h(i)) * x(i) = g(i) * (h(i) * x(i)). \tag{A.16}$$

This can be derived by changing the order of sums and shifting the operands as shown below:

$$\begin{aligned} (g(i) * h(i)) * x(i) &= \sum_{j=-\infty}^{\infty} (g(j) * h(j)) \cdot x(i-j) \\ &= \sum_{j=-\infty}^{\infty} \left(\sum_{l=-\infty}^{\infty} g(l) \cdot h(j-l) \right) \cdot x(i-j) \\ &= \sum_{j=-\infty}^{\infty} \left(\sum_{l=-\infty}^{\infty} g(l) \cdot h(j-l) \cdot x(i-j) \right) \\ &= \sum_{l=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} g(l) \cdot h(j-l) \cdot x(i-j) \\ &= \sum_{l=-\infty}^{\infty} g(l) \cdot \sum_{j=-\infty}^{\infty} h(j-l) \cdot x(i-j) \\ &= \sum_{l=-\infty}^{\infty} g(l) \cdot \sum_{j=-\infty}^{\infty} h(j) \cdot x(i-l-j) \\ &= \sum_{l=-\infty}^{\infty} g(l) \cdot (h(i-l) * x(i-l)) \\ &= g(i) * (h(i) * x(i)). \end{aligned} \tag{A.17}$$

A.2.4 Distributivity

The order of different linear operations is irrelevant due to the distributive property of the convolution, for example:

$$g(i) * (h(i) + x(i)) = g(i) * h(i) + g(i) * x(i). \tag{A.18}$$

²Strictly speaking this is only true for unlimited word length. The lower the word length the more the output signal differs from the expected signal.

This means that two signals, one computed by applying a filter to two different signals and summing them together afterwards, the other computed by applying the filter to the sum of the signals, are identical:

$$\begin{aligned}
 g(i) * (h(i) + x(i)) &= \sum_{j=-\infty}^{\infty} g(j) \cdot (h(i-j) + x(i-j)) \\
 &= \sum_{j=-\infty}^{\infty} g(j) \cdot h(i-j) + g(j) \cdot x(i-j) \\
 &= \sum_{j=-\infty}^{\infty} g(j) \cdot h(i-j) + \sum_{j=-\infty}^{\infty} g(j) \cdot x(i-j) \\
 &= g(i) * h(i) + g(i) * x(i).
 \end{aligned} \tag{A.19}$$

A.2.5 Circularity

The convolution with a periodic signal will result in a periodic output signal. The periodic signal $x(i)$ is the sum of the shifted (fundamental) periods with length N :

$$x(i) = \sum_{n=-\infty}^{\infty} x_N(i + nN). \tag{A.20}$$

With $x_N(i) = 0$ for $i < 0 \vee i \geq N$. We can show that

$$\begin{aligned}
 x(i) * h(i) &= \sum_{j=-\infty}^{\infty} h(i-j) \sum_{n=-\infty}^{\infty} x_N(j + nN) \\
 &= \sum_{n=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} h(i-j) \cdot x_N(j + nN) \\
 &= \sum_{n=-\infty}^{\infty} x_N(i + nN) * h(i).
 \end{aligned} \tag{A.21}$$

The multiplication of two spectra computed with the DFT will result in a circular convolution; the result will be the convolution of the two periodically continued sample blocks.

A.2.6 Simple Filter Examples

One typical linear system in digital signal processing is the filter. Every filtering process is a convolution, but for practical filter implementations this is only of interest for filters with a finite length impulse response, so-called *Finite Impulse Response (FIR)* filters. Filters with an infinite length impulse response are unsurprisingly referred to as *Infinite Impulse Response (IIR)* filters and have a feedback path which complicates the determination of the impulse response in comparison with FIR filters for which the impulse response simply equals its filter coefficients.

In the following, two simple and frequently used digital low-pass filters will be presented, the *Moving Average (MA)* filter as a prototypical FIR filter and the single-pole filter as a simple IIR filter.

A.2.6.1 Moving Average Filter

The filter equation for an MA filter with an impulse response of length \mathcal{J} is

$$y(i) = \sum_{j=0}^{\mathcal{J}-1} b(j) \cdot x(i-j). \tag{A.22}$$

The coefficients $b(j)$ of a typical MA filter have two main properties; the filter coefficients are identical:

$$b(0) = b(j) \quad \text{for } 0 \leq j \leq \mathcal{J} - 1, \tag{A.23}$$

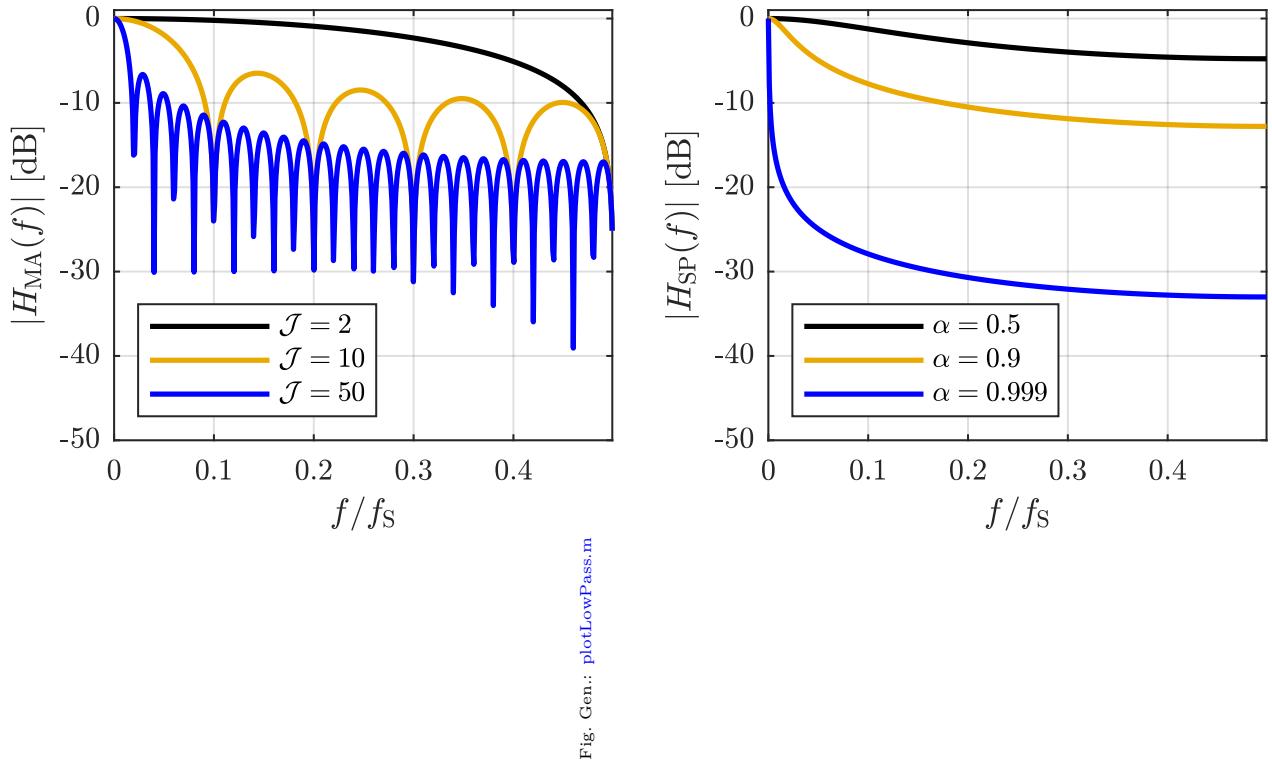


Fig. Gen.: plotLowPass.m

Figure A.5: Magnitude frequency response of a moving average low-pass filter with the impulse response lengths of 2, 10, and 50 samples (left) and frequency response of a single-pole low-pass filter with an α of 0.5, 0.9, and 0.999 (right).

and the sum of all coefficients is normalized to 1:

$$\sum_{j=0}^{\mathcal{J}-1} b(j) = 1. \quad (\text{A.24})$$

Equations (A.23) and (A.24) result in the coefficients $b(j)$ being identical and normalized to $b(j) = 1/\mathcal{J}$. Alternatively they can also be weighted by an arbitrary window function. Typical window functions are symmetric and weight center samples higher than lateral samples. The window shape allows to elongate the impulse response. One of the simplest window shapes would be triangular.

Applying a MA filter with unweighted coefficients twice to the signal is equivalent to applying a MA filter with a triangular shaped impulse response of double length. Thus, the repeated use of a MA filter is similar to increasing the impulse response length and applying a window function to the coefficients.

The MA filter's cut-off frequency, the frequency at which the magnitude first drops by 3 dB, will decrease with an increasing number of coefficients as depicted in Fig. A.5 (left). MA filters are commonly used for smoothing. Smith gives a detailed introduction to MA filters in [Smi99].

The number of multiply and add operations per hop increases linearly with the impulse response length of the MA filter. If no window function had been applied to the coefficients, i.e., all coefficients are identical

$b(k) = b$, the number of operations can be drastically reduced by implementing the filter recursively:

$$\begin{aligned}
 y(i) &= \sum_{j=0}^{\mathcal{J}-1} b \cdot x(i-j) \\
 &= b \cdot (x(i) - x(i-\mathcal{J})) + \underbrace{\sum_{j=1}^{\mathcal{J}} b \cdot x(i-j)}_{y(i-1)} \\
 &= b \cdot (x(i) - x(i-\mathcal{J})) + y(i-1).
 \end{aligned} \tag{A.25}$$

A recursive implementation is not applicable with weighted filter coefficients.

If the filter coefficients of the MA filter are symmetric with $h(i) = h(\mathcal{J}-1-i)$ (which is only possible for an FIR filter), then the filter will have a linear phase response and thus a constant group delay.

A.2.6.2 Single-Pole Low-Pass Filter

The *single-pole filter* is — mainly due to its simplicity and efficiency — one of the most frequently used low-pass filters for smoothing. The filter equation is

$$y(i) = \alpha \cdot y(i-1) + (1-\alpha) \cdot x(i) \quad \text{with } 0 \leq \alpha < 1. \tag{A.26}$$

Figure A.5 (right) shows the magnitude of the frequency response for three different α . The coefficient α depends on the required integration time T_I ; if the integration is defined to be the time required for the filter's response on a step function to rise from 10% to 90%,³ the coefficient is

$$\alpha = \exp\left(\frac{-2.2}{f_S \cdot T_I}\right). \tag{A.27}$$

The cut-off frequency is then

$$\omega_0(\alpha) = \arccos(2 - \cosh(\log(\alpha))) \tag{A.28}$$

which is only defined for $\alpha \geq e^{-\operatorname{arccosh}(3)}$.

A.2.7 Zero Phase Filtering with IIRs

No non-trivial causal system provides a *zero phase response*. A zero phase response means that the group delay at each frequency is zero as well. Zero phase response systems output the unmodified timing characteristics of each individual frequency group.

In the case of linear phase FIR filters a zero phase response can be reconstructed in an anti-causal way by removing the introduced delay from the filter output (which could also be interpreted as moving the impulse response to be symmetric around time 0).

In the case of IIR filters, however, this procedure is not applicable. In a non-real-time environment in which the whole input file is available, it is, however, possible to generate a filtered signal with zero phase shift by using an IIR filter. This can be done by applying the filter twice in series to the signal while using the time-reversed input signal the second time. The second filter run has the effect of leveling out the phase shifts introduced by the first run. The magnitude response of the complete filter process is then the filter's squared magnitude response. Figure A.6 visualizes the intermediate and the final result of this process in the time domain.

To explain this behavior mathematically, the following property is of importance:

$$\Im\{h(-i)\} = H(-j\omega), \tag{A.29}$$

³Note that there exist other approaches to defining the integration time that may differ significantly.

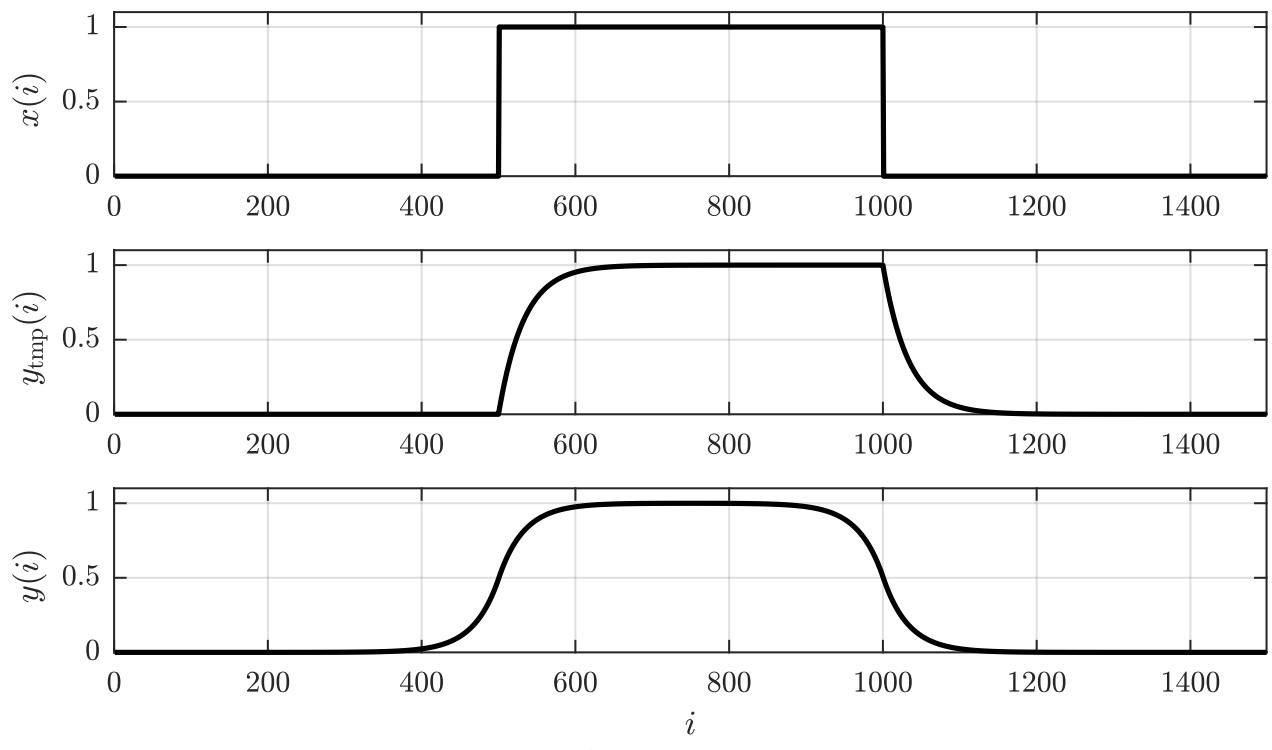


Fig. Gen.: [plotZeroPhase.m](#)

Figure A.6: Example of zero phase filtering: input signal (top), low-pass filtered signal after the first forward pass (mid), and final result of both forward and backward paths (bottom).

meaning that the *FT* \mathfrak{F} (see Sect. 3.4.1) of a time-reversed signal equals the complex-conjugate of the transform of the original signal. The flip function is a tool which helps us in the derivation:

$$\text{flip}(x(i)) = x(-i). \quad (\text{A.30})$$

The output signal $y(i)$ is computed via the temporary output $y_{\text{tmp}}(i) = h(i) * x(i)$ by

$$y(i) = \text{flip}(h(i) * y_{\text{tmp}}(-i)) \quad (\text{A.31})$$

$$= h(-i) * y_{\text{tmp}}(i) \quad (\text{A.32})$$

$$= h(-i) * (h(i) * x(i)). \quad (\text{A.33})$$

It follows in the Fourier domain:

$$Y(j\omega) = H(-j\omega) \cdot (H(j\omega) \cdot X(j\omega)) \quad (\text{A.34})$$

$$= |H(j\omega)|^2 \cdot X(j\omega). \quad (\text{A.35})$$

A.3 Correlation Function

The *correlation function* can be used to compute the similarity between two signals at a lag η . Given two signals $x(i)$ and $y(i)$, the so-called *Cross Correlation Function (CCF)* is defined by

$$r_{xy}(\eta) = \sum_{i=-\infty}^{\infty} x(i) \cdot y(i + \eta). \quad (\text{A.36})$$

Assuming the input signals are only blocks of $\mathcal{K} = i_e(n) - i_s(n) + 1$ samples, we can imagine an infinite number of zeros to the right and to the left of the actual samples; Eq. (A.36) can still be applied (although in a real application the multiplications with zero can obviously be omitted) and the CCF of two blocks of samples is thus

$$r_{xy}(\eta, n) = \sum_{i=i_s(n)}^{i_e(n)-\eta} x(i) \cdot y(i + \eta) \quad (\text{A.37})$$

and will equal zero for all lags larger than $\mathcal{K} = i_e(n) - i_s(n) + 1$:

$$r_{xy}(\eta, n) = 0 \quad \text{if } |\eta| \geq \mathcal{K}. \quad (\text{A.38})$$

Another possible interpretation of the block-wise CCF is the infinite correlation function of signals multiplied with a rectangular window $w_R(i)$:

$$r_{xy}(\eta, n) = \sum_{i=-\infty}^{\infty} x(i) w_R(i) \cdot y(i + \eta) w_R(i + \eta). \quad (\text{A.39})$$

A.3.1 Normalization

The correlation function is usually multiplied by a normalization factor λ_c . If only a block of samples of the signals $x(i)$ and $y(i)$ is being considered, then

$$\lambda_c = \frac{1}{\sqrt{\left(\sum_{i=i_s(n)}^{i_e(n)} x^2(i) \right) \cdot \left(\sum_{i=i_s(n)}^{i_e(n)} y^2(i) \right)}}. \quad (\text{A.40})$$

The number of non-zero multiplications decreases linearly with increasing lag until it is only one for $\eta = K - 1$. Therefore, the resulting correlation function is shaped triangular with the shape's maximum at lag $\eta = 0$. This

Autocorrelation Function Properties

- *Autocorrelation Function (ACF) at lag 0:*
 $r_{xx}(0, n) = 1$ when normalized according to Eq. (A.40), otherwise it is the *RMS* (see Sect. 8.3.1) of the block.
- *Maximum:*
 $|r_{xx}(\eta, n)| \leq r_{xx}(0, n)$ since the signal can at no lag be more similar to itself than without lag ($\eta = 0$).
- *Symmetry:*
 $r_{xx}(\eta, n) = r_{xx}(-\eta, n)$. The ACF is symmetric around lag $\eta = 0$. This means that the calculation of negative lags can be omitted for reasons of computational efficiency.
- *Periodicity:*
The ACF of a periodic signal will be periodic with the period length of the input signal.

triangular weighting can be avoided by taking into account the current lag η for the normalization:

$$\lambda_c(\eta) = \frac{\mathcal{K}}{(\mathcal{K} - |\eta|) \cdot \sqrt{\left(\sum_{i=i_s(n)}^{i_e(n)} x^2(i) \right) \cdot \left(\sum_{i=i_s(n)}^{i_e(n)} y^2(i) \right)}}. \quad (\text{A.41})$$

For large lags η the results will, however, become unreliable due to the limited amount of samples from which they are calculated.

A different way of avoiding the triangular shape of the correlation function is to use signal blocks of different sizes, namely a block length of $3\mathcal{K}$ for the signal x and a block length of \mathcal{K} samples for the signal y . Typically, the number of samples for x is increased by appending \mathcal{K} preceding and succeeding samples, respectively. While this leads to a extended theoretical non-zero range of r_{xy} for $|\eta| \leq 2\mathcal{K}$, the results for $|\eta| > \mathcal{K}$ are usually discarded. The normalization procedure is in this case unclear, although three options present themselves:

- *normalization per lag* requiring the computation of the denominator of Eq. (A.40) for every individual lag,
- *overall normalization* by computing the denominator from unequal length sequences, and
- *two-stage normalization* by first finding the maximum of the unnormalized cross correlation and then computing the denominator for two short length sequences around the maximum.

If only the current block of samples from signal x is available, this block may also be copied and pasted to the start and end of this block, resulting in the same amount of $3\mathcal{K}$ samples. If the block of samples from signal x would theoretically be pasted an infinite number of times, then the signal would be periodic with \mathcal{K} , which in turn would result in the periodicity of the correlation function: $r_{xy}(\eta) = r_{xy}(\eta + \mathcal{K})$. This is then called the *Circular Correlation Function (CCF)*; the circularity aspects have to be taken into account when the CCF is computed in the frequency domain as described in Sect. A.3.4.

The normalized CCF at lag $\eta = 0$ is also called the *Pearson correlation coefficient*.

A.3.2 Autocorrelation Function

The ACF is a special case of the correlation function with identical input signals $x(i) = y(i)$ and is therefore referred to as $r_{xx}(\eta)$. It is a measure of self-similarity, and its properties are summarized in the box above.

Figure A.7 shows the ACF computed with a block of a sinusoid and a block of white noise.

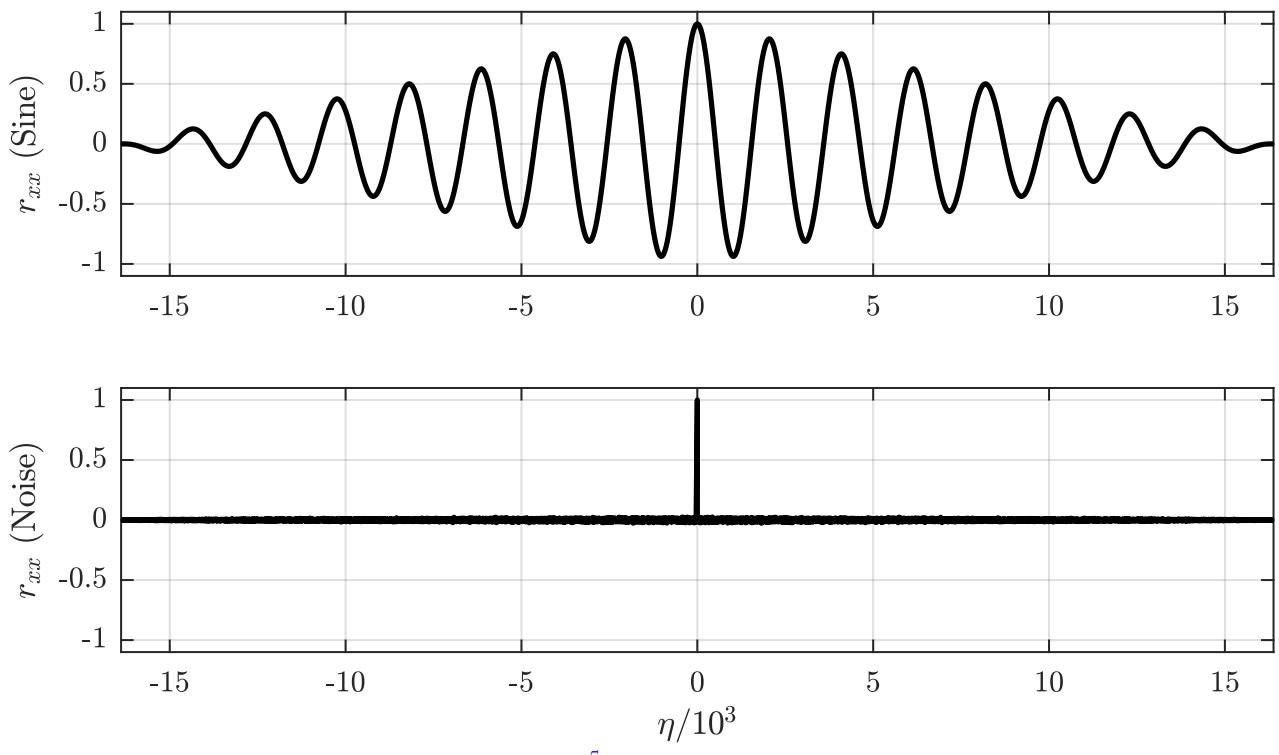
Fig. Gen.: `plotAcfExample.m`

Figure A.7: ACF of a sinusoid (top) and white noise (bottom).

A.3.3 Applications

The CCF can be used to find a specific latency or offset between a signal and the delayed signal. The lag of the maximum of the CCF indicates the delay between the signals. A typical example would be a radar signal with its echo coming back after being reflected by an obstacle. Finding the maximum of the CCF between the sent and the reflected signal should indicate the transmission time and thus the distance.

A CCF is sometimes also computed between a real signal and a constructed signal to find out whether the original signal shows similarities to the synthetic signal. An example would be to compute the CCF of the envelope of a drum loop with a series of weighted peaks to find a pattern in the signal.

The ACF is usually used to find periodicities within the signal. A periodic input signal will lead to a period ACF with peaks at the length of the period and its integer multiples. The ACF can be thus used to find a (fundamental) frequency (see Sect. 7.3.3.2) or to find the time interval between beats in music.

A.3.4 Calculation in the Frequency Domain

Rewriting either the CCF or the ACF as a convolution operation reveals an important property:

$$\begin{aligned} r_{xx}(t) &= \int_{-\infty}^{\infty} x(\tau) \cdot x(t + \tau) d\tau \\ &= x(t) * x(-t). \end{aligned} \quad (\text{A.42})$$

Using Eqs. (B.7) and (B.25) it can be deduced that

$$\begin{aligned} \mathfrak{F}\{r_{xx}(\tau)\} &= R_{xx}(j\omega) \\ &= \mathfrak{F}\{x(\tau) * x(-\tau)\} \\ &= X(j\omega) \cdot X^*(j\omega) \\ &= |X(j\omega)|^2. \end{aligned} \quad (\text{A.43})$$

This relation is called the *Wiener-Khinchin theorem*.

The computation of the ACF in the time domain can thus be replaced by a simple multiplication in the frequency domain preceded and followed by an FT and an inverse FT, respectively. For long blocks this will save computing cycles and reduce computational workload when using an FFT algorithm. Note that with blocks of sampled data the result will be a *circular* ACF if the transformed block of data has not been correctly padded with zeros before transforming it. More specifically, computing the CCF of two blocks of length \mathcal{K} requires a minimum FFT length of $2\mathcal{K}$.

Frequency Domain Compression In certain applications such as auditory processing it might be of interest to apply a non-linear compression function to the magnitude spectrum before transforming it back. Tolonen and Karjalainen named such a combined approach the *generalized ACF* to be computed by (see Pg. 126, [TK00])

$$r_{xx}^{\beta}(\eta, n) = \mathfrak{F}^{-1}\left\{|X(j\omega)|^{\beta}\right\}. \quad (\text{A.44})$$

A value $\beta = 2$ would result in the normal ACF.

Appendix B

Fourier Transform

The Fourier Transform (FT) is widely used in audio signal analysis and synthesis. Understanding its properties is crucial for the design of audio processing systems.

Deriving the FTs fundamental properties is easier for continuous signals; we will thus focus on the continuous domain first and will then discuss the FT of windowed signals, the FT of sampled signals, and finally the Discrete Fourier Transform (DFT).

Periodic signals can be represented as a Fourier series as introduced in Eq. (3.3). The fundamental frequency ω_0 determines the “frequency resolution” of the series. For the analysis of non-periodic signals we let the period length grow $T_0 \rightarrow \infty$ (or equivalently $\omega_0 \rightarrow 0$). This has the effect that the previously discrete frequency resolution becomes continuous with $k\omega_0 \rightarrow \omega$. Due to the resulting infinite resolution of the frequency axis, the coefficients will decrease $a_k \rightarrow 0$. The formula for the Fourier series given in Eq. (3.3) thus changes into the FT:

$$X(j\omega) = \mathfrak{F}\{x(t)\} = \int_{-\infty}^{\infty} x(t)e^{-j\omega t} dt \quad (\text{B.1})$$

and $X(j\omega)$ is the so-called *spectrum* of the signal $x(t)$.

The real and imaginary parts represent the cosine and sine functions, respectively. A common form of visualizing the results is to represent the spectrum as magnitude $|X(j\omega)|$ and phase $\Phi_X(j\omega)$ instead of real and imaginary parts. Frequently only the *magnitude spectrum* is being used for the visualization of the spectrum while the phase spectrum is ignored. Another common representation is the *power spectrum* which is the squared magnitude spectrum.

B.1 Properties of the Fourier Transformation

B.1.1 Inverse Fourier Transform

The FT is invertible. $X(j\omega)$ can be converted back from the frequency domain into the time domain signal $x(t)$ by applying the *Inverse Fourier Transform (IFT)*:

$$x(t) = \mathfrak{F}^{-1}\{X(j\omega)\} = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(j\omega)e^{j\omega t} d\omega. \quad (\text{B.2})$$

That means that time and frequency representation are equivalent, i.e., no information is gained or lost by applying the FT to a signal; it just changes the representation of the signal.

It becomes also obvious from comparing Eqs. (B.1) and (B.2) that forward and inverse transform are very similar operations (see also Sect. B.1.6).

B.1.2 Superposition

If the signal $y(t)$ is the weighted addition of the signals $x_1(t)$ and $x_2(t)$

$$y(t) = c_1 \cdot x_1(t) + c_2 \cdot x_2(t), \quad (\text{B.3})$$

then the same relationship is true for their frequency transformation:

$$\begin{aligned} Y(j\omega) &= \int_{-\infty}^{\infty} (c_1 \cdot x_1(t) + c_2 \cdot x_2(t)) \cdot e^{-j\omega t} dt \\ &= c_1 \cdot \int_{-\infty}^{\infty} x_1(t) e^{-j\omega t} dt + c_2 \cdot \int_{-\infty}^{\infty} x_2(t) e^{-j\omega t} dt \\ &= c_1 \cdot X_1(j\omega) + c_2 \cdot X_2(j\omega). \end{aligned} \quad (\text{B.4})$$

B.1.3 Convolution and Multiplication

The convolution of signal $x(t)$ with the impulse response $h(t)$

$$\begin{aligned} y(t) &= h(t) * x(t) \\ &= \int_{-\infty}^{\infty} h(\tau) \cdot x(t - \tau) d\tau \end{aligned} \quad (\text{B.5})$$

corresponds to a multiplication in the spectral domain

$$Y(j\omega) = H(j\omega) \cdot X(j\omega). \quad (\text{B.6})$$

The derivation involves clever grouping and expansion:

$$\begin{aligned} Y(j\omega) &= \int_{-\infty}^{\infty} y(t) e^{-j\omega t} dt \\ &= \int_{-\infty}^{\infty} \left(\int_{-\infty}^{\infty} h(\tau) \cdot x(t - \tau) d\tau \right) e^{-j\omega t} dt \\ &= \int_{-\infty}^{\infty} h(\tau) \int_{-\infty}^{\infty} x(t - \tau) e^{-j\omega t} dt d\tau \\ &= \int_{-\infty}^{\infty} h(\tau) e^{-j\omega \tau} \underbrace{\int_{-\infty}^{\infty} x(t - \tau) e^{-j\omega(t-\tau)} d(t - \tau)}_{X(j\omega)} d\tau \\ &= \int_{-\infty}^{\infty} h(\tau) e^{-j\omega \tau} d\tau \cdot X(j\omega) \\ &= H(j\omega) \cdot X(j\omega). \end{aligned} \quad (\text{B.7})$$

This property allows the efficient computation of the convolution of a signal with an FIR filter with a long impulse response in the frequency domain [Gar95].

The same relationship exists for convolution in the frequency domain. The convolution operation

$$Y(j\omega) = H(j\omega) * X(j\omega) \quad (\text{B.8})$$

could be replaced by a multiplication in the time domain

$$y(t) = h(t) \cdot x(t). \quad (\text{B.9})$$

B.1.4 Parseval's Theorem

The energy of the signal can be calculated in both the time and the spectral domain:

$$\int_{-\infty}^{\infty} x^2(t) dt = \frac{1}{2\pi} \int_{-\infty}^{\infty} |X(j\omega)|^2 d\omega. \quad (\text{B.10})$$

This can be shown by using the equivalence between multiplication in the frequency domain and convolution in the time domain. Writing

$$\int_{-\infty}^{\infty} h(\tau) \cdot x(t - \tau) d\tau = \frac{1}{2\pi} \int_{-\infty}^{\infty} H(j\omega) \cdot X(j\omega) e^{j\omega t} d\omega \quad (\text{B.11})$$

and replacing $H(j\omega)$ with the conjugate-complex spectrum $X^*(j\omega)$ and $h(\tau)$ with $x(-\tau)$,¹ respectively, the result at $t = 0$ is

$$\begin{aligned} \int_{-\infty}^{\infty} x(-\tau) \cdot x(-\tau) d\tau &= \frac{1}{2\pi} \int_{-\infty}^{\infty} X^*(j\omega) \cdot X(j\omega) d\omega, \\ \int_{-\infty}^{\infty} x^2(t) dt &= \frac{1}{2\pi} \int_{-\infty}^{\infty} |X(j\omega)|^2 d\omega. \end{aligned} \quad (\text{B.12})$$

B.1.5 Time and Frequency Shift

The transform of a signal shifted by a constant in time $y(t) = x(t - t_0)$ is

$$Y(j\omega) = X(j\omega)e^{-j\omega t_0}. \quad (\text{B.13})$$

This means that the magnitude spectrum will be identical but the phase spectrum will have a linear offset $\Phi_Y(\omega) = \Phi_X(\omega) - \omega t_0$:

$$\begin{aligned} \int_{-\infty}^{\infty} x(t - t_0) e^{-j\omega t} dt &= \int_{-\infty}^{\infty} x(\tau) e^{-j\omega(\tau + t_0)} d\tau \\ &= e^{-j\omega t_0} \int_{-\infty}^{\infty} x(\tau) e^{-j\omega\tau} d\tau \\ &= e^{-j\omega t_0} \cdot X(j\omega). \end{aligned} \quad (\text{B.14})$$

Equivalently, the shifted spectrum² $Y(j\omega) = X(j(\omega - \omega_0))$ corresponds to the time domain signal $y(t) = x(t) \cdot e^{j\omega_0 t}$ which is the original signal modulated by a sinusoidal signal:

$$\begin{aligned} \frac{1}{2\pi} \int_{-\infty}^{\infty} X(j\omega - \omega_0) e^{j\omega t} d\omega &= \frac{1}{2\pi} \int_{-\infty}^{\infty} X(j\phi) e^{j(\phi + \omega_0)t} d\phi \\ &= e^{j\omega_0 t} \cdot x(t). \end{aligned} \quad (\text{B.15})$$

¹Only real-valued time domain functions $x(t)$ are considered here.

²In real-valued time signals, this shift has to be applied symmetrically to the negative frequencies.

B.1.6 Symmetry

If the time domain signal $x(t)$ is real-valued, then its frequency transform will be symmetric with $X(j\omega) = X^*(-j\omega)$. The magnitude is symmetric around the y-axis:

$$|X(j\omega)| = |X(-j\omega)| \quad (\text{B.16})$$

while the phase is symmetric around the origin:

$$\Phi_X(\omega) = -\Phi_X(-\omega). \quad (\text{B.17})$$

Vice versa, if the frequency transform is real-valued, then the time domain signal will be symmetric with $x(t) = x(-t)$ and if $X(j\omega)$ is imaginary it means that $x(t) = -x(-t)$. This can be shown by representing the time signal $x(t)$ as a sum of an even component x_e and an odd component $x_o(t)$:

$$x(t) = \underbrace{\frac{1}{2}(x(t) + x(-t))}_{x_e(t)} + \underbrace{\frac{1}{2}(x(t) - x(-t))}_{x_o(t)}. \quad (\text{B.18})$$

The FT of the even and odd signal components is then

$$X_e(j\omega) = \int_{-\infty}^{\infty} x_e(t) \cos(\omega t) dt - j \underbrace{\int_{-\infty}^{\infty} x_e(t) \sin(\omega t) dt}_{=0}, \quad (\text{B.19})$$

$$X_o(j\omega) = \underbrace{\int_{-\infty}^{\infty} x_o(t) \cos(\omega t) dt}_{=0} - j \int_{-\infty}^{\infty} x_o(t) \sin(\omega t) dt. \quad (\text{B.20})$$

The transform of the even signal is thus purely real $X_e(j\omega) = \Re[X(j\omega)]$, and the transform of the odd part is purely imaginary $X_o(j\omega) = \Im[X(j\omega)]$. Furthermore, due to the property $\cos(\omega t) = \cos(-\omega t)$, it becomes clear that the real part is again an even function symmetric around $\omega = 0$. The imaginary part is odd due to $\sin(\omega t) = -\sin(-\omega t)$. It follows that the magnitude spectrum is an even function and the phase spectrum is an odd function.

We have seen that the FT is very similar to the IFT; thus, if $X(j\omega)$ is the FT of the signal $x(t)$, then it would also be true that $2\pi \cdot x(-j\omega)$ is the transform of $x(t)$. This can be shown by substituting t with $-\omega$ in the IFT:

$$\begin{aligned} x(t) &= \frac{1}{2\pi} \int_{-\infty}^{\infty} X(j\omega) e^{j\omega t} d\omega, \\ x(-t) &= \frac{1}{2\pi} \int_{-\infty}^{\infty} X(j\omega) e^{-j\omega t} d\omega, \\ x(-j\omega) &= \frac{1}{2\pi} \int_{-\infty}^{\infty} X(t) e^{-j\omega t} dt. \end{aligned} \quad (\text{B.21})$$

B.1.7 Time and Frequency Scaling

The FT of a signal modified in the time domain by scaling the time axis $y(t) = x(c \cdot t)$ will be scaled inversely:

$$Y(j\omega) = \frac{1}{|c|} X\left(j\frac{\omega}{c}\right). \quad (\text{B.22})$$

The derivation (for positive c) is

$$\begin{aligned}
 Y(j\omega) &= \int_{-\infty}^{\infty} x(c \cdot t) e^{-j\omega t} dt \\
 &= \int_{-\infty}^{\infty} x(\tau) e^{-j\omega \frac{\tau}{c}} d\frac{\tau}{c} \\
 &= \frac{1}{c} \int_{-\infty}^{\infty} x(\tau) e^{-j\frac{\omega}{c}\tau} d\tau \\
 &= \frac{1}{c} X\left(j\frac{\omega}{c}\right).
 \end{aligned} \tag{B.23}$$

For negative c , the result is $Y(j\omega) = -\frac{1}{c}X(j\frac{\omega}{c})$. The spectrum of a stretched signal ($c > 1$) will thus be compressed and vice versa.

From the above equation it directly follows for $c = -1$ that

$$\mathfrak{F}\{x(-t)\} = X(-j\omega) \tag{B.24}$$

and for a real-valued signal $x(t)$

$$\mathfrak{F}\{x(-t)\} = X^*(j\omega). \tag{B.25}$$

B.1.8 Derivatives

The transform of the n th derivative of the signal has the following property (without derivation):

$$\mathfrak{F}\left\{\frac{d^n x(t)}{dt^n}\right\} = (j\omega)^n X(j\omega). \tag{B.26}$$

B.2 Spectrum of Example Time Domain Signals

B.2.1 Delta Function

The *delta function* $\delta(t)$, sometimes also named *dirac impulse* or *delta impulse*, equals zero for all points in time except $t = 0$. It represents an ideal impulse and is defined by

$$\int_{-\infty}^{\infty} \delta(t) dt = 1, \tag{B.27}$$

$$\delta(t) = 0 \text{ for all } t \neq 0. \tag{B.28}$$

This also means that the integration of the multiplication of signal $x(t)$ with this delta function results in

$$\int_{-\infty}^{\infty} x(t) \cdot \delta(t) dt = x(0). \tag{B.29}$$

Thus, the result of the FT is

$$\Delta(j\omega) = \int_{-\infty}^{\infty} \delta(t) e^{-j\omega t} dt = e^{j\omega \cdot 0} = 1. \tag{B.30}$$

The spectrum is therefore a real-valued constant; it follows that the delta function incorporates all frequencies with the same strength.

B.2.2 Constant

The symmetry of FT and IFT shown in Eq. (B.21), in combination with Eq. (B.30), tells us also that the spectrum of a constant valued time domain signal $x(t) = 1/2\pi$ will be $X(j\omega) = \delta(\omega)$.

B.2.3 Cosine

A sinusoidal time domain signal can be interpreted as a modulated constant value. Applying the frequency shift property from Eq. (B.15) thus shows that the spectrum of a cosine is the spectrum of a constant value shifted by the cosine's frequency ω_0 , the delta function $\delta(\omega - \omega_0)$.

B.2.4 Rectangular Window

The rectangular window is defined by

$$w_R(t) = \begin{cases} 1, & -\frac{1}{2} \leq t \leq \frac{1}{2} \\ 0, & \text{otherwise} \end{cases}. \quad (\text{B.31})$$

The spectrum of this window function is

$$\begin{aligned} W_R(j\omega) &= \int_{-\infty}^{\infty} w_R(t)e^{-j\omega t} dt \\ &= \int_{-1/2}^{1/2} e^{-j\omega t} dt \\ &= \frac{1}{-j\omega} \underbrace{(e^{-j\omega/2} - e^{j\omega/2})}_{=-2j \sin(\omega/2)} \\ &= \frac{\sin(\omega/2)}{\omega/2} = \text{sinc}\left(\frac{\omega}{2}\right). \end{aligned} \quad (\text{B.32})$$

B.2.5 Delta Pulse

The *delta pulse* is a series of individual delta impulses, i.e., a superposition of delta functions. It is defined by

$$\delta_T(t) = \sum_{i=-\infty}^{\infty} \delta(t - iT_0). \quad (\text{B.33})$$

Each delta impulse has a distance T_0 from its neighbor. Using FT of a delta function given by Eq. (B.27) and the superposition property from Eq. (B.3) in combination with the time shift property from Eq. B.13, the FT of $\delta_T(t)$ is

$$\Delta_T(j\omega) = \sum_{i=-\infty}^{\infty} e^{-j\omega iT_0}. \quad (\text{B.34})$$

With help from the geometric series it can be shown [OL07] that

$$\begin{aligned} \Delta_T(j\omega) &= \frac{2\pi}{T} \sum_{i=-\infty}^{\infty} \delta\left(\omega - \frac{2\pi i}{T} j\omega T\right) \\ &= \omega_T \delta_{\omega_T}(\omega). \end{aligned} \quad (\text{B.35})$$

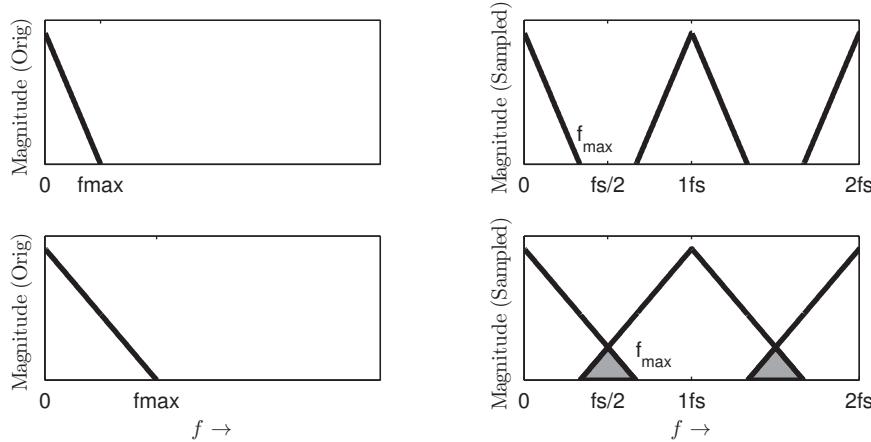


Figure B.1: Schematic visualization of the spectrum of a continuous time domain signal (left) and the sampled signal in accordance (top) and violation (bottom) of the sampling theorem

B.3 Transformation of Sampled Time Signals

A sampled time signal can be represented by the multiplication of a continuous time signal $x(t)$ multiplied by a delta pulse $\delta_T(t)$. Equation (B.8) states that a multiplication of two time signals corresponds to the convolution of their frequency transforms. This means that

$$\begin{aligned} \mathfrak{F}\{x(i)\} &= \mathfrak{F}\{x(t) \cdot \delta_T(t)\} \\ &= \mathfrak{F}\{x(t)\} * \mathfrak{F}\{\delta_T(t)\} \\ &= X(j\omega) * \Delta_T(j\omega). \end{aligned} \quad (\text{B.36})$$

Note that although the time domain signal is discrete, the resulting spectrum is still continuous. As can be seen from Eq. (B.36), the spectrum is repeated periodically with ω_T , the sample rate. This allows a very intuitive explanation of the sampling theorem stated in Eq. (A.2) since the periodically repeated spectra would overlap if signal $x(t)$ contains higher frequencies than $\omega_T/2$ (see Fig. B.1). In that case, reconstruction of the original signal $x(t)$ is impossible, while otherwise perfect reconstruction is possible by applying an ideal low-pass filter with a cut-off frequency of $\omega_T/2$ to the sampled signal $x(i)$. The effect of overlapping spectra is called aliasing and is visualized in Fig. B.1.

B.4 Short Time Fourier Transform of Continuous Signals

Up to this point, we have dealt mostly with signals unlimited in time. In the real world, signals will usually have a defined start and stop time. We might also be interested in transforming only segments of such signals. This can be seen as multiplying an infinite time signal with a window function that equals zero outside the time boundaries of interest. In signal analysis, typical segment lengths range — dependent on the task at hand — between 10 and 300 ms. Smith points out three reasons for choosing segments of this length [Smi07]:

- “Perhaps most fundamentally, the ear similarly Fourier analyzes only a short segment of audio signals at a time (on the order of 10–20 ms worth). Therefore, to match our spectrum analysis to human hearing, we desire to limit the time window of the analysis.”
- “Audio signals typically have spectra which change over time. It is therefore usually most meaningful to restrict analysis to a time window over which the spectrum stays rather constant.”
- “It can be extremely time consuming to compute the Fourier transform of an audio signal of typical length, and it will rarely fit in computer memory all at once.”

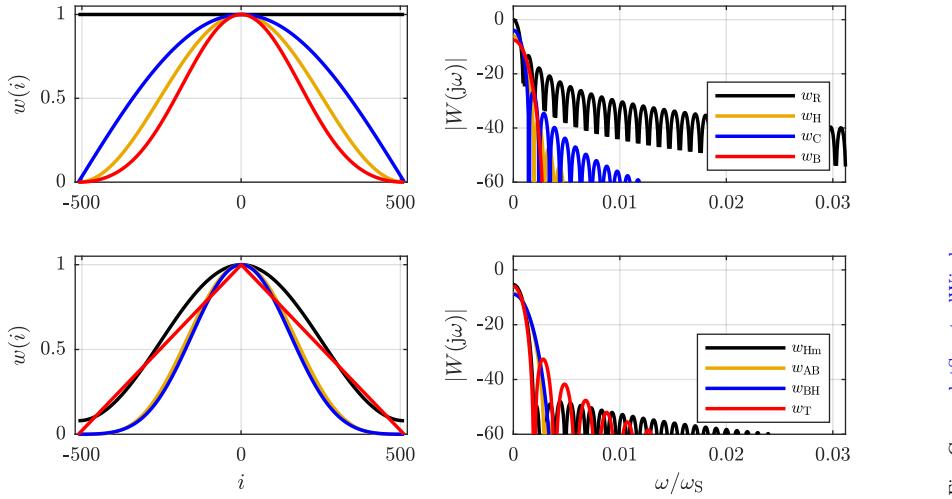
Fig. Gen.: `plotSpectralWindows.m`

Figure B.2: Windows in time domain (left) and frequency domain (right).

B.4.1 Window Functions

Since every multiplication in the time domain corresponds to a convolution of the corresponding spectra, the spectrum of the signal is convolved with the spectrum of the window. The spectrum of the window thus has influence on the resulting spectrum. The most simple window in the time domain is a rectangular window introduced in Sect. B.2.4. The typical spectral shape of a window consists of a main lobe and many side lobes with more or less decreasing amplitude.

When the signal $x(t)$ of interest is a sinusoid, then the resulting FT of the windowed signal will therefore be a superposition of two window functions with their main lobes located at the signal's frequency $\omega_0, -\omega_0, \dots$, so the delta functions are effectively “smeared” by windowing artifacts. This undesired side effect is referred to as spectral leakage. It is usually characterized by

- the width of the main lobe,
- the height of the first (closest) side lobe peak, and
- the rolloff or attenuation of the subsequent side lobe peaks.

In order to optimize these properties toward individual use cases, different window functions have been suggested in the past. Figure B.2 shows the presented window functions in time domain (left) and frequency domain (right).

B.4.1.1 Rectangular Window

The FT of the *rectangular window* has already been derived in Sect. B.2.4 to be a sinc function:

$$W_R(j\omega) = \text{sinc}\left(\frac{\omega}{2}\right) \quad (\text{B.37})$$

B.4.1.2 Bartlett Window

The *Bartlett window* has a triangular shape. It is defined by

$$\begin{aligned} w_T(t) &= \begin{cases} t+1, & -1/2 \leq t \leq 0 \\ 1-t, & 0 \leq t \leq 1/2 \\ 0, & \text{otherwise} \end{cases} \\ &= w_R(2t) * w_R(2t). \end{aligned} \quad (\text{B.38})$$

Using Eq. (B.7) it can be deduced that

$$\begin{aligned} W_T(j\omega) &= \mathfrak{F}\{w_R(2t)\} \cdot \mathfrak{F}\{w_R(2t)\} \\ &= \frac{1}{2} \cdot \text{sinc}^2\left(\frac{\omega}{4}\right) \end{aligned} \quad (\text{B.39})$$

B.4.1.3 Generalized Superposed Cosines

It is possible to generalize many window functions with

$$w_{\text{sup}}(t) = w_R(t) \sum_{j=0}^{\mathcal{O}-1} b_j \cos\left(\frac{\pi}{2}jt\right). \quad (\text{B.40})$$

Different values for \mathcal{O} result in different window families:

- $\mathcal{O} = 1$: rectangular window $w_R(t)$
- $\mathcal{O} = 2$: Hamming family of windows:
 - cosine window $w_C(t)$:
 $b_0 = 0, b_1 = 1$
 - von-Hann window $w_H(t)$:
 $b_0 = 1/2, b_1 = 1/2$
 - Hamming window $w_{\text{Hm}}(t)$:
 $b_0 = 25/46, b_1 = 42/46$
- $\mathcal{O} = 3$
 - classic Blackman window $w_B(t)$:
 $b_0 = 7938/18608, b_1 = 9240/18608, b_2 = 1430/18608$
- $\mathcal{O} = 4$
 - Blackman-Harris window $w_{\text{BH}}(t)$:
 $b_0 = 0.35875, b_1 = 0.48829, b_2 = 0.14128, b_3 = 0.01168$

B.4.1.4 Generalized Power of Cosine

A different generalization of window functions is (compare [Smi07])

$$w_{\text{pow}}(t) = w_R(t) \cos^\beta\left(\frac{\pi}{2}t\right). \quad (\text{B.41})$$

Again, different windows can be derived for different β :

- rectangular window $w_R(t)$:
 $\beta = 0$
- cosine window $w_C(t)$:
 $\beta = 1$
- von-Hann window $w_H(t)$:
 $\beta = 2$
- alternative Blackman window $w_{\text{AB}}(t)$:
 $\beta = 4$

B.5 Discrete Fourier Transform

In computer applications, a discrete representation of the signal's spectrum is required; it can only be defined at discrete frequency bins. The frequency bins are evenly distributed over the interesting range of frequencies with the distance

$$\Delta\Omega = \frac{2\pi}{\mathcal{K}}. \quad (\text{B.42})$$

The DFT of the n th block of the signal $x(i)$ will be referred to as STFT and is defined by

$$X(k) = \sum_{i=0}^{\mathcal{K}-1} x(i) \exp\left(-jki\frac{2\pi}{\mathcal{K}}\right) \quad (\text{B.43})$$

with $k = 0, 1, \dots, \mathcal{K} - 1$.

Thus, the DFT of a block of samples of length \mathcal{K} also consists of exactly \mathcal{K} complex values; however, since the signal $x(i)$ is real, the result will be symmetric with

$$X(\mathcal{K} - k) = X^*(k) \quad (\text{B.44})$$

and only $\mathcal{K}/2$ complex results need to be computed.

The spectrum $X(k, n)$ can be interpreted as the (continuous) FT of the block n of signal $x(i)$ sampled at equidistant bins at the positions $k \cdot \Delta\Omega$. It has to be periodic

$$X(k) = X(k + \mathcal{K}) \quad (\text{B.45})$$

because the time domain signal is discrete.

The spectrum can only be discrete if the time domain signal is periodic (compare the Fourier series). Therefore, the DFT can be interpreted as the FT applied to the current block of samples periodically continued.

The *Inverse Discrete Fourier Transform (IDFT)* allows reconstruction of the time samples that had been transformed:

$$x(i) = \sum_{k=0}^{\mathcal{K}-1} X(k) e^{jki\Delta\Omega}. \quad (\text{B.46})$$

The properties of the DFT correspond to the properties introduced for the continuous FT, but a few details have to be kept in mind: the multiplication of two DFTs corresponds to a circular convolution (similar to the CiCF) in the time domain. The same is true for time and frequency shift operations.

B.5.1 Window Functions

The discrete window functions are sampled (a potentially shifted) versions of the continuous window functions given above. The rectangular window is

$$w_R(i) = \begin{cases} 1, & -\frac{\mathcal{K}-1}{2} \leq i \leq \frac{\mathcal{K}-1}{2} \\ 0, & \text{otherwise} \end{cases}, \quad (\text{B.47})$$

and the superposed cosine window is

$$w_{\text{sup}}(i) = w_R(i) \sum_{j=0}^{\mathcal{O}-1} a_j \cos\left(\frac{j \cdot \pi}{\mathcal{K}} i\right). \quad (\text{B.48})$$

The DFT of a rectangular window is

$$W_R(k, n) = \exp\left(-j\frac{\mathcal{K}-1}{2} \frac{2\pi k}{\mathcal{K}}\right) \cdot \frac{\sin\left(\frac{\mathcal{K} \frac{2\pi k}{\mathcal{K}}}{2}\right)}{\sin\left(\frac{\frac{2\pi k}{\mathcal{K}}}{2}\right)}. \quad (\text{B.49})$$

Table B.1: Frequency domain properties of the most common windows (from [Har78])

Window	ΔB [Bins]	A_{SL} [dB]	S_{SL} [dB/Oct]	A_{WC} [dB]
w_R	0.89	-13	-6	3.92
w_T	1.28	-27	-12	3.07
w_C	1.20	-23	-12	3.01
w_H	1.44	-32	-18	3.18
w_{Hm}	1.30	-43	-6	3.10
w_B	1.68	-58	-18	3.47
w_{AB}	1.66	-39	-24	3.47
w_{BH}	1.66	-67	-6	3.45

Note that the phase shift term originates in moving the first window sample to sample 0. When transforming a windowed sine, the result will be shifted in the frequency domain

$$X(k, n) = \exp\left(-j\frac{\mathcal{K}-1}{2}\frac{2\pi k}{\mathcal{K}} - \Omega_0\right) \cdot \frac{\sin\left(\frac{\mathcal{K}^2\pi k}{2} - \Omega_0\right)}{\sin\left(\frac{2\pi k}{2} - \Omega_0\right)}. \quad (\text{B.50})$$

If the sinusoidal frequency exactly fits the frequency of a bin with index k , then all bins will be zero for $k \neq k_0$. In this case, all the zero crossings of the window function fall at the spectral bin positions. However, if k_0 is between two frequency bins, then two artifacts appear: the main peak has lower level, the so-called *process loss*, and the frequency bins are now directly located at the main peaks of the side lobes. Two special cases are the *best case* and the *worst case* scenario with k_0 being directly on a bin or exactly between two bins. In the time domain, the best case means that one or more periods of the sinusoidal fit exactly into the window with length \mathcal{K} .

B.5.1.1 Discrete Window Properties

The following properties can be used to characterize the frequency domain representation of a window function:

- width of main lobe ΔB (3 dB bandwidth in bins),
- peak level of highest side lobe A_{SL} (dB),
- side lobe fall-off S_{SL} (dB/Oct),
- worst case process loss A_{WC} (dB).

A smaller main lobe width yields better frequency resolution and both a smaller side lobe peak level and higher side lobe fall-off results in less cross-talk between sinusoids of different frequencies. The smaller the worst case process loss, the higher the resulting amplitude accuracy.

Table B.1 summarizes these properties for common windows. For detailed introductions to spectral leakage see Harris [Har78] and Smith [Smi07].

B.5.2 Fast Fourier Transform

An efficient way to compute the DFT is the *Fast Fourier Transform (FFT)*. The FFT is equivalent to the “normal” DFT; it just computes the result more efficiently. More specifically, the difference in the number of operations is approximately $O(\mathcal{K}^2)$ for the normal DFT compared to $O(\mathcal{K} \log \mathcal{K})$ for the FFT (compare [CT65, CCF⁺67]). There are different algorithms to compute the FFT; most FFT implementations require an input block length which equals a power of 2.

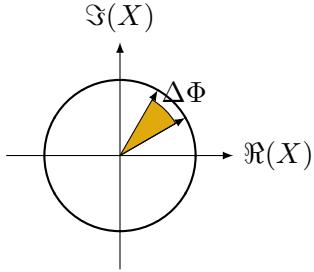


Figure B.3: add caption

B.6 Frequency Reassignment: Instantaneous Frequency

Frequency reassignment is a method to virtually improve the spectral resolution by mapping the frequency data to coordinates closer to its “true” coordinates. Frequency reassignment is not a fundamental frequency detection technique but can help to increase the frequency accuracy of spectral frequency analysis by utilizing the phase spectrum.

The reassignment process may basically cover both frequency reassignment as well as time reassignment. For an overview of (time-) frequency reassignment, see Fulop and Fitz [FF06b] and Lagrange [LM07]. The focus will be exclusively on frequency reassignment in the following.

The basic idea of frequency reassignment is that it is possible to estimate the frequency of a signal via its phase. The frequency is the derivative of the phase $\Phi(k, t)$:

$$\omega(k, t) = \frac{d}{dt} \Phi(k, t). \quad (\text{B.51})$$

Two common approaches are used to compute the so-called *instantaneous frequency* from the discrete STFT. The first one is to literally compute the phase difference of consecutive STFT spectra, and the second one is to compute two STFTs with different windows, namely with the second window being the derivative of the first window.

The derivation operation from Eq. (B.51) can be easily approximated by computing the difference of the phases of consecutive STFT phases:

$$\omega_I(k, n) = \frac{\Delta\Phi_u(k, n)}{\mathcal{H}} \cdot f_S. \quad (\text{B.52})$$

The actual computation, however, is not as trivial as it seems at first glance as $\Delta\Phi_u(k, n)$ represents the *unwrapped* phase difference while the computed phase spectrum gives the *wrapped* phase in the range of $] -\pi; \pi]$. In order to estimate the unwrapped phase difference we first compute the estimate of the unwrapped phase of the current analysis block by

$$\hat{\Phi}(k, n) = \Phi(k, n - 1) + \frac{2\pi k}{\mathcal{K}} \mathcal{H}. \quad (\text{B.53})$$

Since the phase $\Phi(k, n)$ is in the range of $] -\pi; \pi]$, the unwrapped phase can then be formulated as

$$\Phi_u(k, n) = \hat{\Phi}(k, n) + \text{princarg} [\Phi(k, n) - \hat{\Phi}(k, n)] \quad (\text{B.54})$$

with the *princarg* function being the principal argument of the phase with a resulting range of $] -\pi; \pi]$. The phase difference is then

$$\begin{aligned} \Delta\Phi_u(k, n) &= \Phi_u(k, n) - \Phi(k, n - 1) \\ &= \hat{\Phi}(k, n) + \text{princarg} [\Phi(k, n) - \hat{\Phi}(k, n)] - \Phi(k, n - 1). \end{aligned}$$

The final result can be retrieved by substituting $\hat{\Phi}(k, n)$ with Eq. (B.53):

$$\Delta\Phi_u(k, n) = \frac{2\pi k}{\mathcal{K}} \mathcal{H} + \text{princarg} [\Phi(k, n) - \Phi(k, n - 1) - \frac{2\pi k}{\mathcal{K}} \mathcal{H}]. \quad (\text{B.55})$$

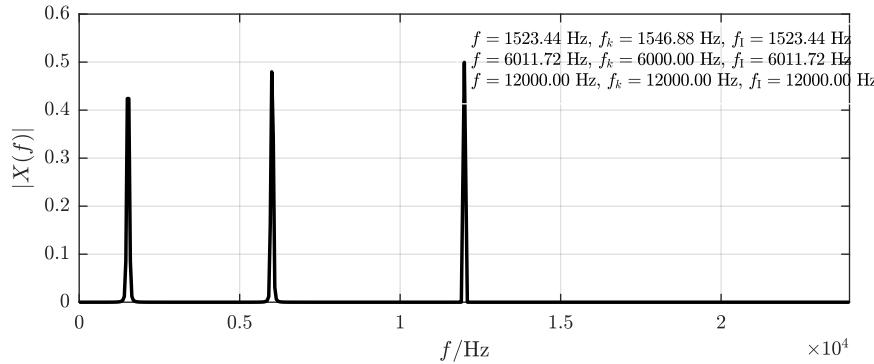
Fig. Gen.: [plotInstantaneousFreq.m](#)

Figure B.4: Magnitude spectrum of a signal composed of three sinusoids with the input frequency, the frequency of the maximum bin, and the computed instantaneous frequency.

To improve the reliability of the phase unwrapping process the hop size \mathcal{H} should be as small as possible.

An alternative to computing the phase difference directly is to use two different windows for computing two STFTs of one analysis block. Then, the instantaneous frequency can be computed by

$$\omega_I(k, n) = \omega(k) + \Im \left\{ \frac{X_D(k, n) \cdot X^*(k, n)}{|X(k, n)|^2} \right\} \quad (\text{B.56})$$

with $X_D(k, n)$ being the STFT computed using the derivative of the window used for the calculation of $X(k, n)$ [AF95]. This approach is independent of the hop size but requires the computation of two DFTs per block.

Figure B.4 shows the effect of computing the instantaneous frequency on a mixture of three sinusoids, one which falls exactly between frequency bins ($f = 1523.44\text{Hz}$), one which falls directly on a frequency bin ($f = 12\text{kHz}$), and one that is a quarter bin away from the closest frequency bin ($f = 6011.72\text{Hz}$), naming the actual frequencies, the frequency of the closest bin, and the computed instantaneous frequencies.

Appendix C

Principal Component Analysis

Principal Component Analysis (PCA) maps the input variables — in our case usually a vector of features \mathbf{v} — to a new coordinate system by a linear combination of the individual features:

$$\mathbf{u}(n) = \mathbf{T}^T \cdot \mathbf{v}(n). \quad (\text{C.1})$$

The resulting vector $\mathbf{u}(n)$ is the data in the new coordinate system for observation n and transformation matrix \mathbf{T}^T contains different linear combinations for the input feature vector $\mathbf{v}(n)$. The number of features in the vector will be referred to as \mathcal{F} . Formulating Eq. (C.1) not only for one observation but for a series of feature vectors \mathbf{V} leads to

$$\mathbf{U} = \mathbf{T}^T \cdot \mathbf{V}. \quad (\text{C.2})$$

The transformation matrix is a square matrix with the dimensions $\mathcal{F} \times \mathcal{F}$. It is composed of vectors defining the linear combinations of the input features:

$$\mathbf{T} = \begin{bmatrix} \mathbf{c}_0 & \mathbf{c}_1 & \dots & \mathbf{c}_{\mathcal{F}-1} \end{bmatrix}. \quad (\text{C.3})$$

The transformation matrix has the following main properties:

- the vectors \mathbf{c}_i are in the direction of the highest variance in the data and the variance is concentrated in as few output components as possible,
- the vectors \mathbf{c}_i are orthogonal to each other

$$\mathbf{c}_i^T \cdot \mathbf{c}_j = 0 \quad \forall i \neq j \quad (\text{C.4})$$

- and the transformation is invertible:

$$\mathbf{v}(n) = \mathbf{T} \cdot \mathbf{u}(n). \quad (\text{C.5})$$

Figure C.1 shows the scatter plots of two hypothetical variables x_1 and x_2 with the original axes on the left and rotated axes on the right.

Figure C.2 shows the application of PCA to the 5 audio features spectral centroid, spectral rolloff, spectral spread, zero crossing rate, peak envelope. The eigenvalues in the bottom clearly show that the two first principal components cover most of the variance in the features.

C.1 Computation of the Transformation Matrix

The first step in computing the matrix \mathbf{T} is the calculation of the feature covariance matrix:

$$\mathbf{R} = \frac{1}{\mathcal{F} - 1} \cdot (\mathbf{v} - \boldsymbol{\mu}_v) (\mathbf{v}^T - \boldsymbol{\mu}_v^T) \quad (\text{C.6})$$

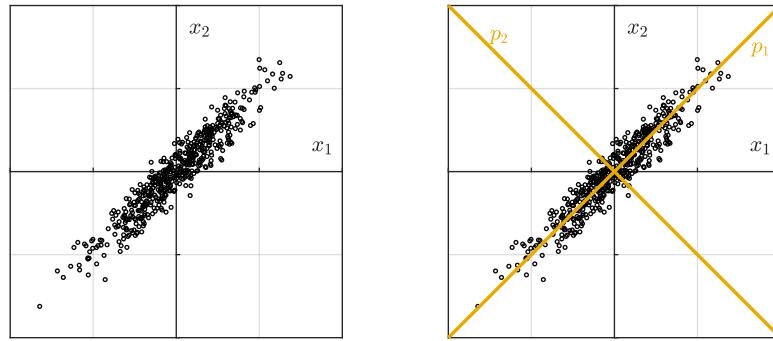
Fig. Gen.: `plotPca.m`

Figure C.1: Scatter plot of a two-dimensional data set with variables x_1, x_2 , and the rotated coordinate system after PCA with the component axes p_1, p_2 .

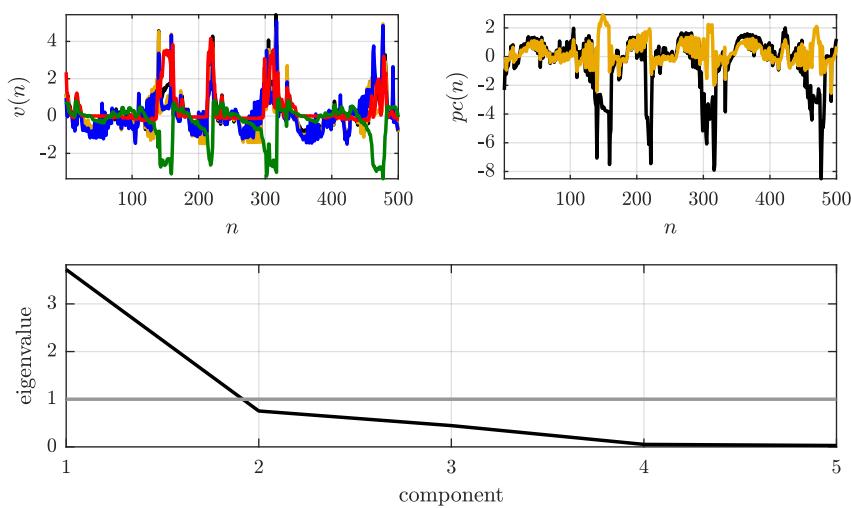
Fig. Gen.: `plotPcaExample.m`

Figure C.2: Five input features (top left) and the two resulting principal components (top right); the eigenvalues of the principal components are plotted in the bottom.

with the vector μ_v containing the arithmetic mean of each feature. The covariance matrix is square, symmetric, and has only positive entries.

The eigenvectors of the covariance matrix represent the axes of the new coordinate system and thus comprise the transformation matrix. Usually, they are ordered with decreasing eigenvalues; the vector in the first column c_0 is the vector with the highest eigenvalue and the vector in the last column has the lowest eigenvalue.

C.2 Interpretation of the Transformation Matrix

The transformation matrix T contains useful information on the input data set. Each column is a different linear combination of the input features, and they are sorted according to their eigenvalues or, in other words, according to the variance this component contributes to the overall variance. Features with high influence on the first components can be assumed to be of higher importance than features with high influence on the last components. Thus, PCA can be useful for both feature subset selection and feature space transformation. As the last components have only limited impact on the result, they might be discarded. The matrix T can then be truncated from the dimensions $\mathcal{F} \times \mathcal{F}$ to $\mathcal{F} \times \mathcal{L}$ with \mathcal{L} being the required number of components in the space transformation process.

Appendix D

Linear Regression

The minimum can be found using the first derivative of the squared error with respect to both b and m . For the bias b , we set

$$0 = \frac{\partial}{\partial b} \sum_{r=0}^{\mathcal{R}-1} (y(r) - \hat{y}(r))^2 \quad (\text{D.1})$$

$$0 = \frac{\partial}{\partial b} \sum_{r=0}^{\mathcal{R}-1} (y(r) - mv(r) - b)^2 \quad (\text{D.2})$$

$$0 = \sum_{r=0}^{\mathcal{R}-1} -2(y(r) - mv(r) - b) \quad (\text{D.3})$$

$$b = \frac{\sum_{r=0}^{\mathcal{R}-1} y(r) - m \sum_{r=0}^{\mathcal{R}-1} v(r)}{\mathcal{R} - 1} \quad (\text{D.4})$$

$$b = \mu_y - m \cdot \mu_v \quad (\text{D.5})$$

Similarly, the slope can be determined with

$$0 = \frac{\partial}{\partial m} \sum_{r=0}^{\mathcal{R}-1} (y(r) - mv(r) - b)^2 \quad (\text{D.6})$$

$$0 = \sum_{r=0}^{\mathcal{R}-1} -2(y(r) - mv(r) - b) \cdot v(r) \quad (\text{D.7})$$

$$0 = \sum_{r=0}^{\mathcal{R}-1} y(r)v(r) - mv^2(r) - bv(r) \quad (\text{D.8})$$

$$0 = \sum_{r=0}^{\mathcal{R}-1} y(r)v(r) - mv^2(r) - (\mu_y - m \cdot \mu_v)v(r) \quad (\text{D.9})$$

$$0 = \sum_{r=0}^{\mathcal{R}-1} y(r)v(r) - mv^2(r) - \mu_y \cdot v(r) + m \cdot \mu_v \cdot v(r) \quad (\text{D.10})$$

$$0 = \sum_{r=0}^{\mathcal{R}-1} y(r)v(r) - \mu_y \cdot v(r) - m \sum_{r=0}^{\mathcal{R}-1} v^2(r) - \mu_v \cdot v(r) \quad (\text{D.11})$$

$$m = \frac{\sum_{r=0}^{\mathcal{R}-1} (y(r) - \mu_y) \cdot v(r)}{\sum_{r=0}^{\mathcal{R}-1} (v(r) - \mu_v) \cdot v(r)} \quad (\text{D.12})$$

$$m = \frac{\sum_{r=0}^{\mathcal{R}-1} (y(r) - \mu_y) \cdot (v(r) - \mu_v)}{\sum_{r=0}^{\mathcal{R}-1} (v(r) - \mu_v)^2} = \frac{\sigma_{vy}^2}{\sigma_{vv}^2} \quad (\text{D.13})$$

Appendix E

Software for Audio Analysis

this becomes outdated too fast. Maybe replace whole chapter with a list of datasets? The process of developing software for audio analysis can generally be split into two steps, the algorithmic design including the prototyping as well as the evaluation and the implementation of “production-quality” software which is made available to customers and users. The final software is commonly implemented in the programming languages *C++* [14803] and *Java*.¹ Many developers consider *C++* as the language that allows the most workload-efficient implementations and *Java* as the language that allows a comparably rapid development cycle.

While *C++* and *Java* are also used for algorithm prototyping, other languages such as *Python*,² often extended by the packages *NumPy*, *SciPy*, and *IPython*,³ are more common. *Matlab*⁴ (or the largely compatible open-source software *Octave*⁵) provides an environment with a simple script language with a large set of already included routines as well as various possibilities of data visualization. Furthermore, visual audio programming environments such as *Max*⁶ or *Pure Data (PD)*⁷ are used especially in the context of designing real-time audio algorithms.

The aim of this appendix is to provide a short overview about software supporting the design and implementation of audio analysis algorithms. The solutions presented below include toolboxes, libraries, frameworks, and applications. They cover a multitude of ACA-related tasks, and each solution focuses on improving or accelerating the prototyping or development process in different areas. The main areas of interest can be identified as

- *annotation* of (audio) files and generation of ground truth data,
- *feature extraction*,
- *pattern recognition* and machine learning algorithms, and
- *visualization* of features and properties of (collections of) audio files.

The software may also differ in the level of user expertise (developer vs. non-technical user), in its real-time capabilities, and in the input data sources (e.g., audio vs. MIDI).

E.1 Software Frameworks and Applications

This section presents software frameworks offering comprehensive possibilities for file input and output, signal processing, audio analysis, and machine learning. Thus, they offer the prototyping and possibly the building of

¹Java. <http://www.java.com>. Last retrieved on Jan. 28, 2012.

²Python. <http://www.python.org>. Last retrieved on Jan. 28, 2012.

³NumPy. <http://numpy.scipy.org>, SciPy. <http://www.scipy.org>, IPython. <http://ipython.org>. Last retrieved on Jan. 28, 2012.

⁴Matlab. <http://www.mathworks.com/products/matlab>. Last retrieved on Jan. 28, 2012.

⁵Octave. <http://www.octave.org>. Last retrieved on Jan. 28, 2012.

⁶Max. <http://cycling74.com/products/max>. Last retrieved on Jan. 28, 2012.

⁷Pure Data. <http://puredata.info>. Last retrieved on Jan. 28, 2012.

ACA systems. These frameworks do not depend on any specific environment and do not require the usage of other software.

E.1.1 Marsyas

Marsyas is a software framework in C++ designed by George Tzanetakis. It offers both real-time and offline processing of audio data. While Marsyas features also audio synthesis and (effect) processing, its emphasis is on MIR. It is written in C++ but allows users access to configuring and using Marsyas, for instance, through a Python front end.

Since its first publication [TC00], Marsyas has grown into a powerful set of classes which allow to construct basically any MIR-related system [Tza08]. Numerous of the algorithms described in this book are already implemented in Marsyas.

The design of Marsyas is modular; it consists of processing blocks which can be composed into data flow networks. A block can represent different functionality; it can, for instance, be an audio or text file IO, a sound card audio IO or an algorithmic operation on audio samples or features. Marsyas also includes a set of machine learning tools for training and classification.

The latest information and the source code of Marsyas is available from the project's web page.⁸

E.1.2 CLAM

A software framework developed by a research team of the Music Technology Group of Pompeu Fabra University, Spain, is *C++ Framework for Audio and Music (CLAM)*. It is implemented in C++ and its emphasis is on spectral modeling and spectral processing.

The first version of CLAM was presented in 2002 [ABRG02, AAR02]. The basic design principles seem to be similar to Marsyas in the way that the researcher can build networks of individual processing blocks. Nowadays, CLAM comes with a visual network editor that allows the user to build the processing networks through a graphical user interface [Ama07]; its main focus remains on audio processing and less on audio analysis. It does, for instance, not include any machine learning blocks.

The latest information and the source code of CLAM is available from the project's web page.⁹

E.1.3 jMIR

The software framework *jMIR*, implemented in Java, is developed and maintained at the Music Technology Group at McGill University. Its focus is on machine-learning-related tasks on music such as musical genre classification and mood classification.

The framework consists of a set of tools for feature extraction and machine learning algorithms as well as tools for meta data annotation. The tools can also be used individually [MF09, MBF09]. One of the most notable differences to the functionality of other frameworks is the built-in functionality to extract features directly from symbolic data such as MIDI. It comes with both an audio and a MIDI ground truth data set for musical genre classification.

The latest information and the source code of jMIR is available from the project's web page.¹⁰

E.1.4 CoMIRVA

CoMIRVA is a Java framework for the visualization of large collections of music data. It is developed at the Johannes Kepler University Linz.

The software offers various possibilities of analyzing and visualizing the relationship (such as the similarity) of music files in large collections. This functionality is complemented by some feature extraction routines and web data mining tools [Sch06].

⁸Marsyas. <http://marsyas.info>. Last retrieved on Jan. 28, 2012.

⁹CLAM. <http://clam-project.org>. Last retrieved on Jan. 28, 2012.

¹⁰jMIR. <http://jmir.sourceforge.net>. Last retrieved on Jan. 31, 2012.

The latest information and the source code of CoMIRVA is available from the project's web page.¹¹

E.1.5 Sonic Visualiser

Sonic Visualiser is a software for the visualization of various properties of one or more audio files. The project was initiated at the Centre for Digital Music at Queen Mary University of London.

Sonic Visualiser was first presented in 2006 and has been under development since then [CLS10]. It offers numerous possibilities to visualize and annotate audio. Compared to the frameworks presented above, it does not focus as much on the algorithm developer but on (non-technical) expert users such as musicologists who need access to objective information on the audio recording. In order to be able to easily integrate new tools providing analysis data for the visualization, it offers a plugin interface called *VAMP* (see below).

The latest information and the source code of Sonic Visualiser is available from the project's web page.¹²

E.2 Software Libraries and Toolboxes

In contrast to the frameworks presented above, the following software solutions focus on specific aspects such as feature extraction. Many of them require either a programming environment (for instance, Matlab) or they are libraries to be linked against a custom-designed software.

E.2.1 Feature Extraction

The *MIRtoolbox* is a toolbox for Matlab and has been developed by a team at the University of Jyväskylä. It provides an extensive set of functions for the extraction of low-level features as well as for the analysis of tonal, structural, and temporal properties [LT07]. The latest information and the source code of the MIRtoolbox is available from the project's web page.¹³

The *libXtract* library has been presented by Bullock from the Birmingham Conservatoire [Bul07]. It is programmed in C and deals with the extraction of low-level features. The libXtract library allows to arbitrarily cascade the feature extraction routines for the computation of subfeatures on different hierarchical levels. The latest information and the source code of libXtract is available from the project's web page.¹⁴

Content-based Audio and Music Extraction Library (CAMEL) is a comparably new library for feature extraction and aggregation aspiring to become a framework for various MIR-related tasks. It is implemented in C++ and was presented by a team of the University of Lethbridge [SBZ10]. The latest information and the source code of CAMEL is available from the project's web page.¹⁵

Yet Another Audio Feature Extractor (YAAFE) is a command line tool for the extraction of low-level features published by the Telecom ParisTech [MEF⁺10]. It is implemented in C++ but provides Python bindings as well. YAAFE aims at very efficient feature extraction by utilizing a *feature plan parser*, determining and removing redundant processing steps in the calculation of different features. The latest information and the source code of YAAFE is available from the project's web page.¹⁶

The *Timbre Toolbox* is a Matlab toolbox for the extraction of a large set of (low-level) features. It is the result of a joint effort of IRCAM and McGill University [PGS⁺11]. The source code of the Timbre Toolbox is available online.¹⁷

SCMIR is an extension for the *SuperCollider* audio programming language. It is developed by Collins [Col11] and allows access to ACA routines through a high-level programming language. The emphasis is on feature extraction in order to use the extracted information for “creative musical activity.” SCMIR offers the

¹¹CoMIRVA: Collection of Music Information Retrieval and Visualization Applications. www.cp.jku.at/CoMIRVA. Last retrieved on Jan. 31, 2012.

¹²Sonic Visualiser. <http://www.sonicvisualiser.org>. Last retrieved on Jan. 31, 2012.

¹³MIRtoolbox. <https://www.jyu.fi/hum/laitokset/musiikki/en/research/coe/materials/mirtoolbox>. Last retrieved on Jan. 31, 2012.

¹⁴libXtract. <http://libxtract.sourceforge.net>. Last retrieved on Jan. 31, 2012.

¹⁵CAMEL – A Framework for Audio Analysis. <http://camel-framework.sourceforge.net>. Last retrieved on Jan. 31, 2012.

¹⁶Yaafe – audio features extraction. <http://yaafe.sourceforge.net>. Last retrieved on Jan. 31, 2012.

¹⁷<http://recherche.ircam.fr/pub/timbretoolbox>. Last retrieved on Jan. 31, 2012.

extraction of standard features, the analysis of tempo and beat, as well as structural analyses. The latest information and the source code of SCMIR is available from the project's web page.¹⁸

Maaate is an audio analysis toolkit written in C++. It was designed to perform feature extraction and analysis routines directly on encoded Motion Picture Experts Group (MPEG) Layer 1–3 files [PP01]. The latest information and the source code of Maaate is available from the project's web page.¹⁹

A library designed for feature extraction in a real-time context is *aubio*. It is written in C++ and enables the user to extract several low-level features as well as both onsets and tempo in a causal way in order to support real-time applications. The latest information and the source code of aubio is available from the project's web page.²⁰

OpenSMILE is a C++ library for feature extraction. It addresses both speech processing and music processing by combining features typically used in both domains in one feature extractor [EWS10]. The latest information and the source code of OpenSMILE is available from the project's web page.²¹

E.2.2 Plugin Interfaces

One idea to face the problem of multiple and apparently redundant implementations of the same (low-level) features in various libraries and frameworks is to define a plugin *Application Programmer's Interface (API)*. The advantage of using plugins for feature extraction is the possibility of dynamically loading new plugins at runtime without compilation or static linking, complemented by the simple exchange of plugins (features) between researchers. A plugin API itself is thus a rather general interface definition of how to extract an unknown feature as opposed to the actual implementation of various features.

E.2.2.1 FEAPI

The *Feature Extraction Application Programmer's Interface (FEAPI)*, named here for historic reasons, was a plugin API for the extraction of low-level features. It was a joint effort with participants from the four institutions Ghent University, IRCAM, Technical University of Berlin and zplane.development [LET05].

The project is not actively maintained or developed anymore. One of the most likely reasons why FEAPI was not accepted by the research community was the lack of host applications. Although a Max port existed and had been used, other (graphical) user interfaces except for a simple command line interface did not exist.

The documentation and source code of FEAPI, complemented by example implementations of several spectral and loudness features, can still be found on the FEAPI project web page.²²

E.2.2.2 VAMP

VAMP is the only reasonably widespread plugin interface for feature extraction from audio signals. It has been defined and developed by a team of the Centre for Digital Music of the Queen Mary University of London [CLSB06]. VAMP plugins can be hosted by the *Sonic Visualiser* (see above). The basic functionality is similar to the functionality of FEAPI.

Several plugins, as well as the latest information and the source code of VAMP is available from the project's web page.²³

E.2.3 Other Software

There is a multitude of other software applications and libraries that can be used in the context of ACA. Very important are the various machine learning and data mining solutions; probably the most-cited software is

¹⁸SCMIR. <http://www.sussex.ac.uk/Users/nc81/code.html>. Last retrieved on Jan. 31, 2012.

¹⁹Maaate!. <http://maaate.sourceforge.net>. Last retrieved on Jan. 31, 2012.

²⁰aubio, a library for audio labelling. <http://aubio.sourceforge.net>. Last retrieved on Jan. 31, 2012.

²¹openSMILE: The Munich Versatile and Fast Open-Source Audio Feature Extractor. <http://opensmile.sourceforge.net>. Last retrieved on Jan. 31, 2012.

²²FEAPI. <http://feapi.sf.net>. Last retrieved on Jan. 31, 2012.

²³VAMP Plugins: The Vamp audio analysis plugin system. <http://vamp-plugins.org>. Last retrieved on Jan. 31, 2012.

the *Waikato Environment for Knowledge Analysis (WEKA)* software [HFH⁺09].²⁴ WEKA is a comprehensive collection of machine learning algorithms and data pre-processing tools such as algorithms for regression, classification and clustering. *Torch* provides a Matlab-like environment for machine learning tasks [CKF11].²⁵ It can be used with the programming language *Lua*.²⁶ Other libraries focus on specific areas of machine learning; examples are *libsvm*²⁷ [CL11] and *The Spider*, a machine learning environment for Matlab.²⁸ *HTK* is a toolkit for building and manipulating hidden Markov models [YEH⁺02].²⁹ Due to its focus on speech recognition HTK also offers the extraction of various features.

²⁴Weka 3: Data Mining Software in Java. <http://www.cs.waikato.ac.nz/ml/weka>. Last retrieved on Jan. 31, 2012.

²⁵Torch 5. <http://torch5.sourceforge.net>. Last retrieved on Jan. 31, 2012.

²⁶The Programming Language Lua. <http://www.lua.org>. Last retrieved on Jan. 31, 2012.

²⁷LIBSVM – A Library for Support Vector Machines. <http://www.csie.ntu.edu.tw/~cjlin/libsvm>. Last retrieved on Jan. 31, 2012.

²⁸The Spider. <http://people.kyb.tuebingen.mpg.de/spider>. Last retrieved on Jan. 31, 2012.

²⁹HTK Speech Recognition Toolkit. <http://htk.eng.cam.ac.uk>. Last retrieved on Jan. 31, 2012.

Appendix F

Datasets

Bibliography

- [14803] 14882:2003, ISO/IEC J.: Programming languages – C++ / ISO/IEC. 2003 (14882:2003). – Standard
- [15902] 15938-4:2002, ISO/IEC J.: Information technology – Multimedia content description interface – Part 4: Audio / ISO/IEC. 2002 (15938-4:2002). – Standard
- [16:75] 16:1975, ISO: Acoustics – Standard tuning frequency (Standard musical pitch) / ISO. 1975 (16:1975). – Standard
- [45691] 45631:1991, DIN: Berechnung des Lautstärkepegels und der Lautheit aus dem Geräuschspektrum / DIN. 1991 (45631:1991). – Standard
- [AAR02] AMATRIAIN, Xavier ; ARUMÍ, Pau ; RAMÍREZ, Miguel: CLAM, Yet Another Library for Audio and Music Processing? In: *Proceedings of the 17th ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*. New York : ACM Press, November 2002. – ISBN 1-58113-626-9, 46
- [ABRG02] AMATRIAIN, Xavier ; BOER, Maarten de ; ROBLEDO, Enrique ; GARCIA, David: CLAM: An OO Framework for Developing Audio and Music Applications. In: *Proceedings of the 17th ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*. New York : ACM Press, November 2002. – ISBN 1-58113-626-9, 22
- [ACKM04] ARIFI, Vlora ; CLAUSEN, Michael ; KURTH, Frank ; MÜLLER, Meinard: Score-PCM Music Synchronization based on Extracted Score Parameters. In: *Proceedings of the 2nd International Symposium on Computer Music Modeling and Retrieval (CMMR)*. Trondheim, 2004
- [AD52] ANDERSON, Theodore W. ; DARLING, Donald A.: Asymptotic Theory of Certain “Goodness of Fit” Criteria Based on Stochastic Processes. In: *Annals of Mathematical Statistics* 23 (1952), Nr. 2, S. 193–212
- [AD90] ALLEN, Paul E. ; DANNENBERG, Roger B.: Tracking Musical Beats in Real Time. In: *Proceedings of the International Computer Music Conference (ICMC)*. Glasgow, September 1990, S. 140–143
- [AF95] AUGER, Francois ; FLANDRIN, Patrick: Improving the readability of time-frequency and time-scale representations by the reassignment method. In: *Transactions on Signal Processing* 43 (1995), Nr. 5, 1068–1089. <http://dx.doi.org/10.1109/78.382394>. – DOI 10.1109/78.382394. – ISSN 1053587X
- [AFL⁺06] ALOUPIS, Greg ; FEVENS, Thomas ; LANGERMAN, Stefan ; MATSUI, Tomomi ; MESA, Antonio ; NUNEZ, Yurai ; RAPPAPORT, David ; TOUSSAINT, Godfried T.: Algorithms for Computing Geometric Measures of Melodic Similarity. In: *Computer Music Journal* 30 (2006), Nr. 3, 67–76. <https://muse.jhu.edu/article/202595>. – ISSN 1531–5169
- [AH01] AKSOY, Selim ; HARALICK, Robert M.: Feature Normalization and Likelihood-Based Similarity Measures for Image Retrieval. In: *Pattern Recognition Letters* 22 (2001), Nr. 5, 563–582. [http://dx.doi.org/10.1016/S0167-8655\(00\)00112-4](http://dx.doi.org/10.1016/S0167-8655(00)00112-4). – DOI 10.1016/S0167-8655(00)00112-4. – ISSN 01678655

- [AHG⁺14] ABESSER, Jakob ; HASSELHORN, Johannes ; GROLLMISCH, Sascha ; DITTMAR, Christian ; LEHMANN, Andreas: Automatic Competency Assessment of Rhythm Performances of Ninth-grade and Tenth-grade Pupils. In: *Proceedings of the International Computer Music Conference (ICMC)*. Athens, 2014
- [AJ80] AERTSEN, A M H J. ; JOHANNESMA, P I M.: Spectro-Temporal Receptive Fields of Auditory Neurons in the Grassfrog. In: *Biological Cybernetics* 38 (1980), November, Nr. 4, 223–234. <http://dx.doi.org/10.1007/BF00337015>. – DOI 10.1007/BF00337015. – ISSN 0340–1200
- [ALP03] AGOSTINI, Giulio ; LONGARI, Maurizio ; POLLASTRI, Emanuele: Musical Instrument Timbres Classification with Spectral Features. In: *Journal on Applied Signal Processing* 1 (2003), S. 5–14
- [Ama07] AMATRIAIN, Xavier: CLAM: A Framework for Audio and Music Application Development. In: *IEEE Software* 24 (2007), Januar, Nr. 1, 82–85. <http://dx.doi.org/10.1109/MS.2007.8>. – DOI 10.1109/MS.2007.8. – ISSN 0740–7459
- [AP02] AUCOUTURIER, Jean-Julien ; PACHET, Francois: Music Similarity Measures: What's the Use? In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Paris, 2002
- [APF09] ALVES, David S. ; PAULUS, Jouni ; FONSECA, José: Drum Transcription from Multichannel Recordings with Non-Negative Matrix Factorization. In: *Proceedings of the European Signal Processing Conference (EUSIPCO)*. Glasgow, 2009
- [AR07] ALBADA, Sacha J. ; ROBINSON, Peter A.: Transformation of arbitrary distributions to the normal distribution with applications to EEG test-retest reliability. In: *Journal of Neuroscience Methods* 161 (2007), Nr. 2, S. 205–211
- [Ari02] ARIFI, Vlora: *Algorithmen zur Synchronisation von Musikdateien im Partitur-, MIDI- und PCM-Format*. Bonn, Rheinische Friedrich-Wilhelms-Universität, Dissertation, 2002
- [AS02] AUCOUTURIER, Jean-Julien ; SANDLER, Mark: Finding Repeating Patterns in Acoustic Musical Signals: Applications for Audio Thumbnailing, Audio Engineering Society, Juni 2002
- [ASA60] ASA: Acoustical Terminology / American Standards Association (ASA). 1960. – Standard
- [AYS17] ALJANAKI, Anna ; YANG, Yi-Hsuan ; SOLEYMANI, Mohammad: Developing a Benchmark for Emotional Analysis of Music. In: *PLOS ONE* 12 (2017), März, Nr. 3, e0173392. <http://dx.doi.org/10.1371/journal.pone.0173392>. – DOI 10.1371/journal.pone.0173392. – ISSN 1932–6203. – Publisher: Public Library of Science
- [BAK12] BÖCK, Sebastian ; ARZT, Andreas ; KREBS, Florian: Online Real-time Onset Detection with Recurrent Neural Networks. In: *Proceedings of the International Conference on Digital Audio Effects (DAFx)*. York, 2012, S. 4
- [Bar88] BARTLETT, James C.: Scale Structure and Similarity of Melodies. In: *Music Perception* 5 (1988), Nr. 3, S. 285–314
- [Bar93] BARNES, Arthur E.: Instantaneous Spectral Bandwidth and Dominant Frequency with Applications to Seismic Reflection Data. In: *GEOPHYSICS* 58 (1993), März, Nr. 3, 419–428. <http://dx.doi.org/10.1190/1.1443425>. – DOI 10.1190/1.1443425. – ISSN 0016–8033. – Publisher: Society of Exploration Geophysicists
- [BB19] BITTNER, Rachel ; BOSCH, Juan J.: Generalized Metrics for Single-F0 Estimation Evaluation. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Delft, Netherlands, 2019, S. 8

- [BBV10] BERTIN, Nancy ; BADEAU, Roland ; VINCENT, Emmanuel: Enforcing Harmonicity and Smoothness in Bayesian Non-Negative Matrix Factorization Applied to Polyphonic Music Transcription. In: *IEEE Transactions on Audio, Speech, and Language Processing* 18 (2010), März, Nr. 3, S. 538–549. <http://dx.doi.org/10.1109/TASL.2010.2041381>. – DOI 10.1109/TASL.2010.2041381. – ISSN 1558–7924
- [BBY17] BOZKURT, Barış ; BAYSAL, Ozan ; YÜRET, Deniz: A Dataset and Baseline System for Singing Voice Assessment. In: *Proceedings of the International Symposium on CMMR*. Matosinhos, 2017
- [BBZ90] BAIRD, Bridget ; BLEVINS, Donald ; ZAHLER, Noel: The Artificially Intelligent Computer Performer: The Second Generation. In: *Interface – Journal of New Music Research* 19 (1990), S. 197–204
- [BBZ93] BAIRD, Bridget ; BLEVINS, Donald ; ZAHLER, Noel: Artificial Intelligence and Music: Implementing an Interactive Computer Performer. In: *Computer Music Journal* 17 (1993), Nr. 2, S. 73–79
- [BC64] Box, George E P. ; Cox, David R.: An Analysis of Transformations. In: *Journal of the Royal Statistical Society* 26 (1964), Nr. 2, S. 211–252
- [BD85] BLOCH, Joshua J. ; DANNENBERG, Roger B.: Real-Time Computer Accompaniment of Keyboard Performances. In: *Proceedings of the International Computer Music Conference (ICMC)*. Vancouver, 1985
- [BDA⁺05] BELLO, Juan P. ; DAUDET, Laurent ; ABDALLAH, Samer ; DUXBURY, Chris ; DAVIES, Mike ; SANDLER, Mark B.: A Tutorial on Onset Detection in Music Signals. In: *IEEE Transactions on Speech and Audio Processing* 13 (2005), Nr. 5, 1035–1047. <http://dx.doi.org/10.1109/TSA.2005.851998>. – DOI 10.1109/TSA.2005.851998. – ISSN 1063–6676
- [BDDE19] BENETOS, Emmanouil ; DIXON, Simon ; DUAN, Zhiyao ; EWERT, Sebastian: Automatic Music Transcription: An Overview. In: *IEEE Signal Processing Magazine* 36 (2019), Januar, Nr. 1, S. 20–30. <http://dx.doi.org/10.1109/MSP.2018.2869928>. – DOI 10.1109/MSP.2018.2869928. – ISSN 1558–0792. – Conference Name: IEEE Signal Processing Magazine
- [BDK19] BÖCK, Sebastian ; DAVIES, Matthew E. ; KNEES, Peter: Multi-Task Learning of Tempo and Beat: Learning One to Improve the Other. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Delft, Netherlands, 2019
- [Bel72] BELLMAN, Richard: *Dynamic Programming*. 6. Princeton, NJ : Princeton Univ. Pr, 1972. – ISBN 978–0–691–07951–6
- [Ben12] BENGIO, Yoshua: Deep Learning of Representations for Unsupervised and Transfer Learning. In: *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, JMLR Workshop and Conference Proceedings, Juni 2012, 17–36. – ISSN: 1938-7228
- [Ber42] BERKSON, Joseph: Tests of Significance Considered as Evidence. In: *Journal of the American Statistical Association* 37 (1942), Nr. 219, S. 325–335
- [BFD15] BURGOYNE, John A. ; FUJINAGA, Ichiro ; DOWNIE, J. S.: Music Information Retrieval. Version: 2015. <http://dx.doi.org/10.1002/9781118680605.ch15>. In: SCHREIBMAN, Susan (Hrsg.) ; SIEMENS, Ray (Hrsg.) ; UNSWORTH, John (Hrsg.): *A New Companion to Digital Humanities*. John Wiley & Sons, Ltd, 2015. – DOI 10.1002/9781118680605.ch15. – ISBN 978–1–118–68060–5, 213–228
- [BGV92] BOSER, Bernhard E. ; GUYON, Isabelle M. ; VAPNIK, Vladimir N.: A Training Algorithm for Optimal Margin Classifiers. In: *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*. New York, NY, USA : Association for Computing Machinery, 1992 (COLT '92). – ISBN 0–89791–497–X, 144–152. – event-place: Pittsburgh, Pennsylvania, USA

- [BHB⁺15] BERCKMANS, Dries ; HEMERYCK, Martijn ; BERCKMANS, Daniel ; VRANKEN, Erik ; WATERSCHOOT, Toon v.: Animal Sound... Talks! Real-time Sound Analysis for Health Monitoring in Livestock. In: *Proceedings of the International Symposium on Animal Environment and Welfare (ISAEW)*. Chongqing, China, 2015
- [BHM01] BROWN, Judith C. ; HOUIX, Olivier ; MCADAMS, Stephen: Feature dependence in the automatic identification of musical woodwind instruments. In: *The Journal of the Acoustical Society of America* 109 (2001), Nr. 3, 1064–1072. <http://dx.doi.org/10.1121/1.1342075>. – DOI 10.1121/1.1342075. – ISSN 00014966
- [BHP20] BRIOT, Jean-Pierre ; HADJERES, Gaëtan ; PACHET, François-David: *Deep Learning Techniques for Music Generation*. Cham : Springer International Publishing, 2020 (Computational Synthesis and Creative Systems). <http://dx.doi.org/10.1007/978-3-319-70163-9>. <http://dx.doi.org/10.1007/978-3-319-70163-9>. – ISBN 978-3-319-70162-2 978-3-319-70163-9
- [Big90] BIGAND, Emmanuel: Abstraction of Two Forms of Underlying Structure in a Tonal Melody. In: *Psychology of Music* 18 (1990), April, Nr. 1, 45–59. <http://dx.doi.org/10.1177/0305735690181004>. – DOI 10.1177/0305735690181004. – ISSN 0305–7356
- [Bis74] BISMARCK, Gottfried von: Sharpness as an Attribute of the Timbre of Steady Sounds. In: *Acustica* 30 (1974), S. 159–172
- [Bis06] BISHOP, Christopher M.: *Pattern Recognition and Machine Learning*. Berlin : Springer, 2006 (Information Science and Statistics). – ISBN 978-0-387-31073-2
- [BJ78] BOER, E de ; JONGH, H R.: On cochlear encoding: Potentialities and limitations of the reverse-correlation technique. In: *Journal of the Acoustical Society of America (JASA)* 63 (1978), Januar, Nr. 1, 115. <http://dx.doi.org/10.1121/1.381704>. – DOI 10.1121/1.381704. – ISSN 00014966
- [BKK06a] BENETOS, Emmanouil ; KOTTI, Margarita ; KOTROPOULOS, Constantine: Application of Non-negative Matrix Factorization to Musical Instrument Classification. In: *Proceedings of the International Symposium on Communications, Control and Signal Processing*. Marrakesh, 2006
- [BKK06b] BENETOS, Emmanouil ; KOTTI, Margarita ; KOTROPOULOS, Constantine: Musical Instrument Classification Using Non-negative Matrix Factorization Algorithms. In: *Proceedings of the International Symposium on Circuits and Systems (ISCAS)*, IEEE, 2006
- [BKW16] BÖCK, Sebastian ; KREBS, Florian ; WIDMER, Gerhard: Joint Beat and Downbeat Tracking with Recurrent Neural Networks. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, New York City, 2016, S. 255–261
- [BL03] BURRED, Juan J. ; LERCH, Alexander: A Hierarchical Approach to Automatic Musical Genre Classification. In: *Proceedings of the 6th International Conference on Digital Audio Effects (DAFX)*. London, September 2003
- [BL04] BURRED, Juan J. ; LERCH, Alexander: Hierarchical Automatic Audio Signal Classification. In: *Journal of the Audio Engineering Society (JAES)* 52 (2004), Nr. 7/8, 724–739. http://www.musicinformatics.gatech.edu/wp-content_nondefault/uploads/2016/10/Burred-and-Lerch-2004-Hierarchical-Automatic-Audio-Signal-Classification.pdf
- [BLBV13] BOULANGER-LEWANDOWSKI, Nicolas ; BENGIO, Yoshua ; VINCENT, Pascal: Audio Chord Recognition with Recurrent Neural Networks. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Curitiba : ISMIR, 2013

- [BLEW04] BERENZWEIG, Adam ; LOGAN, Beth ; ELLIS, Daniel P. ; WHITMAN, Brian: A Large-Scale Evaluation of Acoustic and Subjective Music-Similarity Measures. In: *Computer Music Journal* 28 (2004), Juni, Nr. 2, 63–76. <http://dx.doi.org/10.1162/014892604323112257>. – DOI 10.1162/014892604323112257. – ISSN 0148–9267
- [Blu28] BLUMENBACH, J. F.: *The Elements of Physiology*. 4th Editio. London : Longman, Rees, Orme, Brown, and Green, 1828
- [BM99] BERAN, Jan ; MAZZOLA, Guerino: Analyzing Musical Structure and Performance – A Statistical Approach. In: *Statistical Science* 14 (1999), Nr. 1, S. 47–79
- [BMG02] BATLLE, Eloi ; MASIP, Jaume ; GUAUS, Enric: Automatic Song Identification in Noisy Broadcast Audio. In: *Proceedings of the International Conference on Signal and Image Processing (SIP)*. Kauai, 2002
- [BMS03] BELLO, Juan P. ; MONTI, Giuliano ; SANDLER, Mark B.: Phase-Based Note Onset Detection for Music Signals. In: *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* Bd. 5. Hong Kong, 2003
- [BMS18] BELLO, Juan P. ; MYDLARZ, Charlie ; SALAMON, Justin: Sound Analysis in Smart Cities. Version: 2018. https://doi.org/10.1007/978-3-319-63450-0_13. In: VIRTANEN, Tuomas (Hrsg.) ; PLUMBLEY, Mark D. (Hrsg.) ; ELLIS, Dan (Hrsg.): *Computational Analysis of Sound Scenes and Events*. Cham : Springer International Publishing, 2018. – ISBN 978–3–319–63450–0, 373–397
- [BP92] BROWN, Judith C. ; PUCKETTE, Miller S.: An efficient algorithm for the calculation of a constant Q transform. In: *Journal of the Acoustical Society of America (JASA)* 92 (1992), Nr. 5, 2698. <http://dx.doi.org/10.1121/1.404385>. – DOI 10.1121/1.404385. – ISSN 00014966
- [BP05] BELLO, Juan P. ; PICKENS, Jeremy: A Robust Mid-level Representation for Harmonic Content in Music Signals. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. London, September 2005
- [BPJ02] BURGES, Christopher J C. ; PLATT, John C. ; JANA, Soumya: Extracting noise-robust features from audio data. In: *Proceedings of the International Conference on Acoustics Speech and Signal Processing (ICASSP)*, IEEE, 2002. – ISBN 0–7803–0946–4
- [BR99] BARRETT, Lisa F. ; RUSSELL, James A.: The Structure of Current Affect: Controversies and Emerging Consensus. In: *Current Directions in Psychological Science* 8 (1999), Februar, Nr. 1, 10–14. <http://dx.doi.org/10.1111/1467-8721.00003>. – DOI 10.1111/1467–8721.00003. – ISSN 0963–7214, 1467–8721
- [Bre94] BREGMAN, Albert S.: *Auditory Scene Analysis*. MIT Press, 1994
- [Bre01] BREIMAN, Leo: Random Forests. In: *Machine Learning* 45 (2001), Oktober, Nr. 1, 5–32. <http://dx.doi.org/10.1023/A:1010933404324>. – DOI 10.1023/A:1010933404324. – ISSN 1573–0565
- [Bri98] BRINER, Ermanno: *Musikinstrumentenführer*. 3rd Editio. Stuttgart : Philipp Reclam jun., 1998
- [Bro91] BROWN, Judith C.: Calculation of a constant Q spectral transform. In: *Journal of the Acoustical Society of America (JASA)* 89 (1991), Nr. 1, 425–434. <http://dx.doi.org/10.1121/1.400476>. – DOI 10.1121/1.400476. – ISSN 00014966
- [Bro93] BROWN, Judith C.: Determination of the meter of musical scores by autocorrelation. In: *Journal of the Acoustical Society of America (JASA)* 94 (1993), Nr. 4, 1953. <http://dx.doi.org/10.1121/1.407518>. – DOI 10.1121/1.407518. – ISSN 00014966

- [Bro99] BROWN, Judith C.: Computer Identification of Musical Instruments Using Pattern Recognition with Cepstral Coefficients as Features. In: *Journal of the Acoustical Society of America (JASA)* 105 (1999), März, Nr. 3, 1933–41. <http://www.ncbi.nlm.nih.gov/pubmed/10089614>. – ISSN 0001-4966
- [BS.86] BS.468:1986, ITU-R.: Measurement of Audio-Frequency Noise Voltage Level in Sound / ITU. 1986. – Recommendation
- [BS05] BURRED, Juan J. ; SIKORA, Thomas: On the Use of Auditory Representations for Sparsity-Based Sound Source Separation. In: *5th International Conference on Information Communications & Signal Processing*. Bangkok : Ieee, 2005. – ISBN 0-7803-9283-3, 1466–1470
- [BS.06] BS.1770:2006, ITU-R.: Algorithms to measure audio programme loudness and true-peak audio level / ITU. 2006. – Recommendation
- [BS12] BÖCK, Sebastian ; SCHEDL, Markus: Polyphonic Piano Note Transcription with Recurrent Neural Networks. In: *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Kyoto, Japan, März 2012, S. 121–124. – ISSN: 2379-190X
- [BSD⁺14] BIMBOT, Frédéric ; SARGENT, Gabriel ; DERUTY, Emmanuel ; GUICHAOUA, Corentin ; VINCENT, Emmanuel: Semiotic Description of Music Structure: An Introduction to the Quaero/Metiss Structural Annotations, Audio Engineering Society, Januar 2014
- [BSWH09] BOGDANOV, Dmitry ; SERRÀ, Joan ; WACK, Nicolas ; HERRERA, Perfecto: From Low-Level to High-Level: Comparative Study of Music Similarity Measures. In: *Proceedings of the 11th International Symposium on Multimedia* (2009), 453–458. <http://dx.doi.org/10.1109/ISM.2009.72>. – DOI 10.1109/ISM.2009.72
- [Bul07] BULLOCK, Jamie: LIBXTRACT: A Lightweight Library for Audio Feature Extraction. In: *Proceedings of the International Computer Music Conference (ICMC)*. Copenhagen, August 2007, S. 3–6
- [BW01] BARTSCH, Mark A. ; WAKEFIELD, Gregory H.: To Catch a Chorus: Using Chroma-Based Representations for Audio Thumbnailing. In: *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. New Paltz : IEEE, 2001
- [BWF11] BURGOYNE, John A. ; WILD, Jonathan ; FUJINAGA, Ichiro: An Expert Ground-Truth Set for Audio Chord Recognition and Music Analysis. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Miami, FL, 2011, S. 6
- [CB14] CHO, Taemin ; BELLO, Juan P.: On the Relative Importance of Individual Components of Chord Recognition Systems. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 22 (2014), Februar, Nr. 2, S. 477–492. <http://dx.doi.org/10.1109/TASLP.2013.2295926>. – DOI 10.1109/TASLP.2013.2295926. – ISSN 2329–9304. – Conference Name: IEEE/ACM Transactions on Audio, Speech, and Language Processing
- [CBG⁺05] CANO, Pedro ; BATLLE, Eloi ; GOMEZ, Emilia ; GOMES, Leandro De C T. ; BONNET, Madeleine: Audio Fingerprinting: Concepts And Applications. In: HALGAMUGE, Saman K. (Hrsg.) ; WANG, Lipo (Hrsg.): *Computational Intelligence for Modelling and Prediction*. Berlin : Springer, 2005, S. 233–245
- [CBKH02] CANO, Pedro ; BATLE, E. ; KALKER, T. ; HAITSMA, Jaap: A Review of Algorithms for Audio Fingerprinting. In: *Proceedings of the Workshop on Multimedia Signal Processing (MMSP)*, IEEE, 2002. – ISBN 0-7803-7713-3, 169–173

- [CBKH05] CANO, Pedro ; BATLLE, Eloi ; KALKER, Ton ; HAITSMAN, Jaap: A Review of Audio Fingerprinting. In: *The Journal of VLSI Signal Processing-Systems for Signal, Image, and Video Technology* 41 (2005), November, Nr. 3, 271–284. <http://dx.doi.org/10.1007/s11265-005-4151-3>. – DOI 10.1007/s11265-005-4151-3. – ISSN 0922-5773
- [CBMN02] CANO, Pedro ; BATLLE, Eloi ; MAYER, Harald ; NEUSCHMIED, Helmut: Robust Sound Modeling for Song Detection in Broadcast Audio. In: *Proceedings of the Audio Engineering Society Convention*. Munich : AES, 2002
- [CC05] CHUAN, Ching-Hua ; CHEW, Elaine: Polyphonic Audio Key Finding Using the Spiral Array CEG Algorithm. In: *International Conference on Multimedia and Expo*, IEEE, 2005. – ISBN 0-7803-9331-7, 21–24
- [CC19] CHOI, Keunwoo ; CHO, Kyunghyun: Deep Unsupervised Drum Transcription. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Delft, Netherlands, 2019
- [CCF⁺67] COCHRAN, William T. ; COOLEY, James W. ; FAVIN, David L. ; HELMS, Howard D. ; KAENEL, Reginald A. ; LANG, William W. ; MALING, George C. ; NELSON, David E. ; RADER, Charles M. ; WELCH, Peter D.: What is the Fast Fourier Transform? In: *Transactions on Audio and Electroacoustics* 15 (1967), Juni, Nr. 2, 45–55. <http://dx.doi.org/10.1109/TAU.1967.1161899>. – DOI 10.1109/TAU.1967.1161899. – ISSN 0018-9278
- [CCTM16] CROCCO, Marco ; CRISTANI, Marco ; TRUCCO, Andrea ; MURINO, Vittorio: Audio Surveillance: A Systematic Review. In: *ACM Computing Surveys* 48 (2016), Februar, Nr. 4, 52:1–52:46. <http://dx.doi.org/10.1145/2871183>. – DOI 10.1145/2871183. – ISSN 0360-0300
- [CD04] CREMER, Markus ; DERBOVEN, Claas: A System for Harmonic Analysis of Polyphonic Music. In: *Proceedings of the 25th International Audio Engineering Society Conference: Metadata for Audio*, 2004
- [CFSC17] CHOI, Keunwoo ; FAZEKAS, György ; SANDLER, Mark B. ; CHO, Kyunghyun: Transfer Learning for Music Classification and Regression Tasks. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Suzhou, China, 2017, 141–149
- [CGML18] CUESTA, Helena ; GÓMEZ, Emilia ; MARTORELL, Agustín ; LOÁICIGA, Felipe: Analysis of Intonation in Unison Choir singing. In: *Proceedings of the International Conference on Music Perception and Cognition (ICMPC)*. Graz, Austria : Zenodo, 2018. – Type: dataset
- [Che98] CHEW, Elaine: *Towards a Mathematical Model of Tonality*, Massachusetts Institute of Technology, Dissertation, 1998
- [Che06] CHEVEIGNÉ, Alain D.: Multiple F0 Estimation. In: WANG, DeLiang (Hrsg.) ; BROWN, Guy J. (Hrsg.): *Computational Auditory Scene Analysis*. New Jersey : IEEE Press/Wiley, 2006, S. 45–79. – 00000
- [Che10] CHEVEIGNÉ, Alain d.: Pitch Perception. Version: Januar 2010. <https://www.oxfordhandbooks.com/view/10.1093/oxfordhb/9780199233557.001.0001/oxfordhb-9780199233557-e-04>. In: PLACK, Christopher J. (Hrsg.): *Oxford Handbook of Auditory Science: Hearing*. New York : Oxford University Press, Januar 2010 (Oxford Handbook of Auditory Science). – ISBN 978-0-19-923355-7
- [CJK⁺85] CHAFE, Chris ; JAFFE, David ; KASHIMA, Kyle ; MONT-REYNAUD, Bernard ; SMITH, Julius O.: Techniques for Note Identification in Polyphonic Music. In: *Proceedings of the International Computer Music Conference (ICMC)*, ICMA, 1985
- [CK99] CHEVEIGNÉ, Alain D. ; KAWAHARA, Hideki: Multiple period estimation and pitch perception model. In: *Speech Communication* 27 (1999), S. 175–185

- [CK02] CHEVEIGNÉ, Alain D. ; KAWAHARA, Hideki: YIN, a fundamental frequency estimator for speech and music. In: *Journal of the Acoustical Society of America (JASA)* 111 (2002), Nr. 4, 1917–1930. <http://dx.doi.org/10.1121/1.1458024>. – DOI 10.1121/1.1458024. – ISSN 00014966
- [CKDH01] CEMGIL, Ali T. ; KAPPEN, Bert ; DESAIN, Peter ; HONING, Henkjan: On tempo tracking: Tempogram representation and Kalman filtering. In: *Journal of New Music Research* 28 (2001), Nr. 4, S. 259–273
- [CKF11] COLLOBERT, Ronan ; KAVUKCUOGLU, Koray ; FARABET, Clément: Torch7: A Matlab-like Environment for Machine Learning. In: *Proceedings of the NIPS Workshop on Parallel and Large-scale Machine Learning*. Sierra Nevada, 2011. – 00015
- [CL11] CHANG, Chih-chung ; LIN, Chih-jen: LIBSVM: A Library for Support Vector Machines. In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 2 (2011), Nr. 3. <http://dx.doi.org/10.1145/1961189.1961199>. – DOI 10.1145/1961189.1961199
- [CL14] COLER, Henrik v. ; LERCH, Alexander: CMMSD: A Data Set for Note-Level Segmentation of Monophonic Music. In: *Proceedings of the AES 53rd International Conference on Semantic Audio*. London, UK : Audio Engineering Society (AES), 2014
- [Cla02a] CLARKE, Eric F.: Listening to performance. In: RINK, John (Hrsg.): *Musical Performance – A Guide to Understanding*. Cambridge : Cambridge University Press, 2002
- [Cla02b] CLARKE, Eric F.: Understanding the psychology of performance. In: RINK, John (Hrsg.): *Musical Performance – A Guide to Understanding*. Cambridge : Cambridge University Press, 2002
- [Cla04] CLARKE, Eric F.: Empirical Methods in the Study of Performance. In: CLARKE, Eric (Hrsg.) ; COOK, Nicholas (Hrsg.): *Empirical Musicology*. Oxford : Oxford University Press, 2004
- [CLB99] CANO, Pedro ; LOSCOS, Alex ; BONADA, Jordi: Score-Performance Matching using HMMs. In: *Proceedings of the International Computer Music Conference (ICMC)*. Beijing, China, 1999
- [CLP⁺21] CHANG, Sungkyun ; LEE, Donmoon ; PARK, Jeongsoo ; LIM, Hyungui ; LEE, Kyogu ; KO, Karam ; HAN, Yoonchang: Neural Audio Fingerprint for High-specific Audio Retrieval based on Contrastive Learning. In: *Proceedings of the International Conference on Acoustics Speech and Signal Processing (ICASSP)*, 2021. – arXiv: 2010.11910
- [CLS10] CANNAM, Chris ; LANDONE, Christian ; SANDLER, Mark: Sonic Visualiser: An Open Source Application for Viewing, Analysing, and Annotating Music Audio Files. In: *Proceedings of the International Conference on Multimedia (MM)*. New York : ACM Press, 2010. – ISBN 978–1–60558–933–6, 1467–1468
- [CLSB06] CANNAM, Chris ; LANDONE, Christian ; SANDLER, Mark B. ; BELLO, Juan P.: The Sonic Visualiser: A Visualisation Platform for Semantic Descriptors from Musical Signals. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Victoria, 2006
- [CM63] COOPER, Grosvenor W. ; MEYER, Leonard B.: *The Rhythmic Structure of Music*. University of Chicago Press, 1963. – ISBN 978–0–226–11522–1. – Google-Books-ID: V2yXrIWDTIQC
- [CMSW05] CACLIN, Anne ; MCADAMS, Stephen ; SMITH, Bennett K. ; WINSBERG, Suzanne: Acoustic correlates of timbre space dimensions: A confirmatory study using synthetic tones. In: *Journal of the Acoustical Society of America (JASA)* 118 (2005), Nr. 1, 471–482. <http://dx.doi.org/10.1121/1.1929229>. – DOI 10.1121/1.1929229. – ISSN 00014966
- [Col11] COLLINS, Nick: SCMIR: A SuperCollider Music Information Retrieval Library. In: *Proceedings of the International Computer Music Conference (ICMC)*. Huddersfield : ICMA, 2011

- [Con06] CONT, Arshia: Realtime Audio to Score Alignment for Polyphonic Music Instruments using Sparse Non-Negative Constraints and Hierarchical HMMs. In: *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* Bd. 5. Toulouse, 2006
- [CPLT99] CAREY, Michael J. ; PARRIS, Eluned S. ; LLOYD-THOMAS, Harvey: A Comparison of Features for Speech, Music Discrimination. In: *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Phoenix, 1999
- [CPNK04] CANTÚ-PAZ, Erick ; NEWSAM, Shawn ; KAMATH, Chandrika: Feature Selection in Scientific Applications. In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. Seattle : ACM Press, August 2004. – ISBN 1581138889
- [CSSR07] CONT, Arshia ; SCHWARZ, Diemo ; SCHNELL, Norbert ; RAPHAEL, Christopher: Evaluation of Real-Time Audio-To-Score Alignment. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Vienna, 2007
- [CT65] COOLEY, James W. ; TUKEY, John W.: An Algorithm for the Machine Calculation of Complex Fourier Series. In: *Mathematics of Computation* 19 (1965), April, Nr. 90, 297. <http://dx.doi.org/10.2307/2003354>. – DOI 10.2307/2003354. – ISSN 00255718
- [CT11] CLERCQ, Trevor d. ; TEMPERLEY, David: A Corpus Analysis of Rock Harmony. In: *Popular Music* 30 (2011), Januar, Nr. 1, 47–70. <http://dx.doi.org/10.1017/S026114301000067X>. – DOI 10.1017/S026114301000067X. – ISSN 0261–1430, 1474–0095
- [CVG⁺08] CASEY, Michael A. ; VELTKAMP, Remco ; GOTO, Masataka ; LEMAN, Marc ; RHODES, Christophe ; SLANEY, Malcolm: Content-Based Music Information Retrieval: Current Directions and Future Challenges. In: *Proceedings of the IEEE* 96 (2008), April, Nr. 4, S. 668–696. <http://dx.doi.org/10.1109/JPROC.2008.916370>. – DOI 10.1109/JPROC.2008.916370. – ISSN 1558–2256. – Conference Name: Proceedings of the IEEE
- [CWSB19] CRAMER, Jason ; WU, Ho-Hsiang ; SALAMON, Justin ; BELLO, Juan P.: Look, Listen, and Learn More: Design Choices for Deep Audio Embeddings. In: *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Brighton, UK : Institute of Electrical and Electronics Engineers (IEEE), Mai 2019. – ISBN 978–1–4799–8131–1, 3852–3856
- [CYWC15] CHEN, Yu-An ; YANG, Yi-Hsuan ; WANG, Ju-Chiang ; CHEN, Homer: The AMG1608 Dataset for Music Emotion Recognition. In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, S. 693–697. – ISSN: 2379-190X
- [CZPA09] CICHOCKI, Andrzej ; ZDUNEK, Rafal ; PHAN, Anh H. ; AMARI, Shun-ichi: *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation*. John Wiley & Sons, 2009 <https://dl.acm.org/doi/abs/10.5555/1822971>. – ISBN 978–0–470–74666–0. – Archive Location: world
- [Dan84] DANNENBERG, Roger B.: An On-Line Algorithm for Real-Time Accompaniment. In: *Proceedings of the International Computer Music Conference (ICMC)*. Paris, 1984
- [Dan06] DANNENBERG, Roger B.: The Interpretation of MIDI Velocity. In: *Proceedings of the International Computer Music Conference (ICMC)*. New Orleans, November 2006
- [Dan10] DANNENBERG, Roger B.: Style in Music. Version: 2010. https://doi.org/10.1007/978-3-642-12337-5_3. In: ARGAMON, Shlomo (Hrsg.) ; BURNS, Kevin (Hrsg.) ; DUBNOV, Shlomo (Hrsg.): *The Structure of Style: Algorithmic Approaches to Understanding Manner and Meaning*. Berlin, Heidelberg : Springer, 2010. – ISBN 978–3–642–12337–5, 45–57

- [DB79] DAVIES, David L. ; BOULDIN, Donald W.: A Cluster Separation Measure. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-1 (1979), April, Nr. 2, S. 224–227. <http://dx.doi.org/10.1109/TPAMI.1979.4766909>. – DOI 10.1109/TPAMI.1979.4766909. – ISSN 1939–3539. – Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence
- [DB14] DAVIES, Matthew ; BÖCK, Sebastian: Evaluating the Evaluation Measures for Beat Tracking. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Taipei, Taiwan, 2014
- [DBDS03] DUXBURY, Chris ; BELLO, Juan P. ; DAVIES, Mike ; SANDLER, Mark B.: Complex Domain Onset Detection for Musical Signals. In: *Proceedings of the 6th International Conference on Digital Audio Effects (DAFX)*. London, 2003
- [DBS11] DIELEMAN, Sander ; BRAKEL, Philemon ; SCHRAUWEN, Benjamin: Audio-Based Music Classification with a Pretrained Convolutional Network. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Miami, FL, 2011, S. 6
- [DBVB17] DEFFERRARD, Michaël ; BENZI, Kirell ; VANDERGHEYNST, Pierre ; BRESSON, Xavier: FMA: A Dataset For Music Analysis. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Suzhou, China, September 2017. – arXiv: 1612.01840
- [DC00] DIXON, Simon ; CAMBOUROPOULOS, Emilios: Beat Tracking with Musical Knowledge. In: *Proceedings of the 14th European Conference on Artificial Intelligence (ECAI)*. Berlin, August 2000
- [DCL10] DESSEIN, Arnaud ; CONT, Arshia ; LEMAITRE, Guillaume: Real-time Polyphonic Music Transcription with Non-negative Matrix Factorization and Beta-Divergence. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Utrecht, Netherlands, 2010, 489–494
- [DDH08] DEUTSCH, Diana ; DOOLEY, Kevin ; HENTHORN, Trevor: Pitch circularity from tones comprising full harmonic series. In: *Journal of the Acoustical Society of America (JASA)* 124 (2008), Juli, Nr. 1, 589–597. <http://dx.doi.org/10.1121/1.2931957>. – DOI 10.1121/1.2931957. – ISSN 1520–8524. – 00010
- [DDLM14] DEGANI, Alessio ; DALAI, Marco ; LEONARDI, Riccardo ; MIGLIORATI, Pierangelo: Comparison of tuning frequency estimation methods. In: *Multimedia Tools and Applications* 74 (2014), Nr. 15, 5917–5934. <http://dx.doi.org/10.1007/s11042-014-1897-2>. – DOI 10.1007/s11042-014-1897-2. – ISSN 1573–7721
- [Des92] DESAIN, Peter: A (De)Composable Theory of Rhythm Perception. In: *Music Perception* 9 (1992), Nr. 4, S. 439–454
- [Dev14] DEVANEY, Johanna: Estimating Onset and Offset Asynchronies in Polyphonic Score-Audio Alignment. In: *Journal of New Music Research* 43 (2014), Juli, Nr. 3, 266–275. <http://dx.doi.org/10.1080/09298215.2014.890630>. – DOI 10.1080/09298215.2014.890630. – ISSN 0929–8215
- [Dev16] DEVANEY, Johanna: Inter-Versus Intra-Singer Similarity and Variation in Vocal Performances. In: *Journal of New Music Research* 45 (2016), Juli, Nr. 3, 252–264. <http://dx.doi.org/10.1080/09298215.2016.1205631>. – DOI 10.1080/09298215.2016.1205631. – ISSN 0929–8215
- [DG08] DANNENBERG, Roger B. ; GOTO, Masataka: Music Structure Analysis from Acoustic Signals. Version: 2008. https://doi.org/10.1007/978-0-387-30441-0_21. In: HAVELOCK, David (Hrsg.) ; KUWANO, Sonoko (Hrsg.) ; VORLÄNDER, Michael (Hrsg.): *Handbook of Signal Processing in Acoustics*. New York, NY : Springer, 2008. – ISBN 978–0–387–30441–0, 305–331

- [DG14] DITTMAR, Christian ; GÄRTNER, Daniel: Real-time Transcription and Separation of Drum Recordings based on NMF Decomposition. In: *Proceedings of the International Conference on Digital Audio Effects (DAFX)*. Erlangen, Germany, 2014, S. 8
- [DH93] DESAIN, Peter ; HONING, Henkjan: Tempo Curves Considered Harmful. In: *Time in Contemporary Musical Thought, Contemporary Music Review* 7 (1993), Nr. 2, S. 123–138. <http://dx.doi.org/10.1080/07494469300640081>. – DOI 10.1080/07494469300640081
- [DH03] DANNENBERG, Roger B. ; HU, Ning: Polyphonic Audio Matching for Score Following and Intelligent Audio Editors. In: *Proceedings of the International Computer Music Conference (ICMC)*. Singapore, 2003
- [DHH97] DESAIN, Peter ; HONING, Henkjan ; HEIJINK, Hank: Robust Score-Performance Matching: Taking Advantage of Structural Information. In: *Proceedings of the International Computer Music Conference (ICMC)*. Thessaloniki : International Computer Music Association, September 1997
- [DHS00] DUDA, Richard O. ; HART, Peter E. ; STORK, David G.: *Pattern Classification*. 2nd Editio. New York : John Wiley & Sons, Inc, 2000. – ISBN 0-471-05669-3. – 25694
- [DHYG14] DAVIES, Matthew E P. ; HAMEL, Philippe ; YOSHII, Kazuyoshi ; GOTO, Masataka: AutoMashUpper: Automatic Creation of Multi-Song Music Mashups. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 22 (2014), Dezember, Nr. 12, S. 1726–1737. <http://dx.doi.org/10.1109/TASLP.2014.2347135>. – DOI 10.1109/TASLP.2014.2347135. – ISSN 2329-9304
- [Dil01] DILLON, Roberto: Extracting audio cues in real time to understand musical expressiveness. In: *Proceedings of the MOSART workshop*. Barcelona, November 2001
- [Dil03] DILLON, Roberto: A Statistical Approach to Expressive Intention Recognition in Violin Performances. In: *Proceedings of the Stockholm Music Acoustics Conference (SMAC)*. Stockholm, August 2003
- [Dix96] DIXON, Simon: A Dynamic Modelling Approach to Music Recognition. In: *Proceedings of the International Computer Music Conference (ICMC)*. Hong Kong, August 1996
- [Dix99] DIXON, Simon: A Beat Tracking System for Audio Signals. In: *Proceedings of the Conference on Mathematical and Computational Methods in Music*. Vienna, 1999
- [Dix00] DIXON, Simon: A Lightweight Multi-Agent Musical Beat Tracking System. In: *Proceedings of the Pacific Rim International Conference on Artificial Intelligence (PRICAI)*. Melbourne, 2000
- [Dix06] DIXON, Simon: Onset Detection Revisited. In: *Proceedings of the 9th International Conference on Digital Audio Effects (DAFX)*. Montreal, September 2006
- [Dix07] DIXON, Simon: Evaluation of the Audio Beat Tracking System BeatRoot. In: *Journal of New Music Research* 36 (2007), März, Nr. 1, 39–50. <http://dx.doi.org/10.1080/09298210701653310>. – DOI 10.1080/09298210701653310. – ISSN 0929-8215. – Publisher: Routledge _eprint: <https://doi.org/10.1080/09298210701653310>
- [DJA⁺18] DORFER, Matthias ; JR, Jan H. ; ARZT, Andreas ; FROSTEL, Harald ; WIDMER, Gerhard: Learning Audio-Sheet Music Correspondences for Cross-Modal Retrieval and Piece Identification. In: *Transactions of the International Society for Music Information Retrieval* 1 (2018), September, Nr. 1, 22–33. <http://dx.doi.org/10.5334/tismir.12>. – DOI 10.5334/tismir.12. – ISSN 2514-3298. – Number: 1 Publisher: Ubiquity Press

- [DM80] DAVIS, Steven B. ; MERMELSTEIN, Paul: Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. In: *Transactions on Acoustics, Speech, and Signal Processing* 28 (1980), August, Nr. 4, 357–366. <http://dx.doi.org/10.1109/TASSP.1980.1163420>. – DOI 10.1109/TASSP.1980.1163420. – ISSN 0096–3518
- [DM88] DANNENBERG, Roger B. ; MUKAINO, Hirofumi: New Techniques for Enhanced Quality of Computer Accompaniment. In: *Proceedings of the International Computer Music Conference (ICMC)*. Cologne, September 1988
- [DMEF11] DEVANEY, Johanna ; MANDEL, Michael I. ; ELLIS, Daniel P. ; FUJINAGA, Ichiro: Automatically extracting performance data from recordings of trained singers. In: *Psychomusicology: Music, Mind and Brain* 21 (2011), Nr. 1-2, S. 108–136. <http://dx.doi.org/10.1037/h0094008>. – DOI 10.1037/h0094008. – ISSN 2162–1535(Electronic),0275–3987(Print)
- [Dor42] DORIAN, Frederick: *The History of Music in Performance – The Art of Musical Interpretation from the Renaissance to Our Day*. New York : W. W. Norton & Company Inc, 1942
- [Dow03] DOWNIE, J S.: Music Information Retrieval. In: *Annual Review of Information Science and Technology* 37 (2003), S. 295–340
- [DPW96] DELLAERT, Frank ; POLZIN, Thomas ; WAIBEL, Alex: Recognizing Emotion in Speech. In: *Proceedings of the International Conference on Spoken Language Processing (ICSLP)* Bd. 3. Philadelphia, PA, Oktober 1996, S. 1970–1973 vol.3
- [DPW03] DIXON, Simon ; PAMPALK, Elias ; WIDMER, Gerhard: Classification of Dance Music by Periodicity Patterns. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2003, S. 1–7
- [DS07] DRESSLER, Karin ; STREICH, Sebastian: Tuning Frequency Estimation using Circular Statistics. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Wien, September 2007
- [DSC08] DENG, Jeremiah D. ; SIMMERMACHER, Christian ; CRANEFIELD, Stephen: A Study on Feature Analysis for Musical Instrument Classification. In: *Transactions on Systems, Man, and Cybernetics* 38 (2008), April, Nr. 2, 429–38. <http://dx.doi.org/10.1109/TSMCB.2007.913394>. – DOI 10.1109/TSMCB.2007.913394. – ISSN 1083–4419
- [DSD02] DUXBURY, Chris ; SANDLER, Mark B. ; DAVIES, Mike: A Hybrid Approach to Musical Note Onset Detection. In: *Proceedings Of The 5th International Conference On Digital Audio Effects*. Hamburg, 2002
- [DTB11] DIXON, Simon ; TIDHAR, Dan ; BENETOS, Emmanouil: The Temperament Police: The Truth, the Ground Truth, and Nothing but the Truth. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Miami, FL, 2011, S. 7
- [DTW97] DANNENBERG, Roger B. ; THOM, Belinda ; WATSON, David: A Machine Learning Approach to Musical Style Recognition. In: *Proceedings of the International Computer Music Conference (ICMC)*. Thessaloniki, September 1997
- [DW05] DIXON, Simon ; WIDMER, Gerhard: MATCH: A Music Alignment Tool Chest. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. London, September 2005
- [EAKK11] EL AYADI, Moataz ; KAMEL, Mohamed S. ; KARRY, Fakhri: Survey on Speech Emotion Recognition: Features, Classification Schemes, and Databases. In: *Pattern Recognition* 44 (2011), März, Nr. 3, 572–587. <http://dx.doi.org/10.1016/j.patcog.2010.09.020>. – DOI 10.1016/j.patcog.2010.09.020. – ISSN 0031–3203

- [EB03a] EGGINK, Jana ; BROWN, Guy: A Missing Feature Approach to Instrument Identification in Polyphonic Music. In: *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* Bd. 5, IEEE, 2003. – ISBN 0-7803-7663-3, V-553-6
- [EB03b] EGGINK, Jana ; BROWN, Guy J.: Application of Missing Feature Theory to the Recognition of Musical Instruments in Polyphonic Audio. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2003. – 00053
- [EB04] EGGINK, Jana ; BROWN, Guy J.: Instrument recognition in accompanied sonatas and concertos. In: *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* Bd. 4, IEEE, 2004. – ISBN 0-7803-8484-9, iv-217-iv-220
- [EBSG10] EYBEN, Florian ; BÖCK, Sebastian ; SCHULLER, Björn ; GRAVES, Alex: Universal Onset Detection with Bidirectional Long-Short Term Memory Neural Networks. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Utrecht, Netherlands, 2010, S. 589–594
- [EBU10] EBU: R 128: Loudness normalisation and permitted maximum level of audio signals / EBU. 2010 (August). – Forschungsbericht
- [EE08] ESCALONA-ESPINOSA, Bernardo: *Downbeat and Meter Estimation in Audio Signals*, Technische Universität Hamburg-Harburg, Diplomarbeit, 2008
- [Eis08] EISENBERG, Gunnar: *Identifikation und Klassifikation von Musikinstrumentenklängen in monophoner und polyphoner Musik*. Göttingen : Cuvillier Verlag, 2008. – ISBN 978-3-86727-825-6
- [EJLT01] EEROLA, Tuomas ; JÄRVINEN, Topi ; LOUHVUORI, Jukka ; TOIVIAINEN, Petri: Statistical Features and Perceived Similarity of Folk Melodies. In: *Music Perception* 18 (2001), März, Nr. 3, 275–296. <http://dx.doi.org/10.1525/mp.2001.18.3.275>. – DOI 10.1525/mp.2001.18.3.275. – ISSN 0730-7829
- [EK00] ERONEN, Antti ; KLAPURI, Anssi P.: Musical Instrument Recognition using Cepstral Coefficients and Temporal Features. In: *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* Bd. 2, IEEE, 2000. – ISBN 0-7803-6293-4, II753-II756
- [Ell06] ELLIS, Daniel P W.: Extracting Information from Music Audio. In: *Communications of the ACM* 49 (2006), August, Nr. 8, 32–37. <http://dx.doi.org/10.1145/1145287.1145310>. – DOI 10.1145/1145287.1145310. – ISSN 00010782
- [Ell07] ELLIS, Daniel P. W.: Beat Tracking by Dynamic Programming. In: *Journal of New Music Research* 36 (2007), März, Nr. 1, 51–60. <http://dx.doi.org/10.1080/09298210701653344>. – DOI 10.1080/09298210701653344. – ISSN 0929-8215. – Publisher: Routledge _eprint: <https://doi.org/10.1080/09298210701653344>
- [EM11] EWERT, Sebastian ; MÜLLER, Meinard: Chroma Toolbox: Matlab Implementations for Extracting Variants of Chroma-Based Audio Features. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Miami, FL, 2011, S. 6
- [EMKPK00] EL-MALEH, Khaled ; KLEIN, Mark ; PETRUCCI, Grace ; KABAL, Peter: Speech/Music Discrimination for Multimedia Applications. In: *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Istanbul, 2000
- [EMNS20] EREΜENKO, Vsevolod ; MORSI, Alia ; NARANG, Jyoti ; SERRA, Xavier: Performance Assessment Technologies for the Support of Musical Instrument Learning. In: *Proceedings of the International Conference on Computer Supported Education (CSEDU)*. Prague, 2020, S. 12

- [ERD04] ESSID, Slim ; RICHARD, Gaël ; DAVID, Bertrand: Musical instrument recognition on solo performance. In: *Proceedings of the European Signal Processing Conference (EUSIPCO)*, 2004
- [ERD06] ESSID, S. ; RICHARD, G. ; DAVID, B.: Musical instrument recognition by pairwise classification strategies. In: *IEEE Transactions on Audio, Speech and Language Processing* 14 (2006), Juli, Nr. 4, 1401–1412. <http://dx.doi.org/10.1109/TSA.2005.860842>. – DOI 10.1109/TSA.2005.860842. – ISSN 1558–7916. – 00083
- [Ero01] ERONEN, Antti: Comparison of Features for Musical Instrument Recognition. In: *Proceedings of the Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. New Paltz, 2001
- [EV10] EEROLA, Tuomas ; VUOSKOSKI, Jonna K.: A Comparison of the Discrete and Dimensional Models of Emotion in Music. In: *Psychology of Music* 39 (2010), August, Nr. 1, 18–49. <http://dx.doi.org/10.1177/0305735610362821>. – DOI 10.1177/0305735610362821. – ISSN 0305–7356
- [Eva02] EVANGELISTA, Gianpaolo: Time and Frequency Warping Musical Signals. In: ZÖLZER, Udo (Hrsg.): *DAFX – Digital Audio Effects*. John Wiley & Sons Ltd, 2002. – ISBN 0–471–49078–4, S. 439–463
- [Eve06] EVERY, Mark R.: *Separation of musical sources and structure from single-channel polyphonic recordings*, University of York, Dissertation, 2006
- [EWS10] EYBEN, Florian ; WÖLLMER, Martin ; SCHULLER, Björn: OpenSMILE – The Munich Versatile and Fast Open-Source Audio Feature Extractor. In: *Proceedings of the International Conference on Multimedia (MM)*. New York, New York, USA : ACM Press, 2010. – ISBN 978–1–60558–933–6, 1459–1462
- [Fan73] FANT, Gunnar: *Speech Sounds and Features*. Cambridge : MIT Press, 1973. – ISBN 0–262–06051–5
- [Faw06] FAWCETT, Tom: An Introduction to ROC Analysis. In: *Pattern Recognition Letters* 27 (2006), Juni, Nr. 8, 861–874. <http://dx.doi.org/10.1016/j.patrec.2005.10.010>. – DOI 10.1016/j.patrec.2005.10.010. – ISSN 01678655
- [FBD09] FÉVOTTE, Cédric ; BERTIN, Nancy ; DURRIEU, Jean-Louis: Nonnegative Matrix Factorization with the Itakura-Saito Divergence: With Application to Music Analysis. In: *Neural Computation* 21 (2009), Nr. 3, 793–830. <http://dx.doi.org/10.1162/neco.2008.04–08–771>. – DOI 10.1162/neco.2008.04–08–771. – ISSN 0899–7667. – Publisher: MIT Press
- [FBR98] FELDMAN BARRETT, Lisa ; RUSSELL, James A.: Independence and Bipolarity in the Structure of Current Affect. In: *Journal of Personality and Social Psychology* 74 (1998), Nr. 4, 967–984. <https://psycnet.apa.org/doiLanding?doi=10.1037%2F0022-3514.74.4.967>
- [FC01] FOOTE, Jonathan T. ; COOPER, Matthew: Visualizing Musical Structure and Rhythm via Self-Similarity. In: *Proceedings of the International Computer Music Conference (ICMC)*. Habana, September 2001
- [FF06a] FIEBRINK, Rebecca ; FUJINAGA, Ichiro: Feature Selection Pitfalls and Music Classification. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Victoria, 2006
- [FF06b] FULOP, Sean A. ; FITZ, Kelly: Algorithms for computing the time-corrected instantaneous frequency (reassigned) spectrogram, with applications. In: *Journal of the Acoustical Society of America (JASA)* 119 (2006), Nr. 1, 360–371. <http://dx.doi.org/10.1121/1.2133000>. – DOI 10.1121/1.2133000. – ISSN 00014966

- [FG16] FLEXER, Arthur ; GRILL, Thomas: The Problem of Limited Inter-rater Agreement in Modelling Music Similarity. In: *Journal of New Music Research* 45 (2016), Juli, Nr. 3, 239–251. <http://dx.doi.org/10.1080/09298215.2016.1200631>. – DOI 10.1080/09298215.2016.1200631. – ISSN 0929-8215, 1744-5027
- [FH51] FIX, Evelyn ; HODGES, J L.: Discriminatory Analysis - Nonparametric Discrimination: Consistency Properties / University of California Berkeley. Version: Februar 1951. <https://apps.dtic.mil/docs/citations/ADA800276>. 1951. – Forschungsbericht
- [Fle24] FLETCHER, Harvey: The Physical Criterion for Determining the Pitch of a Musical Tone. In: *Physical Review* 23 (1924), März, Nr. 3, 427–437. <http://dx.doi.org/10.1103/PhysRev.23.427>. – DOI 10.1103/PhysRev.23.427. – ISSN 0031-899X
- [Fle40] FLETCHER, Harvey: Auditory Patterns. In: *Reviews of Modern Physics* 12 (1940), Nr. 1, S. 47–65
- [Fle75] FLETCHER, Neville H.: Acoustical correlates of flute performance technique. In: *Journal of the Acoustical Society of America (JASA)* 57 (1975), Nr. 1, 233–237. <http://dx.doi.org/10.1121/1.380430>. – DOI 10.1121/1.380430. – ISSN 00014966
- [Fle07] FLEXER, Arthur: A Closer Look on Artist Filters for Musical Genre Classification. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Vienna, 2007
- [FLTZ11] FU, Zhouyu ; LU, Guojun ; TING, Kai M. ; ZHANG, Dengsheng: A Survey of Audio-Based Music Classification and Annotation. In: *Transactions on Multimedia* 13 (2011), April, Nr. 2, 303–319. <http://dx.doi.org/10.1109/TMM.2010.2098858>. – DOI 10.1109/TMM.2010.2098858. – ISSN 1520-9210
- [FM33] FLETCHER, Harvey ; MUNSON, W A.: Loudness, Its Definition, Measurement and Calculation. In: *Journal of the Acoustical Society of America (JASA)* 5 (1933), Nr. 2, 82. <http://dx.doi.org/10.1121/1.1915637>. – DOI 10.1121/1.1915637. – ISSN 00014966
- [Foo97] FOOTE, Jonathan T.: A Similarity Measure for Automatic Audio Classification. In: *Proceedings of the 14th National Conference on Artificial Intelligence*. Providence, 1997
- [Foo00] FOOTE, Jonathan T.: Automatic Audio Segmentation Using a Measure of Audio Novelty. In: *Proceedings of the International Conference on Multimedia and Expo (ICME)*. New York, 2000, S. 452–455
- [FR98] FLETCHER, Neville H. ; ROSSING, Thomas D.: *The Physics of Musical Instruments*. 2nd Editio. Springer, 1998. – ISBN 978-0-387-98374-5. – 01508
- [Fra78] FRAISSE, Paul: Time and Rhythm Perception. Version: Januar 1978. <http://dx.doi.org/10.1016/B978-0-12-161908-4.50012-7>. In: CARTERETTE, Edward C. (Hrsg.) ; FRIEDMAN, Morton P. (Hrsg.): *Perceptual Coding*. Academic Press, Januar 1978. – DOI 10.1016/B978-0-12-161908-4.50012-7. – ISBN 978-0-12-161908-4, 203–254
- [FS92] FRIBERG, Anders ; SUNDBERG, Johan: Perception of just noticeable time displacement of a tone presented in a Metrical Sequence at Different Tempos. In: *STL-QPSR* 33 (1992), Nr. 4, S. 97–108
- [FS10] FLEXER, Arthur ; SCHNITZER, Dominik: Effects of Album and Artist Filters in Audio Similarity Computed for Very Large Music Databases. In: *Computer Music Journal* 34 (2010), Nr. 3, 20–28. http://www.mitpressjournals.org/doi/pdf/10.1162/COMJ_a_00004
- [FU01] FOOTE, Jonathan T. ; UCHIHASHI, Shingo: The beat spectrum: a new approach to rhythm analysis. In: *Proceedings of the International Conference on Multimedia and Expo (ICME)*. Tokyo, August 2001, S. 881–884

- [FU13] FARBOOD, Morwaread M. ; UPHAM, Finn: Interpreting Expressive Performance through Listener Judgments of Musical Tension. In: *Frontiers in Psychology* 4 (2013). <http://dx.doi.org/10.3389/fpsyg.2013.00998>. – DOI 10.3389/fpsyg.2013.00998. – ISSN 1664–1078
- [Fuj99] FUJISHIMA, Takuya: Realtime Chord Recognition of Musical Sound: a System Using Common Lisp Music. In: *Proceedings of the International Computer Music Conference (ICMC)*, 1999
- [FZP03] FENG, Yazhong ; ZHUANG, Yueling ; PAN, Yunhe: Music Information Retrieval by Detecting Mood via Computational Media Aesthetics. In: *Proceedings of the International Conference on Web Intelligence (WI)*, IEEE, 2003. – ISBN 0–7695–1932–6, 235–241
- [Gab99] GABRIELSSON, Alf: The Performance of Music. In: DEUTSCH, Diana (Hrsg.): *The Psychology of Music*. 2nd Editio. San Diego : Academic Press, 1999
- [GALL19] GROLLMISCH, Sascha ; ABESSER, Jakob ; LIEBETRAU, Judith ; LUKASHEVICH, Hanna: Sounding Industry: Challenges and Datasets for Industrial Sound Analysis. In: *Proceedings of the European Signal Processing Conference (EUSIPCO)*. A Coruna, Spain, September 2019, S. 1–5. – ISSN: 2076-1465
- [Gar95] GARDNER, William G.: Efficient Convolution without Input-Output Delay. In: *Journal of the Audio Engineering Society (JAES)* 43 (1995), Nr. 3, S. 127–136
- [GB03] GOEBL, Werner ; BRESIN, Roberto: Measurement and reproduction accuracy of computer-controlled grand pianos. In: *Journal of the Acoustical Society of America (JASA)* 114 (2003), Nr. 4, S. 2273–2283. <http://dx.doi.org/10.1121/1.1605387>. – DOI 10.1121/1.1605387
- [GB13] GÓMEZ, Emilia ; BONADA, Jordi: Towards Computer-Assisted Flamenco Transcription: An Experimental Comparison of Automatic Transcription Algorithms as Applied to A Cappella Singing. In: *Computer Music Journal* 37 (2013), Juni, Nr. 2, 73–90. https://doi.org/10.1162/COMJ_a_00180. – ISSN 0148–9267
- [GBC16] GOODFELLOW, Ian ; BENGIO, Yoshua ; COURVILLE, Aaron: *Deep learning*. MIT press, 2016
- [GCKT20] GROLLMISCH, Sascha ; CANO, Estefanía ; KEHLING, Christian ; TAENZER, Michael: Analyzing the Potential of Pre-Trained Embeddings for Audio Classification Tasks. In: *Proceedings of the European Signal Processing Conference (EUSIPCO)*. Online, 2020, S. 790–794. – ISSN: 2076-1465
- [GD97] GRUBB, Lorin ; DANNENBERG, Roger B.: A Stochastic Method of Tracking a Vocal Performer. In: *Proceedings of the International Computer Music Conference (ICMC)*. Thessaloniki, September 1997
- [GD98] GRUBB, Lorin ; DANNENBERG, Roger B.: Enhanced Vocal Performance Tracking Using Multiple Information Sources. In: *Proceedings of the International Computer Music Conference (ICMC)*. Ann Arbor, 1998
- [GD01] GOEBL, Werner ; DIXON, Simon: Analysis of Tempo Classes in Performances of Mozart Sonatas. In: *Proceedings of the 7th International Symposium on Systematic and Comparative Musicology (ISSCM), 3rd International Conference on Cognitive Musicology (ICCM)*. Jyväskylä, 2001
- [GE03] GUYON, Isabelle ; ELISSEIFI, Andre: An Introduction to Variable and Feature Selection. In: *Journal of Machine Learning Research* 3 (2003), Oktober, Nr. 7-8, 1157–1182. <http://dx.doi.org/10.1162/153244303322753616>. – DOI 10.1162/153244303322753616. – ISSN 1532–4435
- [GE11] GRAIS, Emad M. ; ERDOGAN, Hakan: Single Channel Speech Music Separation Using Nonnegative Matrix Factorization and Spectral Masks. In: *Proceedings of the International Conference on Digital Signal Processing (DSP)*, 2011, S. 1–6. – ISSN: 2165-3577

- [GH03] GOUYON, Fabien ; HERRERA, Perfecto: Determination of the Meter of Musical Audio Signals: Seeking Recurrences in Beat Segment Descriptors. In: *Proceedings of the 114th Audio Engineering Society Convention*. Amsterdam : AES, 2003
- [Gib98] GIBBS, J W.: Fourier's Series. In: *Nature* 59 (1898), Dezember, Nr. 1522, 200–200. <http://dx.doi.org/10.1038/059200b0>. – DOI 10.1038/059200b0. – ISSN 0028–0836
- [Gib99] GIBBS, J W.: Fourier's Series. In: *Nature* 59 (1899), April, Nr. 1539, 606–606. <http://dx.doi.org/10.1038/059606a0>. – DOI 10.1038/059606a0. – ISSN 0028–0836
- [GKD⁺06] GOUYON, Fabien ; Klapuri, Anssi ; DIXON, Simon ; ALONSO, Miguel ; TZANETAKIS, George ; UHLE, Christian ; CANO, Pedro: An Experimental Comparison of Audio Tempo Induction Algorithms. In: *IEEE Transactions on Audio, Speech, and Language Processing* 14 (2006), September, Nr. 5, S. 1832–1844. <http://dx.doi.org/10.1109/TSA.2005.858509>. – DOI 10.1109/TSA.2005.858509. – ISSN 1558–7924
- [GL03] GUO, Guodong ; LI, Stan Z.: Content-based audio classification and retrieval by support vector machines. In: *Transactions on Neural Networks* 14 (2003), Januar, Nr. 1, 209–215. <http://dx.doi.org/10.1109/TNN.2002.806626>. – DOI 10.1109/TNN.2002.806626. – ISSN 1045–9227
- [GL17] GURURANI, Siddharth ; LERCH, Alexander: Automatic Sample Detection in Polyphonic Music. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Suzhou : International Society for Music Information Retrieval (ISMIR), 2017
- [Gla17] GLASMACHERS, Tobias: Limits of End-to-End Learning. In: *Asian Conference on Machine Learning*, PMLR, November 2017, 17–32. – ISSN: 2640-3498
- [GM95] GOTO, Masataka ; MURAOKA, Yoichi: Music Understanding At The Beat Level – Real-time Beat Tracking For Audio Signals. In: *Proceedings of the Workshop on Computational Auditory Scene Analysis (IJCAI)*, 1995
- [GM96] GOTO, Masataka ; MURAOKA, Yoichi: Beat Tracking based on Multiple-agent Architecture – A Real-time Beat Tracking System for Audio Signals. In: *Proceedings of the Second International Conference on Multiagent Systems (ICMAS)*, 1996
- [GM99] GOTO, Masataka ; MURAOKA, Yoichi: Real-time Beat Tracking for Drumless Audio Signals: Chord Change Detection for Musical Decisions. In: *Speech Communication* 27 (1999), S. 311–335
- [GMS08] GATZSCHE, Gabriel ; MEHNERT, Markus ; STÖCKLMEIER, Christian: Interaction with tonal pitch spaces. In: *Proceedings of the 8th International Conference on New Interfaces for Musical Expression (NIME)*. Genova, 2008
- [Gor84] GORDON, John W.: *Perception of Attack Transients in Musical Tones*. Stanford, Stanford University, Center for Computer Research in Music and Acoustics (CCRMA), Dissertation, 1984
- [Got00] GOTO, Masataka: A Robust Predominant-F0 Estimation Method for Real-Time Detection of Melody and Bass Lines in CD Recordings. In: *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* Bd. 2, 2000, S. II757–II760 vol.2. – ISSN: 1520-6149
- [Got02] GOTO, Masataka: RWC Music Database: Popular, Classical, and Jazz Music Databases. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Paris, 2002, S. 2

- [Got06] GOTO, Masataka: A Chorus Section Detection Method for Musical Audio Signals and its Application to a Music Listening Station. In: *IEEE Transactions on Audio, Speech, and Language Processing* 14 (2006), September, Nr. 5, S. 1783–1794. <http://dx.doi.org/10.1109/TSA.2005.863204>. – DOI 10.1109/TSA.2005.863204. – ISSN 1558–7924
- [GP08] GJERDINGEN, Robert O. ; PERROTT, David: Scanning the Dial: The Rapid Recognition of Music Genres. In: *Journal of New Music Research* 37 (2008), Juni, Nr. 2, 93–100. <http://dx.doi.org/10.1080/09298210802479268>. – DOI 10.1080/09298210802479268. – ISSN 0929–8215
- [GPG⁺16] GINGRAS, Bruno ; PEARCE, Marcus T. ; GOODCHILD, Meghan ; DEAN, Roger T. ; WIGGINS, Geraint ; MCADAMS, Stephen: Linking Melodic Expectation to Expressive Performance Timing and Perceived Musical Tension. In: *Journal of Experimental Psychology: Human Perception and Performance* 42 (2016), Nr. 4, S. 594–609. <http://dx.doi.org/10.1037/xhp0000141>. – DOI 10.1037/xhp0000141. – ISSN 1939–1277(Electronic),0096–1523(Print)
- [GR06] GILLET, Olivier ; RICHARD, Gaël: ENST-Drums: an Extensive Audio-Visual Database for Drum Signals Processing. In: *ISMIR*. Victoria, 2006
- [Gre61] GREENWOOD, Donald D.: Critical Bandwidth and the Frequency Coordinates of the Basilar Membrane. In: *Journal of the Acoustical Society of America (JASA)* 33 (1961), Nr. 10, 1344–1356. <http://dx.doi.org/10.1121/1.1908437>. – DOI 10.1121/1.1908437. – ISSN 00014966
- [GRKN07] GROSSE, Roger ; RAINA, Rajat ; KWONG, Helen ; NG, Andrew Y.: Shift-Invariant Sparse Coding for Audio Classification. In: *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*. Arlington, Virginia, USA : AUAI Press, Juli 2007 (UAI'07). – ISBN 978–0–9749039–3–4, S. 149–158
- [GSL19] GURURANI, Siddharth ; SHARMA, Mohit ; LERCH, Alexander: An Attention Mechanism for Music Instrument Recognition. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Delft : International Society for Music Information Retrieval ({ISMIR}), 2019
- [GW19] GADERMAIER, Thassilo ; WIDMER, Gerhard: A Study of Annotation and Alignment Accuracy for Performance Comparison in Complex Orchestral Music. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Delft, Netherlands, Oktober 2019. – arXiv: 1910.07394
- [HABS00] HERRERA, Perfecto ; AMATRIAIN, Xavier ; BATLLE, Eloi ; SERRA, Xavier: Towards Instrument Segmentation for Music Content Description: A Critical Review of Instrument Classification Techniques. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2000
- [HAH⁺01] HELLMUTH, Oliver ; ALLAMANCHE, Eric ; HERRE, Jürgen ; KASTNER, Thorsten ; CREMER, Markus ; HIRSCH, Wolfgang: Advanced Audio Identification Using MPEG-7 Content Description. In: *Proceedings of the 111th Audio Engineering Society Convention (Preprint No. 5463)*. New York, 2001
- [Har78] HARRIS, Fredric J.: On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform. In: *Proceedings of the IEEE* 66 (1978), Nr. 1, S. 51–83
- [Haw04] HAWKINS, Douglas M.: The Problem of Overfitting. In: *Journal of Chemical Information and Computer Sciences* 44 (2004), Januar, Nr. 1, 1–12. <http://dx.doi.org/10.1021/ci0342472>. – DOI 10.1021/ci0342472. – ISSN 0095–2338

- [HB11] HONINGH, Aline ; BOD, Rens: Clustering and Classification of Music by Interval Categories. In: *Proceedings of the International Conference on Mathematics and Computation in Music*. Berlin, Heidelberg : Springer, 2011 (Lecture Notes in Computer Science). – ISBN 978-3-642-21590-2, S. 346–349
- [HB12] HUMPHREY, Eric J. ; BELLO, Juan P.: Rethinking Automatic Chord Recognition with Convolutional Neural Networks. In: *Proceedings of the International Conference on Machine Learning and Applications (ICMLA)* Bd. 2, 2012, S. 357–362
- [HBL13] HUMPHREY, Eric J. ; BELLO, Juan P. ; LECUN, Yann: Feature Learning and Deep Architectures: new Directions for Music Informatics. In: *Journal of Intelligent Information Systems* 41 (2013), Dezember, Nr. 3, 461–481. <http://dx.doi.org/10.1007/s10844-013-0248-5>. – DOI 10.1007/s10844-013-0248-5. – ISSN 0925-9902, 1573–7675
- [HBR10] HUQ, Arefin ; BELLO, Juan P. ; ROWE, Robert: Automated Music Emotion Recognition: A Systematic Evaluation. In: *Journal of New Music Research* 39 (2010), September, Nr. 3, 227–244. <http://dx.doi.org/10.1080/09298215.2010.513733>. – DOI 10.1080/09298215.2010.513733. – ISSN 0929–8215
- [HBR14] HUANG, Xuedong ; BAKER, James ; REDDY, Raj: A Historical Perspective of Speech Recognition. In: *Communications of the ACM* 57 (2014), Januar, Nr. 1, 94–103. <http://dx.doi.org/10.1145/2500887>. – DOI 10.1145/2500887. – ISSN 0001–0782
- [HCE⁺17] HERSEY, Shawn ; CHAUDHURI, Sourish ; ELLIS, Dan P. ; GEMMEKE, Jort F. ; JANSEN, Aren ; MOORE, R C. ; PLAKAL, Manoj ; PLATT, Devin ; SAUROUS, Riff A. ; SEYBOLD, Bryan ; SLANEY, Malcolm ; WEISS, Ron J. ; WILSON, Kevin: CNN Architectures for Large-Scale Audio Classification. In: *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. New Orleans, LA : Institute of Electrical and Electronics Engineers (IEEE), März 2017, S. 131–135. – ISSN: 2379-190X
- [HD07] HU, Xiao ; DOWNIE, J S.: Exploring Mood Metadata: Relationships with Genre, Artist and Usage Metadata. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Vienna, 2007
- [HDT03] HU, Ning ; DANNENBERG, Roger B. ; TZANETAKIS, George: Polyphonic Audio Matching and Alignment for Music Retrieval. In: *Proceedings of the Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. New Paltz : IEEE, 2003
- [HDZ⁺12] HOLZAPFEL, André ; DAVIES, Matthew E P. ; ZAPATA, José R. ; OLIVEIRA, João L. ; GOUYON, Fabien: Selective Sampling for Beat Tracking Evaluation. In: *IEEE Transactions on Audio, Speech, and Language Processing* 20 (2012), November, Nr. 9, S. 2539–2548. <http://dx.doi.org/10.1109/TASL.2012.2205244>. – DOI 10.1109/TASL.2012.2205244. – ISSN 1558–7924. – Conference Name: IEEE Transactions on Audio, Speech, and Language Processing
- [HE02] HOFMANN-ENGL, Ludger: Rhythmic Similarity: A Theoretical and Empirical Approach. In: *Proceedings of the 7th International Conference on Music Perception and Cognition (ICMPC)*. Sydney, 2002, S. 564–567
- [HE10] HAMEL, Philippe ; ECK, Douglas: Learning Features from Music Audio with Deep Belief Networks. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Utrecht, Netherlands, 2010, S. 6
- [Hei96] HEIJINK, Hank: *Matching Scores and Performances*, Nijmegen University, Diplomarbeit, 1996
- [Hel70] HELMHOLTZ, Hermann V.: *Die Lehre von den Tonempfindungen als physiologische Grundlage für die Theorie der Musik*. 3rd Editio. Braunschweig : Vieweg, 1870

- [HES⁺18] HAWTHORNE, Curtis ; ELSEN, Erich ; SONG, Jialin ; ROBERTS, Adam ; SIMON, Ian ; RAFFEL, Colin ; ENGEL, Jesse ; OORE, Sageev ; ECK, Douglas: Onsets and Frames: Dual-Objective Piano Transcription. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Paris, Juni 2018. – arXiv: 1710.11153
- [HFH⁺09] HALL, Mark ; FRANK, Eibe ; HOLMES, Geoffrey ; PFAHRINGER, Bernhard ; REUTEMANN, Peter ; WITTEN, Ian H.: The WEKA Data Mining Software: An Update. In: *ACM SIGKDD Explorations Newsletter* 11 (2009), November, Nr. 1, 10–18. <http://dx.doi.org/10.1145/1656274.1656278>. – DOI 10.1145/1656274.1656278. – ISSN 19310145
- [HHP⁺20] HUANG, Jiawen ; HUNG, Yun-Ning ; PATI, K A. ; GURURANI, Siddharth ; LERCH, Alexander: Score-informed Networks for Music Performance Assessment. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Montreal : International Society for Music Information Retrieval (ISMIR), 2020
- [Hil02] HILL, Peter: From Score to Sound. In: RINK, John (Hrsg.): *Musical Performance – A Guide to Understanding*. Cambridge : Cambridge University Press, 2002. – ISBN 978-0-521-78862-5
- [Hir59] HIRSH, Ira J.: Auditory Perception of Temporal Order. In: *Journal of the Acoustical Society of America (JASA)* 31 (1959), Nr. 6, 759. <http://dx.doi.org/10.1121/1.1907782>. – DOI 10.1121/1.1907782. – ISSN 00014966
- [HJKL11] HENAFF, Mikael ; JARRETT, Kevin ; KAVUKCUOGLU, Koray ; LECUN, Yann: Unsupervised Learning of Sparse Features for Scalable Audio Classification. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Miami, FL, 2011, S. 6
- [HK02] HAITSMA, Jaap ; KALKER, Ton: A Highly Robust Audio Fingerprinting System. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Paris, 2002
- [HKV09] HEITTO LA, Toni ; Klapuri, Anssi P. ; Virtanen, Tuomas: Musical Instrument Recognition in Polyphonic Audio Using Source-Filter Model for Sound Separation. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Kobe, 2009
- [HM03] HAINSWORTH, Stephen W. ; MACLEOD, Malcolm: Onset Detection in Musical Audio Signals. In: *Proceedings of the International Computer Music Conference (ICMC)*. Singapore, 2003
- [HRJH10] HAN, Byeong-jun ; RHO, Seungmin ; JUN, Sanghoon ; HWANG, Eenjun: Music Emotion Classification and Context-Based Music Recommendation. In: *Multimedia Tools and Applications* 47 (2010), August, Nr. 3, 433–460. <http://dx.doi.org/10.1007/s11042-009-0332-6>. – DOI 10.1007/s11042-009-0332-6. – ISSN 1380-7501
- [HSG06] HARTE, Christopher A. ; SANDLER, Mark B. ; GASSER, Martin: Detecting harmonic change in musical audio. In: *Proceedings of the 1st ACM Workshop on Audio and Music Computing Multimedia (AMCMM)*. Santa Barbara : ACM, 2006. – 00088
- [HSL13] HEO, Hoon ; SUNG, Dooyong ; LEE, Kyogu: Note Onset Detection based on Harmonic Cepstrum Regularity. In: *Proceedings of the International Conference on Multimedia and Expo (ICME)*. San Jose, CA : Institute of Electrical and Electronics Engineers (IEEE), Juli 2013, S. 1–6. – ISSN: 1945-788X
- [HSR⁺19] HAWTHORNE, Curtis ; STASYUK, Andriy ; ROBERTS, Adam ; SIMON, Ian ; HUANG, Cheng-Zhi A. ; DILEMAN, Sander ; ELSEN, Erich ; ENGEL, Jesse ; ECK, Douglas: Enabling Factorized Piano Music Modeling and Generation with the MAESTRO Dataset. In: *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019

- [HSS⁺21] HAWTHORNE, Curtis ; SIMON, Ian ; SWAVELY, Rigel ; MANILOW, Ethan ; ENGEL, Jesse: Sequence-to-Sequence Piano Transcription with Transformers. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Online, Juli 2021. – arXiv: 2107.09142
- [HTS02] HUSAIN, Gabriela ; THOMPSON, William F. ; SCHELLENBERG, E G.: Effects of Musical Tempo and Mode on Arousal, Mood, and Spatial Abilities. In: *Music Perception* 20 (2002), Nr. 2, S. 151–171
- [IBWW97] IYER, Vijay ; BILMES, Jeff ; WRIGHT, Matt ; WESSEL, David L.: A Novel Representation for Rhythmic Structure. In: *Proceedings of the International Computer Music Conference (ICMC)*. Thessaloniki : ICMA, 1997, S. 1–5
- [IEC02] IEC 61672:2002: Electroacoustics – Sound Level Meters – Part1: Specifications / IEC. 2002 (61672:2002). – Standard. – 00000
- [IK93] IVERSON, Paul ; KRUMHANSL, Carol L.: Isolating the dynamic attributes of musical timbre. In: *Journal of the Acoustical Society of America (JASA)* 94 (1993), Nr. 5, S. 2595–2603
- [ISO03] ISO: ISO 226:2003. Version: 2003. <https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/03/42/34222.html>. 2003 (ISO 226:2003). – Forschungsbericht
- [ISO17] ISO 532-1:2017: ISO 532-1:2017. Version: 2017. <https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/06/30/63077.html>. 2017. – Forschungsbericht
- [Ita75] ITAKURA, Fumitada: Minimum Prediction Residual Principle applied to Speech Recognition. In: *Transactions on Acoustics, Speech, and Signal Processing* 23 (1975), Februar, Nr. 1, 67–72. <http://dx.doi.org/10.1109/TASSP.1975.1162641>. – DOI 10.1109/TASSP.1975.1162641. – ISSN 0096–3518
- [Izm05] IZMIRLI, Özgür: Template based key finding from audio. In: *Proceedings of the International Computer Music Conference (ICMC)*. Barcelona, September 2005
- [JER09] JODER, Cyril ; ESSID, Slim ; RICHARD, Gaël: Temporal Integration for Audio Classification With Application to Musical Instrument Classification. In: *Transactions on Audio, Speech, and Language Processing* 17 (2009), Januar, Nr. 1, 174–186. <http://dx.doi.org/10.1109/TASL.2008.2007613>. – DOI 10.1109/TASL.2008.2007613. – ISSN 1558–7916
- [JER13] JODER, Cyril ; ESSID, Slim ; RICHARD, Gaël: Learning Optimal Features for Polyphonic Audio-to-Score Alignment. In: *IEEE Transactions on Audio, Speech, and Language Processing* 21 (2013), Oktober, Nr. 10, S. 2118–2128. <http://dx.doi.org/10.1109/TASL.2013.2266794>. – DOI 10.1109/TASL.2013.2266794. – ISSN 1558–7924. – Conference Name: IEEE Transactions on Audio, Speech, and Language Processing
- [JGKM11] JIANG, Nanzhu ; GROSCHÉ, Peter ; KONZ, Verena ; MULLER, Meinard: Analyzing Chroma Feature Types for Automated Chord Recognition. (2011), S. 11
- [JN84] JAYANT, Nuggehally S. ; NOLL, Peter: *Digital Coding of Waveforms – Principles and Applications to Speech and Video*. New Jersey : Prentice Hall, 1984
- [JNC19] JARAMILLO, Alfredo E. ; NIELSEN, Jesper K. ; CHRISTENSEN, Mads G.: A Study on How Pre-whitening Influences Fundamental Frequency Estimation. In: *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, S. 6495–6499. – ISSN: 2379-190X
- [Joh02] JOHNSON, Peter: The legacy of recordings. In: RINK, John (Hrsg.): *Musical Performance – A Guide to Understanding*. Cambridge : Cambridge University Press, 2002

- [Jus00] JUSLIN, Patrik N.: Cue Utilization in Communication of Emotion in Music Performance: Relating Performance to Perception. In: *Journal of Experimental Psychology* 26 (2000), Nr. 6, S. 1797–1813
- [Jus03a] JUSLIN, Patrik N.: Five myths about expressivity in music performance and what to do about them. In: *Proceedings of the International Conference on Arts and Humanities*. Honolulu, Hawaii, 2003
- [Jus03b] JUSLIN, Patrik N.: Studies of Music Performance: A Theoretical Analysis of Empirical Findings. In: *Proceedings of the Stockholm Music Acoustics Conference (SMAC)*. Stockholm, August 2003
- [KAHH02] KASTNER, Thorsten ; ALLAMANCHE, Eric ; HERRE, Jürgen ; HELLMUTH, Oliver: MPEG-7 Scalable Robust Audio Fingerprinting. In: *Proceedings of the 112th Audio Engineering Society Convention* (2002)
- [Kar90] KARTOMI, Margaret J.: *On Concepts and Classifications of Musical Instruments*. University Of Chicago Press, 1990. – ISBN 978-0-226-42549-8
- [Kat04] KATZ, Mark: *Capturing Sound: How Technology has Changed Music*. Berkeley and Los Angeles : University of California Press, 2004. – ISBN 0-520-24380-3
- [KC90] KENDALL, Roger A. ; CARTERETTE, Edward C.: The Communication of Musical Expression. In: *Music Perception* 8 (1990), Nr. 2, S. 129–164
- [KC01] KOSTEK, Bozena ; CZYZEWSKI, Andrzej: Representing Musical Instrument Sounds for their Automatic Classification. In: *Journal of the Audio Engineering Society (JAES)* 49 (2001), Nr. 9, S. 768–785
- [KC10] KULESZA, Maciej ; CZYZEWSKI, Andrzej: Frequency based criterion for distinguishing tonal and noisy spectral components. In: *International Journal of Computer Science and Security* 4 (2010), März, Nr. 1, 1. <http://www.cscjournals.org/csc/manuscriptinfo.php?ManuscriptCode=77.74.67.68.39.47.48.104>
- [KD21] KOH, Eunjeong ; DUBNOV, Shlomo: Comparison and Analysis of Deep Audio Embeddings for Music Emotion Recognition. In: *AAAI Workshop on Affective Content Analysis*, 2021. – arXiv: 2104.06517
- [KDK⁺16] KELZ, Rainer ; DORFER, Matthias ; KORZENIOWSKI, Filip ; BÖCK, Sebastian ; ARZT, Andreas ; WIDMER, Gerhard: On the Potential of Simple Framewise Approaches to Piano Transcription. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. New York, Dezember 2016. – arXiv: 1612.05153
- [KFH⁺15] KNEES, Peter ; FARALDO, Angel ; HERRERA, Perfecto ; VOGL, Richard ; BÖCK, Sebastian ; HORSCHLAGER, Florian ; LE GOFF, Mickael: Two Data Sets for Tempo Estimation and Key Detection in Electronic Dance Music Annotated from User Corrections. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2015, 364–370
- [KHB⁺19] KOOPS, Hendrik V. ; HAAS, W. B. ; BURGOYNE, John A. ; BRANSEN, Jeroen ; KENT-MULLER, Anna ; VOLK, Anja: Annotator Subjectivity in Harmony Annotations of Popular Music. In: *Journal of New Music Research* 48 (2019), Mai, Nr. 3, 232–252. <http://dx.doi.org/10.1080/09298215.2019.1613436>. – DOI 10.1080/09298215.2019.1613436. – ISSN 0929-8215. – Publisher: Routledge _eprint: <https://doi.org/10.1080/09298215.2019.1613436>
- [Kin16] KINNAIRD, Katherine M.: Aligned Hierarchies: A Multi-Scale Structure-Based Representation for Music-Based Data Streams. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. New York, 2016, S. 7

- [KK81] KLEINGINNA, Paul R. ; KLEINGINNA, Anne M.: A Categorized List of Emotion Definitions, with Suggestions for a Consensual Definition. In: *Motivation and Emotion* 5 (1981), Dezember, Nr. 4, 345–379. <http://dx.doi.org/10.1007/BF00992553>. – DOI 10.1007/BF00992553. – ISSN 0146–7239
- [KKM01] KIMURA, Akisato ; KASHINO, Kunio ; KUROZUMI, Takayuki ; MURASE, Hiroshi: Very Quick Audio Searching: Introducing Global Pruning to the Time-Series Active Search. In: *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* Bd. 3, IEEE, 2001. – ISBN 0–7803–7041–4, 1429–1432
- [KL11] KIRCHHOFF, Holger ; LERCH, Alexander: Evaluation of Features for Audio-to-Audio Alignment. In: *Journal of New Music Research* 40 (2011), Nr. 1, 27–41. <http://dx.doi.org/10.1080/09298215.2010.529917>. – DOI 10.1080/09298215.2010.529917
- [Kla99] KLAPURI, Anssi P.: Sound Onset Detection by Applying Psychoacoustic Knowledge. In: *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Phoenix, 1999
- [Kla01] KLAPURI, Anssi P.: Multipitch Estimation and Sound Separation by the Spectral Smoothness Principle. In: *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing. Proceedings (ICASSP)* (2001), 3381–3384. <http://dx.doi.org/10.1109/ICASSP.2001.940384>. – DOI 10.1109/ICASSP.2001.940384
- [Kla03a] KLAPURI, Anssi P.: Multiple Fundamental Frequency Estimation Based on Harmonicity and Spectral Smoothness. In: *Transactions on Speech and Audio Processing* 11 (2003), November, Nr. 6, 804–816. <http://dx.doi.org/10.1109/TSA.2003.815516>. – DOI 10.1109/TSA.2003.815516. – ISSN 1063–6676
- [Kla03b] KLAPURI, Anssi P.: Musical Meter Estimation and Music Transcription. In: *Proceedings of the Cambridge Music Processing Colloquium*. Cambridge, 2003
- [Kla05] KLAPURI, Anssi P.: A Perceptually Motivated Multiple-F0 Estimation Method. In: *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. New Paltz, 2005
- [Kla06] KLAPURI, Anssi P.: Auditory Model-Based Methods for Multiple Fundamental Frequency Estimation. In: KLAPURI, Anssi P. (Hrsg.) ; DAVY, Manuel (Hrsg.): *Signal Processing Methods for Music Transcription*. Berlin : Springer, 2006, S. 229–265
- [KLZ13] KRAFT, Sebastian ; LERCH, Alexander ; ZÖLZER, Udo: The Tonalness Spectrum: Feature-Based Estimation of Tonal Components. In: *Proceedings of the 16th International Conference on Digital Audio Effects*. Maynooth, 2013
- [KM95] KAMINSKYJ, Ian ; MATERKA, A: Automatic Source Identification of Monophonic Musical Instrument Sounds. In: *Proceedings of the International Conference on Neural Networks (ICNN)* Bd. 1, IEEE, 1995. – ISBN 0–7803–2768–3, 189–194
- [KM06] KANTOR-MARTYNUSKA, Joanna: Emotion-relevant characteristics of temperament and the perceived magnitude of tempo and loudness of music. In: *Proceedings of the 9th International Conference on Music Perception and Cognition (ICMPC)*. Bologna, August 2006
- [KM21] KINNAIRD, Katherine M. ; MCCEE, Brian: Automatic Hierarchy Expansion for Improved Structure and Chord Evaluation. In: *Transactions of the International Society for Music Information Retrieval* 4 (2021), Juni, Nr. 1, 81–92. <http://dx.doi.org/10.5334/tismir.71>. – DOI 10.5334/tismir.71. – ISSN 2514–3298. – Number: 1 Publisher: Ubiquity Press

- [KNS07] KAMEOKA, Hirokazu ; NISHIMOTO, Takuya ; SAGAYAMA, Shigeki: A Multipitch Analyzer Based on Harmonic Temporal Structured Clustering. In: *Transactions on Audio, Speech and Language Processing* 15 (2007), März, Nr. 3, 982–994. <http://dx.doi.org/10.1109/TASL.2006.885248>. – DOI 10.1109/TASL.2006.885248. – ISSN 1558–7916
- [Kot03] KOTELNIKOV, Vladimir A.: On the Transmission Capacity of 'Ether' and Wire in Electric Communications. In: *Reprint by the Institute of Radioengineering and Electronics of the Moscow Power Engineering Institute (Technical University) of the original article from 1933* (2003). <http://dx.doi.org/10.1070/PU2006v049n07ABEH006048>. – DOI 10.1070/PU2006v049n07ABEH006048
- [KR05] KAUFMAN, Leonard ; ROUSSEEUW, Peter J.: *Finding Groups in Data: an Introduction to Cluster Analysis*. Hoboken, N.J : Wiley, 2005 (Wiley series in probability and mathematical statistics). – ISBN 978–0–471–73578–6
- [KR16] KUMAR, Anurag ; RAJ, Bhiksha: Audio Event Detection using Weakly Labeled Data. In: *Proceedings of the International Conference on Multimedia (MM)*. New York, NY, USA : Association for Computing Machinery, Oktober 2016 (MM '16). – ISBN 978–1–4503–3603–1, 1038–1047
- [Kri16] KRIG, Scott: *Computer Vision Metrics*. Cham : Springer International Publishing, 2016. <http://dx.doi.org/10.1007/978-3-319-33762-3>. – ISBN 978–3–319–33761–6 978–3–319–33762–3
- [Kru90] KRUMHANSL, Carol L.: *Cognitive Foundations of Musical Pitch*. New York : Oxford University Press, 1990
- [Kru96] KRUMHANSL, Carol L.: A Perceptual Analysis of Mozart's Piano Sonata K. 282: Segmentation, Tension, and Musical Ideas. In: *Music Perception: An Interdisciplinary Journal* 13 (1996), April, Nr. 3, 401–432. <http://dx.doi.org/10.2307/40286177>. – DOI 10.2307/40286177. – ISSN 0730–7829, 1533–8312
- [KS03] KRISHNA, A G. ; SREENIVAS, T V.: Music Instrument Recognition: from Isolated Notes to Solo Phrases. In: *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* Bd. 4, IEEE, 2003. – ISBN 0–7803–8484–9, iv–265–iv–268
- [KS10] KAISER, Florian ; SIKORA, Thomas: Music Structure Discovery in Popular Music using Non-negative Matrix Factorization. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Utrecht, Netherlands, 2010, 429–434
- [KSG19] KNEES, Peter ; SCHEIDL, Markus ; GOTO, Masataka: Intelligent User Interfaces for Music Discovery: The Past 20 Years and What's to Come. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Delft, Netherlands, 2019, 44–53
- [KSM⁺10] KIM, Youngmoo E. ; SCHMIDT, Erik M. ; MIGNECO, Raymond ; MORTON, Brandon G. ; RICHARDSON, Patrick ; SCOTT, Jeffrey ; SPECK, Jacqueline A. ; TURNBULL, Douglas: Music Emotion Recognition: a State of the Art Review. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Utrecht, 2010
- [KT99] KARJALAINEN, Matti ; TOLONEN, Tero: Multi-pitch and periodicity analysis model for sound separation and auditory scene analysis. In: *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Phoenix : IEEE, 1999. – ISBN 0–7803–5041–3, 929–932 vol.2
- [KUF11] KNIGHT, Trevor ; UPHAM, Finn ; FUJINAGA, Ichiro: The potential for automatic assessment of trumpet tone quality. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Miami, 2011, S. 573–578

- [KV96] KAMINSKYJ, Ian ; VOUMARD, P: Enhanced Automatic Source Identification of Monophonic Musical Instrument Sounds. In: *Proceedings of the Australian New Zealand Conference on Intelligent Information Systems (ANZIIS)*. Adelaide : IEEE, November 1996. – ISBN 0-7803-3667-4, 76–79
- [Kvh00] Klapuri, Anssi P. ; Virtanen, Tuomas ; Holm, Jan-Markus: Robust Multipitch Estimation for the Analysis and Manipulation of Polyphonic Musical Signals. In: *Proceedings of the Conference on Digital Audio Effects (DAFX)*. Verona, 2000
- [KW16a] Korzeniowski, Filip ; Widmer, Gerhard: Feature Learning for Chord Recognition: The Deep Chroma Extractor. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. New York, Dezember 2016. – arXiv: 1612.05065
- [KW16b] Korzeniowski, Filip ; Widmer, Gerhard: A Fully Convolutional Deep Auditory Model for Musical Chord Recognition. In: *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, 2016, S. 1–6
- [KW17] Korzeniowski, Filip ; Widmer, Gerhard: End-to-end Musical Key Estimation using a Convolutional Neural Network. In: *Proceedings of the European Signal Processing Conference (EUSIPCO)*, 2017, S. 966–970. – ISSN: 2076-1465
- [KY07] Kim, Sungwoong ; Yoo, Chang D.: Boosted Binary Audio Fingerprint Based on Spectral Subband Moments. In: *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)* Bd. 1, Ieee, April 2007. – ISBN 1-4244-0727-3, I-241-I-244
- [KYKC11] Kim, Minje ; Yoo, Jiho ; Kang, Kyeongok ; Choi, Seungjin: Nonnegative Matrix Partial Co-Factorization for Spectral and Temporal Drum Source Separation. In: *IEEE Journal of Selected Topics in Signal Processing* 5 (2011), Oktober, Nr. 6, S. 1192–1204. <http://dx.doi.org/10.1109/JSTSP.2011.2158803>. – DOI 10.1109/JSTSP.2011.2158803. – ISSN 1941-0484. – Conference Name: IEEE Journal of Selected Topics in Signal Processing
- [LADG09] Lukashevich, Hanna ; Abesser, Jakob ; Dittmar, Christian ; Grossmann, Holger: From Multi-Labeling to Multi-Domain-Labeling: A Novel Two-Dimensional Approach to Music Genre Classification. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Kobe, Japan, 2009, S. 6
- [Lak00] Lakatos, Stephen: A common perceptual space for harmonic and percussive timbres. In: *Perception & Psychophysics* 62 (2000), Nr. 7, S. 1426–1439
- [Lapg19] Lerch, Alexander ; Arthur, Claire ; Pati, Ashis ; Gururani, Siddharth: Music Performance Analysis: A Survey. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Delft : International Society for Music Information Retrieval ({ISMIR}), 2019
- [Lapg20] Lerch, Alexander ; Arthur, Claire ; Pati, K A. ; Gururani, Siddharth: An Interdisciplinary Review of Music Performance Analysis. In: *Transactions of the International Society for Music Information Retrieval (TISMIR)* 3 (2020), Nr. 1, 221–245. <http://dx.doi.org/10.5334/tismir.53>. – DOI 10.5334/tismir.53
- [Lar93] Large, Edward W.: Dynamic programming for the analysis of serial behaviors. In: *Behavior Research Methods, Instruments, and Computers* 25 (1993), Nr. 2, S. 238–241
- [Lar95] Large, Edward W.: Beat Tracking with a Nonlinear Oscillator. In: *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI)*. Montreal, August 1995
- [Lar03] Laroche, Jean: Efficient Tempo and Beat Tracking in Audio Recordings. In: *Journal of the Audio Engineering Society (JAES)* 51 (2003), Nr. 4, S. 226–233

- [LBS⁺20] LEE, Jongpil ; BRYAN, Nicholas J. ; SALAMON, Justin ; JIN, Zeyu ; NAM, Juhan: Disentangled Multidimensional Metric Learning for Music Similarity. In: *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, S. 6–10. – ISSN: 2379-190X
- [LC96] LEVITIN, Daniel J. ; COOK, Perry R.: Memory for musical tempo: Additional evidence that auditory memory is absolute. In: *Perception & Psychophysics* 58 (1996), Januar, Nr. 6, 927–935. <http://dx.doi.org/10.3758/BF03205494>. – DOI 10.3758/BF03205494. – ISSN 1532–5962
- [LD01] LAMONT, Alexandra ; DIBBEN, Nicola: Motivic Structure and the Perception of Similarity. In: *Music Perception* 18 (2001), März, Nr. 3, 245–274. <http://dx.doi.org/10.1525/mp.2001.18.3.245>. – DOI 10.1525/mp.2001.18.3.245. – ISSN 0730–7829
- [LDR04] LEVEAU, Pierre ; DAUDET, Laurent ; RICHARD, Gaël: Methodology and Tools for the Evaluation of Automatic Onset Detection Algorithms in Music. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Barcelona, 2004
- [LE07] LACOSTE, Alexandre ; ECK, Douglas: A Supervised Classification Algorithm for Note Onset Detection. In: *EURASIP Journal on Advances in Signal Processing* 2007 (2007), Nr. 1, 043745. <http://dx.doi.org/10.1155/2007/43745>. – DOI 10.1155/2007/43745. – ISSN 1687–6180
- [LEB03] LOGAN, Beth ; ELLIS, Daniel P W. ; BERENZWEIG, Adam: Toward Evaluation Techniques for Music Similarity. In: *Proceedings of the Workshop on the Evaluation of Music Information Retrieval System at SIGIR*. Toronto, 2003
- [Ler04] LERCH, Alexander: Ein Ansatz zur automatischen Erkennung der Tonart in Musikdateien. In: *Proceedings of the VDT International Audio Convention (23. Tonmeistertagung)*. Leipzig, November 2004
- [Ler06] LERCH, Alexander: On the Requirement of Automatic Tuning Frequency Estimation. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Victoria : ISMIR, 2006
- [Ler09] LERCH, Alexander: *Software-Based Extraction of Objective Parameters from Music Performances*. München : GRIN Verlag, 2009 [10.14279/depositonce-2025](https://doi.org/10.14279/depositonce-2025). – ISBN 978–3–640–29496–1
- [Ler14] LERCH, Alexander: Music Information Retrieval. In: WEINZIERL, Stefan (Hrsg.): *Akustische Grundlagen der Musik*. Laaber, 2014 (Handbuch der Systematischen Musikwissenschaft 5). – ISBN 978–3–89007–699–7, S. 79–102
- [LET05] LERCH, Alexander ; EISENBERG, Gunnar ; TANGHE, Koen: FEAPI: A Low Level Feature Extraction Plugin API. In: *Proceedings of 8th International Conference on Digital Audio Effects (DAFX)*. Madrid, September 2005
- [LG91] LUCAS, Peter J F. ; GAAG, Linda C. d.: *Principles of Expert Systems*. Addison-Wesley, 1991
- [LGWM03] LIU, Ruolun ; GRIFFITH, Niall John L. ; WALKER, Jacqueline ; MURPHY, Peter: Time Domain Note Average Energy Based Music Onset Detection. In: *Proceedings of the Stockholm Music Acoustics Conference (SMAC)*. Stockholm, August 2003
- [Li00] LI, Stan Z.: Content-based audio classification and retrieval using the nearest feature line method. In: *Transactions on Speech and Audio Processing* 8 (2000), Nr. 5, 619–625. <http://dx.doi.org/10.1109/89.861383>. – DOI 10.1109/89.861383. – ISSN 10636676
- [LJ83] LERDAHL, Fred ; JACKENDORF, Ray: *A Generative Theory of Tonal Music*. Cambridge : MIT Press, 1983. – ISBN 978–0–262–62107–6

- [LKD10] LYON, Richard F. ; KATSIAMIS, Andreas G. ; DRAKAKIS, Emmanuel M.: History and future of auditory filter models. In: *Proceedings of the International Symposium on Circuits and Systems (ISCAS)*, IEEE, Mai 2010. – ISBN 978-1-4244-5308-5, 3809–3812
- [LKS⁺98] LAMBOU, T. ; KUDUMAKIS, P. ; SPELLER, R. ; SANDLER, Mark B. ; LINNEY, A.: Classification of audio signals using statistical features on time and wavelet transform domains. In: *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)* 6 (1998), Nr. Mdc, 3621–3624. <http://dx.doi.org/10.1109/ICASSP.1998.679665>. – DOI 10.1109/I-CASSP.1998.679665
- [LL15] LYKARTSIS, Athanasios ; LERCH, Alexander: Beat Histogram Features for Rhythm-based Musical Genre Classification Using Multiple Novelty Functions. In: *Proceedings of the International Conference on Digital Audio Effects (DAFX)*. Trondheim, Norway, 2015
- [LLZ06] LU, Lie ; LIU, Dan ; ZHANG, Hong-Jiang: Automatic Mood Detection and Tracking of Music Audio Signals. In: *Transactions on Audio, Speech and Language Processing* 14 (2006), Januar, Nr. 1, 5–18. <http://dx.doi.org/10.1109/TSA.2005.860344>. – DOI 10.1109/TSA.2005.860344. – ISSN 1558–7916
- [LM07] LAGRANGE, Mathieu ; MARCHAND, Sylvain: Estimating the Instantaneous Frequency of Sinusoidal Components Using Phase-Based Methods. In: *Journal of the Audio Engineering Society (JAES)* 55 (2007), Nr. 5, S. 385–399
- [LMMT04] LIPPENS, Stefaan ; MARTENS, Jean-Pierre ; MULDER, Tom D. ; TZANETAKIS, George: A Comparison of Human and Automatic Musical Genre Classification. In: *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Montreal, 2004
- [LMR02] LAGRANGE, Mathieu ; MARCHAND, Sylvain ; RAULT, Jean-Bernard: Sinusoidal Parameter Extraction and Component Selection in a Non Stationary Model. In: *Proceedings of the 5th International Conference on Digital Audio Effects (DAFX)*. Hamburg, 2002
- [LO04] LI, Tao ; OGIHARA, Mitsunori: Content-based Music Similarity Search and Emotion Detection. In: *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* Bd. 5, IEEE, 2004. – ISBN 0-7803-8484-9, 705–708
- [Log00] LOGAN, Beth: Mel Frequency Cepstral Coefficients for Music Modeling. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Plymouth, 2000
- [Lou90] LOURENS, J G.: Detection and Logging Advertisements using its Sound. In: *Transactions on Broadcasting* 36 (1990), Nr. 3, 231–233. <http://dx.doi.org/10.1109/11.59850>. – DOI 10.1109/11.59850. – ISSN 00189316
- [LP02] LARGE, Edward W. ; PALMER, Caroline: Perceiving Temporal Regularity in Music. In: *Cognitive Science* 26 (2002), Januar, Nr. 1, 1–37. http://doi.wiley.com/10.1207/s15516709cog2601_1. – ISSN 03640213
- [LPLN09] LEE, Honglak ; PHAM, Peter ; LARGMAN, Yan ; NG, Andrew: Unsupervised Feature Learning for Audio Classification using Convolutional Deep Belief Networks. In: *Proceedings of the International Conference on Neural Information Processing Systems (NIPS)* Bd. 22, 2009, 1096–1104
- [LR04] LIVSHIN, Arie A. ; RODET, Xavier: Musical Instrument Identification in Continuous Recordings. In: *Proceedings of the 7th International Conference on Digital Audio Effects (DAFX)*. Naples, 2004
- [LS01a] LEE, Daniel D. ; SEUNG, H S.: Algorithms for Non-negative Matrix Factorization. In: LEEN, T. K. (Hrsg.) ; DIETTERICH, T. G. (Hrsg.) ; TRESP, V. (Hrsg.): *Proceedings of Neural Information Processing Systems (NIPS)*, MIT Press, 2001, 556–562

- [LS01b] LOGAN, Beth ; SALOMON, Ariel: A Music Similarity Function based on Signal Analysis. In: *Proceedings of the International Conference on Multimedia and Expo (ICME)*, 2001, 745–748
- [LS06] LEE, Kyogu ; SLANEY, Malcolm: Automatic Chord Recognition from Audio Using a Supervised HMM Trained with Audio-from-Symbolic Data. In: *Proceedings of the 1st ACM workshop on Audio and music computing multimedia (AMCMM)*. Santa Barbara : ACM, 2006. – 00019
- [LS08] LEVY, Mark ; SANDLER, Mark: Structural Segmentation of Musical Audio by Constrained Clustering. In: *Transactions on Audio, Speech, and Language Processing* 16 (2008), Februar, Nr. 2, 318–326. <http://dx.doi.org/10.1109/TASL.2007.910781>. – DOI 10.1109/TASL.2007.910781. – ISSN 1558–7916
- [LSC⁺19] LOSTANLEN, Vincent ; SALAMON, Justin ; CARTWRIGHT, Mark ; MCFEE, Brian ; FARNSWORTH, Andrew ; KELLING, Steve ; BELLO, Juan P.: Per-Channel Energy Normalization: Why and How. In: *IEEE Signal Processing Letters* 26 (2019), Januar, Nr. 1, S. 39–43. <http://dx.doi.org/10.1109/LSP.2018.2878620>. – DOI 10.1109/LSP.2018.2878620. – ISSN 1558–2361
- [LT07] LARTILLOT, Olivier ; TOIVIAINEN, Petri: A Matlab Toolbox for Musical Feature Extraction from Audio. In: *Proceedings of the 10th International Conference on Digital Audio Effects (DAFX)*. Bordeaux, 2007
- [Luk08] LUKASHEVICH, Hanna: Towards Quantitative Measures of Evaluating Song Segmentation. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Philadelphia, PA, 2008, S. 6
- [Luo15] LUO, Yin-Jyun: Detection of Common Mistakes in Novice Violin Playing. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Malaga, 2015, 316–322
- [LVDV⁺05] LEMAN, Marc ; VERMEULEN, Valery ; DE VOOGDT, Liesbeth ; MOELANTS, Dirk ; LESAFFRE, Micheline: Prediction of Musical Affect Using a Combination of Acoustic Structural Cues. In: *Journal of New Music Research* 34 (2005), März, Nr. 1, 39–67. <http://dx.doi.org/10.1080/09298210500123978>. – DOI 10.1080/09298210500123978. – ISSN 0929–8215
- [LWC98] LIU, Zhu ; WANG, Yao ; CHEN, Tsuhan: Audio feature extraction and analysis for scene classification. In: *Journal of VLSI Signal Processing* (1998), S. 343–348. <http://dx.doi.org/10.1109/MMSP.1997.602659>. – DOI 10.1109/MMSP.1997.602659
- [Mü07] MÜLLER, Meinard: *Information Retrieval for Music and Motion*. Berlin : Springer, 2007. – ISBN 978–3–540–74047–6
- [Mac15] MACPHERSON, Stewart: *Form in Music: With Special Reference to the Designs of Instrumental Music*. London, UK : Joseph Williams, Ltd., 1915 <http://archive.org/details/forminmusicwiths00macp>
- [Mae11] MAEMPEL, Hans-Joachim: Musikaufnahmen als Datenquellen der Interpretationsanalyse. In: LÖSCH, Heinz von (Hrsg.) ; WEINZIERL, Stefan (Hrsg.): *Gemessene Interpretation - Computergestützte Aufführungsanalyse im Kreuzverhör der Disziplinen*. Mainz : Schott, 2011 (Klang und Begriff). – ISBN 978–3–7957–0771–2, S. 157–171
- [Mak75] MAKHOUL, John: Linear Prediction: A Tutorial Review. In: *Proceedings of the IEEE* 63 (1975), April, Nr. 4, S. 561–580. <http://dx.doi.org/10.1109/PROC.1975.9792>. – DOI 10.1109/PROC.1975.9792. – ISSN 1558–2256. – Conference Name: Proceedings of the IEEE
- [Mar04] MAROLT, Matija: A Connectionist Approach to Automatic Transcription of Polyphonic Piano Music. In: *IEEE Transactions on Multimedia* 6 (2004), Juni, Nr. 3, S. 439–449. <http://dx.doi.org/10.1109/TMM.2004.827507>. – DOI 10.1109/TMM.2004.827507. – ISSN 1941–0077. – Conference Name: IEEE Transactions on Multimedia

- [MB96] MASRI, Paul ; BATEMAN, Andrew: Improved Modelling of Attack Transients in Music Analysis-Resynthesis. In: *Proceedings of the International Computer Music Conference (ICMC)*. Hong Kong, 1996
- [MB03] MCKINNEY, Martin F. ; BREEBART, Jeroen: Features for Audio and Music Classification. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Baltimore, 2003
- [MBCHP18] MESEGUR-BROCAL, Gabriel ; COHEN-HADRIA, Alice ; PEETERS, Geoffroy: DALI: A Large Dataset of Synchronized Audio, Lyrics and notes, Automatically Created using Teacher-student Machine Learning Paradigm. In: GÓMEZ, Emilia (Hrsg.) ; HU, Xiao (Hrsg.) ; HUMPHREY, Eric (Hrsg.) ; BENETOS, Emmanouil (Hrsg.): *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Paris, France, 2018, 431–437
- [MBF09] MCKAY, Cory ; BURGOYNE, John A. ; FUJINAGA, Ichiro: jMIR and ACE XML: Tools for Performing and Sharing Research in Automatic Music Classification. In: *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries Workshop on Integrating Digital Library Content with Computational Tools and Services*, 2009
- [MLB09] MAYOR, Oscar ; BONADA, Jordi ; LOSCOS, Alex: Performance Analysis and Scoring of the Singing Voice. In: *Proceedings of the Audio Engineering Society Convention*, 2009, 1–7
- [MC76] MAKHOUL, John ; COSELL, Lynn: LPCW: An LPC vocoder with linear predictive spectral warping. In: *ICASSP '76. IEEE International Conference on Acoustics, Speech, and Signal Processing* Bd. 1, 1976, S. 466–469
- [McA10] McAULEY, J D.: Tempo and Rhythm. Version: 2010. http://dx.doi.org/10.1007/978-1-4419-6114-3{__}. In: RIESS JONES, Mari (Hrsg.) ; FAY, Richard R. (Hrsg.) ; POPPER, Arthur N. (Hrsg.): *Music Perception*. New York, NY : Springer, 2010 (Springer Handbook of Auditory Research). – DOI 10.1007/978-1-4419-6114-3_. – ISBN 978-1-4419-6114-3, 165–199. – 6
- [MCD⁺09] MAUCH, Matthias ; CANNAM, Chris ; DAVIES, Matthew E P. ; DIXON, Simon ; HARTE, Christopher A. ; KOLOZALI, S ; TIDHAR, Dan: OMRAS2 Metadata Project 2009. In: *Late Breaking Demo (Extended Abstract), Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Kobe, 2009, S. 1
- [MCMW03] MAROZEAU, Jeremy ; CHEVEIGNÉ, Alain D. ; MCADAMS, Stephen ; WINSBERG, Suzanne: The dependency of timbre on fundamental frequency. In: *Journal of the Acoustical Society of America (JASA)* 114 (2003), Nr. 5, S. 2946–2957
- [MDP08] MION, Luca ; DE POLI, Giovanni: Score-Independent Audio Features for Description of Music Expression. In: *Transactions on Audio, Speech, and Language Processing* 16 (2008), Februar, Nr. 2, 458–466. <http://dx.doi.org/10.1109/TASL.2007.913743>. – DOI 10.1109/TASL.2007.913743. – ISSN 1558–7916
- [ME13] MAUCH, Matthias ; EWERT, Sebastian: The Audio Degradation Toolbox and its Application to Robustness Evaluation. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Curitiba, Brazil, 2013, S. 6
- [ME14] MCCEE, Brian ; ELLIS, Daniel P W.: Analyzing Song Structure with Spectral Clustering. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Taipei, Taiwan, 2014, S. 6
- [MEF⁺10] MATHIEU, Benoit ; ESSID, Slim ; FILION, Thomas ; PRADO, Jacques ; RICHARD, Gaël: YAAFE, an Easy to Use and Efficient Audio Feature Extraction Software. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2010, S. 441–446

- [Meu02] MEUDIC, Benoit: A Causal Algorithm for Beat Tracking. In: *Proceedings of the 2nd International Conference on Understanding and Creating Music*. Caserta, November 2002
- [Mey56] MEYER, Leonard B.: *Emotion and Meaning in Music*. Chicago : University of Chicago Press, 1956
- [MF06] MCKAY, Cory ; FUJINAGA, Ichiro: Musical Genre Classification: Is it Worth Pursuing and How can it be Improved? In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Victoria, 2006, S. 101–106
- [MF09] MCKAY, Cory ; FUJINAGA, Ichiro: jMIR: Tools for Automatic Music Classification. In: *Proceedings of the International Computer Music Conference (ICMC)*, ICMA, 2009. – 00014
- [MG83] MOORE, Brian C J. ; GLASBERG, Brian R.: Suggested formulae for calculating auditory-filter bandwidths and excitation patterns. In: *Journal of the Acoustical Society of America (JASA)* 74 (1983), Nr. 3, S. 750–753
- [MH92] MEDDIS, Ray ; HEWITT, Michael J.: Modeling the identification of concurrent vowels with different fundamental frequencies. In: *Journal of the Acoustical Society of America (JASA)* 91 (1992), Januar, Nr. 1, 233–245. <http://dx.doi.org/10.1121/1.402767>. – DOI 10.1121/1.402767. – ISSN 0001–4966
- [MH01] MERON, Yoram ; HIROSE, Keikichi: Automatic alignment of a musical score to performed music. In: *Acoustical Science & Technology* 22 (2001), Nr. 3, S. 189–198
- [MHB15] MCFEE, Brian ; HUMPHREY, Eric J. ; BELLO, Juan P.: A Software Framework for Musical Data Augmentation. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Malaga, Spain, 2015, S. 7
- [MID01] MIDI MANUFACTURERS ASSOCIATION: Complete {MIDI} 1.0 Detailed Specification V96.1, 2nd edition / MMA. 2001. – Standard
- [MJ13] MARXER, Ricard ; JANER, Jordi: Study of Regularizations and Constraints in NMF-based Drums Monaural Separation. In: *Proceedings of the International Conference on Digital Audio Effects (DAFX)*. Maynooth, 2013, S. 6
- [MKR04] MÜLLER, Meinard ; KURTH, Frank ; RÖDER, Tido: Towards an Efficient Algorithm for Automatic Score-to-Audio Synchronization. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Barcelona, 2004
- [MM99] MARQUES, Janet ; MORENO, Pedro J.: A Study of Musical Instrument Classification Using Gaussian Mixture Models and Support Vector Machines / Cambridge Research Laboratory CRL 99/4. 1999 (June). – Forschungsbericht
- [MM01] MCADAMS, Stephen ; MATZKIN, Daniel: Similarity, Invariance, and Musical Variation. In: *Annals of the New York Academy of Sciences* 930 (2001), Januar, Nr. 1, 62–76. <http://dx.doi.org/10.1111/j.1749-6632.2001.tb05725.x>. – DOI 10.1111/j.1749–6632.2001.tb05725.x. – ISSN 00778923
- [MM04] MCKINNEY, Martin F. ; MOELANTS, Dirk: Deviations from the Resonance Theory of Tempo Induction. In: PARNCUTT, Richard (Hrsg.) ; KESSLER, A (Hrsg.) ; ZIMMER, F (Hrsg.): *Proceedings of the Conference on Interdisciplinary Musicology*. Graz, 2004
- [MM06] MCKINNEY, Martin F. ; MOELANTS, Dirk: Ambiguity in Tempo Perception: What Draws Listeners to Different Metrical Levels? In: *Music Perception* 24 (2006), Dezember, Nr. 2, 155–166. <http://dx.doi.org/10.1525/mp.2006.24.2.155>. – DOI 10.1525/mp.2006.24.2.155. – ISSN 0730–7829

- [MM12] MARKOV, Konstantin ; MATSUI, Tomoko: Music Genre Classification Using Self-Taught Learning via Sparse Coding. In: *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Kyoto : Institute of Electrical and Electronics Engineers (IEEE), März 2012, S. 1929–1932. – ISSN: 2379-190X
- [MMDK07] MCKINNEY, Martin F. ; MOELANTS, Dirk ; DAVIES, Matthew E P. ; Klapuri, Anssi P.: Evaluation of Audio Beat Tracking and Music Tempo Extraction Algorithms. In: *Journal of New Music Research* 36 (2007), März, Nr. 1, 1–16. <http://dx.doi.org/10.1080/09298210701653252>. – DOI 10.1080/09298210701653252. – ISSN 0929–8215. – Publisher: Routledge _eprint: <https://doi.org/10.1080/09298210701653252>
- [MMK06] MÜLLER, Meinard ; MATTES, Henning ; KURTH, Frank: An Efficient Multiscale Approach to Audio Synchronization. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Victoria, 2006
- [MNFB17] MFEE, Brian ; NIETO, Oriol ; FARBOOD, Morwaread M. ; BELLO, Juan P.: Evaluating Hierarchical Structure in Music Annotations. In: *Frontiers in Psychology* 0 (2017). <http://dx.doi.org/10.3389/fpsyg.2017.01337>. – DOI 10.3389/fpsyg.2017.01337. – ISSN 1664–1078. – Publisher: Frontiers
- [MO97] MEDDIS, Ray ; O'MARD, Lowel: A Unitary Model of Pitch Perception. In: *Journal of the Acoustical Society of America (JASA)* 102 (1997), September, Nr. 3, 1811–20. <http://www.ncbi.nlm.nih.gov/pubmed/9714929>. – ISSN 0001–4966
- [MOCC07] MACDORMAN, Karl F. ; OUGH, Stuart ; CHIN-CHANG, Ho: Automatic Emotion Prediction of Song Excerpts: Index Construction, Algorithm Design, and Empirical Comparison. In: *Journal of New Music Research* 36 (2007), Dezember, Nr. 4, 281–299. <http://dx.doi.org/10.1080/09298210801927846>. – DOI 10.1080/09298210801927846. – ISSN 0929–8215
- [Moo97] MOORE, Brian C J.: *An Introduction to the Psychology of Hearing*. 4th Editio. London : Academic Press, 1997
- [Moo01] MOORE, Allan F.: Categorical Conventions in Music Discourse: Style and Genre. In: *Music & Letters* 82 (2001), Nr. 3, 432–442. <https://www.jstor.org/stable/3526163>. – ISSN 0027–4224. – Publisher: Oxford University Press
- [MP16] MARCHAND, Ugo ; PEETERS, Geoffroy: The Extended Ballroom Dataset. In: *Late Breaking Demo (Extended Abstract), Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. New York, 2016, S. 4
- [MP19] MIGNOT, Rémi ; PEETERS, Geoffroy: An Analysis of the Effect of Data Augmentation Methods: Experiments for a Musical Genre Classification Task. In: *Transactions of the International Society for Music Information Retrieval* 2 (2019), Dezember, Nr. 1, 97–110. <http://dx.doi.org/10.5334/tismir.26>. – DOI 10.5334/tismir.26. – ISSN 2514–3298
- [MPSN01] MOLAU, Sirko ; PITZ, Michael ; SCHLÜTER, Ralf ; NEY, Hermann: Computing Mel-frequency cepstral coefficients on the power spectrum. In: *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing. (ICASSP01)* (2001), 73–76. <http://dx.doi.org/10.1109/ICASSP.2001.940770>. – DOI 10.1109/ICASSP.2001.940770
- [MS01] MILES, Jeremy ; SHEVLIN, Mark: *Applying Regression & Correlation: a Guide for Students and Researchers*. London : Sage Publications, 2001. – ISBN 0–7619–6229–8
- [MS18] MCLEOD, Andrew ; STEEDMAN, Mark: Evaluating Automatic Polyphonic Music Transcription. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Paris, 2018, S. 8

- [MSLS11] MILNE, Andrew J. ; SETHARES, William A. ; LANEY, Robin ; SHARP, David B.: Modelling the Similarity of Pitch Collections with Expectation Tensors. In: *Journal of Mathematics and Music* 5 (2011), März, Nr. 1, 1–20. <http://dx.doi.org/10.1080/17459737.2011.573678>. – DOI 10.1080/17459737.2011.573678. – ISSN 1745–9737
- [MTS99] MIWA, T. ; TADOKORO, Y. ; SAITO, T.: Musical pitch estimation and discrimination of musical instruments using comb filters for transcription. In: *42nd Midwest Symposium on Circuits and Systems (Cat. No.99CH36356)* Bd. 1, 1999, S. 105–108 vol. 1
- [MUTL06] MÖRCHEN, Fabian ; ULTSCH, Alefred ; THIES, Michael ; LÖHKEN, Ingo: Modeling timbre distance with temporal statistics from polyphonic music. In: *Transactions on Audio, Speech and Language Processing* 14 (2006), Januar, Nr. 1, 81–90. <http://dx.doi.org/10.1109/TSA.2005.860352>. – DOI 10.1109/TSA.2005.860352. – ISSN 1558–7916
- [MVSP11] MARTÍNEZ, José I. ; VITOLA, Jaime ; SANABRIA, Adriana ; PEDRAZA, Cesar: Fast Parallel Audio Fingerprinting Implementation in Reconfigurable Hardware and GPUs. In: *Proceedings of the Southern Conference on Programmable Logic (SPL)*, IEEE, April 2011. – ISBN 978–1–4244–8847–6, 245–250
- [MWD⁺95] MCADAMS, Stephen ; WINSBERG, Suzanne ; DONNADIEU, Sophie ; SOETE, Geert D. ; KRIMPHOFF, Jochen: Perceptual scaling of synthesized musical timbres: Common dimensions, specificities, and latent subject classes. In: *Psychological Research* 58 (1995), Nr. 3, 177–192. <http://dx.doi.org/10.1007/BF00419633>. – DOI 10.1007/BF00419633. – ISSN 0340–0727, 1430–2772
- [NAG19] NADAR, Christon-Ragavan ; ABESSER, Jakob ; GROLLMISCH, Sascha: Towards CNN-based Acoustic Modeling of Seventh Chords for Automatic Chord Recognition. In: *Proceedings of the Sound and Music Computing Conference (SMC)*. Malaga, Spain, 2019, S. 7
- [Nak87] NAKAMURA, Toshie: The communication of dynamics between musicians and listeners through musical performance. In: *Perception & Psychophysics* 41 (1987), Nr. 6, S. 525–533
- [Ng18] NG, Andrew: *Machine Learning Yearning: Technical Strategy for AI Engineers, In the Era of Deep Learning*. to be published, 2018 <https://www.deeplearning.ai/machine-learning-yearning/>
- [NGH06] NAKANO, Tomoyasu ; GOTO, Masataka ; HIRAGA, Yuzuru: An automatic singing skill evaluation method for unknown melodies using pitch interval accuracy and vibrato features. In: *Rn* 12 (2006), 1. <https://staff.aist.go.jp/t.nakano/PAPER/INTERSPEECH2006nakano.pdf>
- [NHSS12] NAM, Juhan ; HERRERA, Jorge ; SLANEY, Malcolm ; SMITH, Julius: Learning Sparse Feature Representations For Music Annotation and Retrieval. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Porto, Portugal, 2012, S. 6
- [NJ13] NIETO, Oriol ; JEHAN, Tristan: Convex Non-negative Matrix Factorization for Automatic Music Structure Identification. In: *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2013, S. 236–240
- [NMK07] NOVELLO, Alberto ; MCKINNEY, Martin F. ; KOHLRAUSCH, Armin: Perceptual Evaluation of Music Similarity. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Victoria, 2007
- [NMW⁺20] NIETO, Oriol ; MYSORE, Gautham J. ; WANG, Cheng-i ; SMITH, Jordan B L. ; SCHLÜTER, Jan ; GRILL, Thomas ; MFEE, Brian: Audio-Based Music Structure Analysis: Current Trends, Open Challenges, and Applications. In: *Transactions of the International Society for Music Information Retrieval (TISMIR)* 3 (2020), Nr. 1, S. 246–263

- [Nol64] NOLL, A M.: Short-Time Spectrum and “Cepstrum” Techniques for Vocal-Pitch Detection. In: *Journal of the Acoustical Society of America (JASA)* 36 (1964), Nr. 2, 296. <http://dx.doi.org/10.1121/1.1918949>. – DOI 10.1121/1.1918949. – ISSN 00014966
- [Nol69] NOLL, A M.: Pitch Determination of Human Speech by the Harmonic Product Spectrum, the Harmonic Sum Spectrum, and a Maximum Likelihood Estimate. In: *Proceedings of the Symposium on Computer Processing in Communications* Bd. 19. Brooklyn : Polytechnic Press of the University of Brooklyn, 1969, S. 779–797
- [NPB16] NIETO, Oriol ; PABLO BELLO, Juan: Systematic Exploration of Computational Music Structure Research. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2016, 547–553
- [NR17] NARANG, Krish ; RAO, Preeti: Acoustic Features for Determining Goodness of Tabla Strokes. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Suzhou, China, 2017, S. 257–263
- [Nyq28] NYQUIST, Harry: Certain Topics in Telegraph Transmission Theory. In: *Transactions of the American Institute of Electrical Engineers* 47 (1928), April, Nr. 2, 617–644. <http://dx.doi.org/10.1109/T-AIEE.1928.5055024>. – DOI 10.1109/T-AIEE.1928.5055024. – ISSN 0096–3860. – 01598
- [OD01] ORIO, Nicola ; DÉCHELLE, Francois: Score Following Using Spectral Analysis and Hidden Markov Models. In: *Proceedings of the International Computer Music Conference (ICMC)*. Habana, September 2001
- [OFG11] OUDRE, Laurent ; FEVOTTE, Cédric ; GRENIER, Yves: Probabilistic Template-Based Chord Recognition. In: *IEEE Transactions on Audio, Speech, and Language Processing* 19 (2011), November, Nr. 8, S. 2249–2259. <http://dx.doi.org/10.1109/TASL.2010.2098870>. – DOI 10.1109/TASL.2010.2098870. – ISSN 1558–7924
- [OJS71] OPPENHEIM, Alan V. ; JOHNSON, Donald H. ; STEIGLITZ, K: Computation of Spectra with Unequal Resolution using the Fast Fourier Transform. In: *Proceedings of the IEEE* 59 (1971), Nr. 2, 299–301. <http://dx.doi.org/10.1109/PROC.1971.8146>. – DOI 10.1109/PROC.1971.8146. – ISSN 0018–9219
- [OL07] OHM, Jens-Rainer ; LÜKE, Hans D.: *Signalübertragung*. 10. Berlin, Heidelberg : Springer Berlin Heidelberg, 2007 (Springer-Lehrbuch). <http://www.springerlink.com/index/10.1007/978-3-540-69258-4>. – ISBN 978–3–540–69256–0
- [OL15] O'BRIEN, Cian ; LERCH, Alexander: Genre-Specific Key Profiles. In: *Proceedings of the International Computer Music Conference (ICMC)*. Denton : ICMA, 2015
- [OLS03] ORIO, Nicola ; LEMOUTON, Serge ; SCHWARZ, Diemo: Score Following: State of the Art and New Developments. In: *Proceedings of the Conference of New Interfaces for Musical Expression (NIME)*. Montreal, 2003
- [Ori06] ORIO, Nicola: Music Retrieval: A Tutorial and Review. In: *Foundations and Trends in Information Retrieval* 1 (2006), Nr. 1, S. 1–90
- [O'S87] O'SHAUGHNESSY, Douglas: *Speech Communication: Human and Machine*. reprint. Addison-Wesley, 1987. – ISBN 978–0–201–16520–3
- [OS99] OPPENHEIM, Alan V. ; SCHAFER, Ronald W.: *Discrete-Time Signal Processing*. 2nd. Upper Saddle River : Prentice Hall, 1999. – ISBN 0–13–754920–2. – 05004
- [OS01] ORIO, Nicola ; SCHWARZ, Diemo: Alignment of Monophonic and Polyphonic Music to a Score. In: *Proceedings of the International Computer Music Conference (ICMC)*. Habana, September 2001

- [Pal97] PALMER, Caroline: Music Performance. In: *Annual Review of Psychology* 48 (1997), S. 115–138
- [Par76] PARSONS, Thomas W.: Separation of speech from interfering speech by means of harmonic selection. In: *Journal of the Acoustical Society of America (JASA)* 60 (1976), Nr. 4, 911. <http://dx.doi.org/10.1121/1.381172>. – DOI 10.1121/1.381172. – ISSN 00014966. – 00257
- [Par03] PARNCUTT, Richard: Accents and expression in piano performance. In: NIEMÖLLER, Klaus W. (Hrsg.): *Perspektiven und Methoden einer Systemischen Musikwissenschaft*. Frankfurt/Main : Peter Lang, 2003
- [Pau04] PAUWS, Steffen: Musical key extraction from audio. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Barcelona, 2004
- [PBD16] PANTELI, Maria ; BENETOS, Emmanouil ; DIXON, Simon: Learning a Feature Space for Similarity in World Music. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. New York, 2016, S. 7
- [PBO00] PURWINS, Hendrik ; BLANKERTZ, Benjamin ; OBERMAYER, Klaus: A New Method for Tracking Modulations in Tonal Music in Audio Data Format. In: *IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN'00)* Bd. 6. Como, 2000, S. 6270–6275
- [PC00] PACHET, Francois ; CAZALY, Daniel: A Taxonomy of Musical Genres. In: *Proceedings of the Conference on Content-Based Multimedia Information Access*. Paris, April 2000. – 00219
- [PCZ⁺19] PARK, Daniel S. ; CHAN, William ; ZHANG, Yu ; CHIU, Chung-Cheng ; ZOPH, Barret ; CUBUK, Ekin D. ; LE, Quoc V.: SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition. In: *Proceedings of the Conference of the International Speech Communication Association (Interspeech)*. Graz, Austria, September 2019, 2613–2617. – arXiv: 1904.08779
- [PDW03] PAMPALK, Elias ; DIXON, Simon ; WIDMER, Gerhard: On the Evaluation of Perceptual Similarity Measures for Music. In: *Proceedings of the 6th International Conference on Digital Audio Effects (DAFx)*. London, September 2003
- [PE06] POLINER, Graham E. ; ELLIS, Daniel P. W.: A Discriminative Model for Polyphonic Piano Transcription. In: *EURASIP Journal on Advances in Signal Processing* 2007 (2006), Dezember, Nr. 1, 048317. <http://dx.doi.org/10.1155/2007/48317>. – DOI 10.1155/2007/48317. – ISSN 1687–6180
- [Pee04] PEETERS, Geoffroy: A large set of audio features for sound description (similarity and classification) in the CUIDADO project / IRCAM. 2004. – Project Report (CUIDADO)
- [Pee05] PEETERS, Geoffroy: Time variable tempo detection and beat marking. In: *Proceedings of the International Computer Music Conference (ICMC)*. Barcelona, September 2005
- [Pee06] PEETERS, Geoffroy: Musical Key Estimation of Audio Signal Based on Hidden Markov Modeling of Chroma Vectors. In: *Proceedings of the International Conference on Digital Audio Effects (DAFx)*. Montreal, Quebec, Canada, 2006
- [Pee07] PEETERS, Geoffroy: A Generic System for Audio Indexing: Application to Speech/Music Segmentation and Music Genre Recognition. In: *Proceedings of the International Conference on Digital Audio Effects (DAFx)*. Bordeaux, 2007, S. 8
- [PFW05] PAMPALK, Elias ; FLEXER, Arthur ; WIDMER, Gerhard: Improvements of Audio-Based Music Similarity and Genre Classification. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. London, UK, 2005, S. 8
- [PGL18] PATI, Kumar A. ; GURURANI, Siddharth ; LERCH, Alexander: Assessment of Student Music Performances Using Deep Neural Networks. In: *Applied Sciences* 8 (2018), März, Nr. 4, 507. <http://dx.doi.org/10.3390/app8040507>. – DOI 10.3390/app8040507

- [PGS⁺11] PEETERS, Geoffroy ; GIORDANO, Bruno L. ; SUSINI, Patrick ; MISDARIIS, Nicolas ; MCADAMS, Stephen: The Timbre Toolbox: Extracting Audio Descriptors from Musical Signals. In: *Journal of the Acoustical Society of America (JASA)* 130 (2011), November, Nr. 5, 2902. <http://dx.doi.org/10.1121/1.3642604>. – DOI 10.1121/1.3642604. – ISSN 1520–8524
- [PL92] PUCKETTE, Miller S. ; LIPPE, Cort: Score Following in Practice. In: *Proceedings of the International Computer Music Conference (ICMC)*. San Francisco, 1992
- [PM87] PITT, Mark A. ; MONAHAN, Caroline B.: The perceived similarity of auditory polyrhythms. In: *Perception & Psychophysics* 41 (1987), Juni, Nr. 6, 534–546. <http://www.ncbi.nlm.nih.gov/pubmed/3615150>. – ISSN 0031–5117
- [PMK10] PAULUS, Jouni ; MULLER, Meinard ; Klapuri, Anssi P.: State of the Art Report: Audio-Based Music Structure Analysis. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Utrecht, Netherlands, 2010, 625–636
- [PMP20] PANDA, Renato ; MALHEIRO, Ricardo ; PAIVA, Rui P.: Novel Audio Features for Music Emotion Recognition. In: *IEEE Transactions on Affective Computing* 11 (2020), Oktober, Nr. 4, S. 614–626. <http://dx.doi.org/10.1109/TAFFC.2018.2820691>. – DOI 10.1109/TAFFC.2018.2820691. – ISSN 1949–3045. – Conference Name: IEEE Transactions on Affective Computing
- [PNP⁺18] PONS, Jordi ; NIETO, Oriol ; PROCKUP, Matthew ; SCHMIDT, Erik M. ; EHMANN, Andreas F. ; SERRA, Xavier: End-to-end Learning for Music Audio Tagging at Scale. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Paris, France, September 2018
- [POGS19] PAUWELS, Johan ; O'HANLON, Ken ; GOMEZ, Emilia ; SANDLER, Mark B.: 20 Years of Automatic Chord Recognition from Audio. In: *Proceedings of the 20th Conference of the International Society for Music Information (ISMIR)*. Delft, The Netherlands, November 2019
- [Pov77] POVEL, Dirk-Jan: Temporal Structure of Performed Music. Some Preliminary Observations. In: *Acta Psychologica* Bd. 41, 1977, S. 309–320
- [PP01] PFEIFFER, Silvia ; PARKER, Conrad: bewdy, Maaate! In: *Presentation at the Australian Linux Conference*. Sydney, 2001
- [PP07] PAPADOPOULOS, Hélène ; PEETERS, Geoffroy: Large-scale study of chord estimation algorithms based on chroma representation and HMM. In: *Proceedings of the International Workshop on Content-Based Multimedia Indexing (CBMI)*. Bordeaux, 2007
- [PP13] PAUWELS, Johan ; PEETERS, Geoffroy: Evaluating Automatically Estimated Chord Sequences. In: *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Vancouver, BC, Canada, Mai 2013, S. 749–753. – ISSN: 2379-190X
- [PR98] PEETERS, Geoffroy ; RODET, Xavier: Signal Characterization in terms of Sinusoidal and Non-Sinusoidal Components. In: *Proceedings of the 1st Conference on Digital Audio Effects (DAFX)*. Barcelona, 1998
- [PRM02] PAMPALK, Elias ; RAUBER, Andreas ; MERKL, Dieter: Content-based organization and visualization of music archives. In: *Proceedings of the 10th ACM International Conference on Multimedia* (2002), 570. <http://dx.doi.org/10.1145/641007.641121>. – DOI 10.1145/641007.641121
- [Pro97] PROUT, Ebenezer: *Musical Form*. London, UK : Augener, 1897 <http://archive.org/details/musicalform00prouuoft>

- [PS19] PONS, Jordi ; SERRA, Xavier: musicnn: Pre-Trained Convolutional Neural Networks for Music Audio Tagging. In: *Late Breaking Demo (Extended Abstract), Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Delft, Netherlands, 2019, 2
- [PT14] PERCIVAL, Graham ; TZANETAKIS, George: Streamlined Tempo Estimation Based on Auto-correlation and Cross-correlation With Pulses. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 22 (2014), Dezember, Nr. 12, S. 1765–1776. <http://dx.doi.org/10.1109/TASLP.2014.2348916>. – DOI 10.1109/TASLP.2014.2348916. – ISSN 2329–9304. – Conference Name: IEEE/ACM Transactions on Audio, Speech, and Language Processing
- [Puc95] PUCKETTE, Miller S.: Score Following using the sung voice. In: *Proceedings of the International Computer Music Conference (ICMC)*. Banff, 1995
- [PVM08] PAUWELS, Johan ; VAREWYCK, Matthias ; MARTENS, Jean-Pierre: Audio Chord Extraction using a Probabilistic Model. In: *Proceedings of the Music Information Retrieval EXchange (MIREX) at the 9th International Conference on Music Information Retrieval (ISMIR)*. Philadelphia, September 2008
- [QL19] QIN, Yi ; LERCH, Alexander: Tuning Frequency Dependency in Music Classification. In: *Proceedings of the International Conference on Acoustics Speech and Signal Processing (ICASSP)*. Brighton, UK : Institute of Electrical and Electronics Engineers (IEEE), 2019, 401–405
- [Rö05] RÖBEL, Axel: Onset Detection in Polyphonic Signals by means of Transient Peak Classification. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2005
- [Rab89] RABINER, Lawrence R.: A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. In: *Proceedings of the IEEE* 77 (1989), Nr. 2, 257–286. <http://dx.doi.org/10.1109/5.18626>. – DOI 10.1109/5.18626. – ISSN 00189219
- [Rap99] RAPHAEL, Christopher: Automatic Segmentation of Acoustic Musical Signals Using Hidden Markov Models. In: *Transactions on Pattern Analysis and Machine Intelligence* 21 (1999), Nr. 4, S. 360–370
- [Rap01] RAPHAEL, Christopher: A Probabilistic Expert System for Automatic Musical Accompaniment. In: *Journal of Computational and Graphical Statistics* 10 (2001), Nr. 3, S. 487–512
- [Rap04] RAPHAEL, Christopher: A Hybrid Graphical Model for Aligning Polyphonic Audio with Musical Scores. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Barcelona, 2004
- [Ras79] RASCH, Rudolf A.: Synchronization in Performed Ensemble Music. In: *Acustica* 43 (1979), S. 121–131
- [RB93] RIEDMILLER, Martin ; BRAUN, Heinrich: A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm. In: *Proceedings of the International Conference on Neural Networks*. San Francisco : IEEE, 1993
- [RB18] REISS, Joshua D. ; BRANDTSEGG, Øyvind: Applications of Cross-Adaptive Audio Effects: Automatic Mixing, Live Performance and Everything in Between. In: *Frontiers in Digital Humanities* 5 (2018). <http://dx.doi.org/10.3389/fdigh.2018.00017>. – DOI 10.3389/fdigh.2018.00017. – ISSN 2297–2668
- [Rep92] REPP, Bruno H.: Diversity and commonality in music performance: An analysis of timing microstructure in Schumann's 'Träumerei'. In: *Journal of the Acoustical Society of America (JASA)* 92 (1992), Nr. 5, S. 2546–2568

- [Rep94] REPP, Bruno H.: On Determining the Basic Tempo of an Expressive Music Performance. In: *Psychology of Music* 22 (1994), Nr. 2, S. 157–167
- [Rep96a] REPP, Bruno H.: The dynamics of expressive piano performance: Schumann's 'Träumerei' revisited. In: *Journal of the Acoustical Society of America (JASA)* 100 (1996), Nr. 1, S. 641–650
- [Rep96b] REPP, Bruno H.: Patterns of note onset asynchronies in expressive piano performance. In: *Journal of the Acoustical Society of America (JASA)* 100 (1996), Nr. 6, S. 3917–3932
- [Reu95] REUTER, Christoph: *Der Einschwingvorgang nichtperkussiver Musikanstrumente*. Frankfurt : Peter Lang, 1995
- [RGM94] ROSENTHAL, David ; GOTO, Masataka ; MURAOKA, Yoichi: Rhythm Tracking Using Multiple Hypotheses. In: *Proceedings of the International Computer Music Conference (ICMC)*. Aarhus, September 1994
- [RK06] RAMALINGAM, Arunan ; KRISHNAN, Sridhar: Gaussian Mixture Modeling of Short-Time Fourier Transform Features for Audio Fingerprinting. In: *Transactions on Information Forensics and Security* 1 (2006), Dezember, Nr. 4, 457–463. <http://dx.doi.org/10.1109/TIFS.2006.885036>. – DOI 10.1109/TIFS.2006.885036. – ISSN 1556–6013
- [RK08] RYYNÄNEN, Matti P. ; K LAPURI, Anssi P.: Automatic Transcription of Melody, Bass Line, and Chords in Polyphonic Music. In: *Computer Music Journal* 32 (2008), September, Nr. 3, 72–86. <http://dx.doi.org/10.1162/comj.2008.32.3.72>. – DOI 10.1162/comj.2008.32.3.72. – ISSN 0148–9267
- [RMH⁺14] RAFFEL, Colin ; McFEE, Brian ; HUMPHREY, Eric J. ; SALAMON, Justin ; NIETO, Oriol ; LIANG, Dawen ; ELLIS, Daniel P W.: mir_eval: A Transparent Implementation of Common MIR Metrics. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Taipei, Taiwan, 2014, S. 6
- [RP82] RASCH, Rudolf A. ; PLOMB, Reinier: The Perception of Musical Tones. In: DEUTSCH, Diana (Hrsg.): *The Psychology of Music*. New York : Academic Press, 1982
- [RPRD⁺15] ROMANI PICAS, Oriol ; RODRIGUEZ, H P. ; DABIRI, Dara ; TOKUDA, Hiroshi ; HARIYA, Wataru ; OISHI, Koji ; SERRA, Xavier: A Real-Time System for Measuring Sound Goodness in Instrumental Sounds. In: *Proceedings of the Audio Engineering Society Convention Bd.* 138. Warsaw, 2015
- [RR05] RÖBEL, Axel ; RODET, Xavier: Efficient spectral envelope estimation and its application to pitch shifting and envelope preservation. In: *Proc. DAFX*, 2005
- [RS78] RABINER, Lawrence R. ; SCHAFER, Ronald W.: *Digital Processing of Speech Signals*. New Jersey : Prentice Hall, 1978. – ISBN 0–13–213603–1
- [RSC⁺74] ROSS, Myron J. ; SHAFFER, Harry L. ; COHEN, Andrew ; FREUDBERG, Richard ; MANLEY, Harold J.: Average Magnitude Difference Function Pitch Extractor. In: *Transactions on Acoustics, Speech, and Signal Processing* 22 (1974), Oktober, Nr. 5, 353–362. <http://dx.doi.org/10.1109/TASSP.1974.1162598>. – DOI 10.1109/TASSP.1974.1162598. – ISSN 0096–3518
- [Rus80] RUSSEL, James A.: A Circumplex Model of Affect. In: *Journal of Personality and Social Psychology* 39 (1980), Nr. 6, S. 1161–1178. <http://dx.doi.org/10.1037/h0077714>. – DOI 10.1037/h0077714. – ISSN 1939–1315(Electronic);0022–3514(Print)
- [RVKH00] RICHLY, Gábor ; VARGA, László ; KOVACS, Ferenc ; HOSSZU, Gábor: Short-term Sound Stream Characterization for Reliable, Real-Time Occurrence Monitoring of Given Sound-Prints. In: *Proceedings of the 10th Mediterranean Electrotechnical Conference. Information Technology and Electrotechnology for the Mediterranean Countries (CMeleCon)* Bd. 2, Ieee, 2000. – ISBN 0–7803–6290–X, 526–528. – 00011

- [RWK⁺20] RAMANUJAN, Vivek ; WORTSMAN, Mitchell ; KEMBAVI, Aniruddha ; FARHADI, Ali ; RASTEGARI, Mohammad: What's Hidden in a Randomly Weighted Neural Network? In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Online, 2020, 11893–11902
- [Ryy04] RYYNÄNEN, Matti P.: *Probabilistic Modelling of Note Events in the Transcription of Monophonic Melodies*, Tampere University of Technology, Diplomarbeit, 2004
- [RZR04] RÖBEL, Axel ; ZIVANOVIC, Miroslav ; RODET, Xavier: Signal decomposition by means of classification of spectral peaks. In: *Proceedings of the International Computer Music Conference (ICMC)*. Miami : ICMA, 2004
- [SA02] SUKITTANON, Somsak ; ATLAS, Les E.: Modulation Frequency Features for Audio Fingerprinting. In: *Proceedings of the International Conference on Acoustics Speech and Signal Processing (ICASSP)*, IEEE, 2002. – ISBN 0-7803-0946-4, II-II
- [SAH79] SCHROEDER, M R. ; ATAL, B S. ; HALL, J L.: Optimizing digital speech coders by exploiting masking properties of the human ear. In: *Journal of the Acoustical Society of America (JASA)* 66 (1979), Nr. 6, S. 1647–1652
- [Sau96] SAUNDERS, John: Real-Time Discrimination of Broadcast Speech/Music. In: *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Atlanta, 1996
- [SB03] SMARAGDIS, Paris ; BROWN, Judith C.: Non-Negative Matrix Factorization for Polyphonic Music Transcription. In: *Proceedings of the Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. New Paltz : IEEE, 2003. – ISBN 0-7803-7850-4
- [SB14] SCHLÜTER, Jan ; BÖCK, Sebastian: Improved Musical Onset Detection with Convolutional Neural Networks. In: *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Florence, Italy : Institute of Electrical and Electronics Engineers (IEEE), Mai 2014, S. 6979–6983
- [SBF⁺11] SMITH, Jordan B L. ; BURGOYNE, J A. ; FUJINAGA, Ichiro ; ROURE, David D. ; DOWNIE, J S.: Design and Creation of a Large-Scale Database of Structural Annotations. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Miami, FL, 2011, S. 7
- [SBZ10] SANDEN, Chris ; BEFUS, Chad R. ; ZHANG, John Z.: CAMEL: A Lightweight Framework for Content-Based Audio and Music Analysis. In: *Proceedings of the 5th Audio Mostly Conference on Interaction with Sound (AM)*. New York, New York, USA : ACM Press, 2010. – ISBN 978-1-4503-0046-9
- [SC78] SAKOE, Hiroaki ; CHIBA, Seibi: Dynamic Programming Algorithm Optimization for Spoken Word Recognition. In: *Transactions on Acoustics, Speech, and Signal Processing* 26 (1978), Nr. 1, S. 43–49
- [Sch38] SCHOUTEN, Jan F.: The Perception of Subjective Tones. (1938), S. 5
- [Sch40] SCHOEN, Max: *The Psychology of Music*. New York : The Ronald Press Company, 1940
- [Sch68] SCHROEDER, M R.: Period Histogram and Product Spectrum: New Methods for Fundamental-Frequency Measurement. In: *Journal of the Acoustical Society of America (JASA)* 43 (1968), Nr. 4, 829. <http://dx.doi.org/10.1121/1.1910902>. – DOI 10.1121/1.1910902. – ISSN 00014966. – 00226
- [Sch85] SCHLOSS, W A.: *On the Automatic Transcription of Percussive Music – From Acoustic Signal to High-Level Analysis*. Stanford, Stanford University, Center for Computer Research in Music and Acoustics (CCRMA), Dissertation, 1985

- [Sch95] SCHEIRER, Eric D.: Using Musical Knowledge to Extract Expressive Performance Information from Audio Recordings. In: *Proceedings?*, 1995
- [Sch98] SCHEIRER, Eric D.: Tempo and beat analysis of acoustic musical signals. In: *Journal of the Acoustical Society of America (JASA)* 103 (1998), Nr. 1, S. 588–601
- [Sch99] SCHEIRER, Eric D.: Towards Music Understanding without Separation: Segmenting Music with Correlogram Comodulation. In: *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. New Paltz, NY, USA, Oktober 1999, S. 99–102
- [Sch03a] SCHERER, Klaus R.: Why Music does not Produce Basic Emotions: Pleading for a new Approach to Measuring the Emotional Effects of Music. In: *Proceedings of the Stockholm Music Acoustics Conference (SMAC)*. Stockholm, August 2003
- [Sch03b] SCHUBERT, Emery: Update of the Hevner Adjective Checklist. In: *Perceptual and Motor Skills* 96 (2003), S. 1117–1122
- [Sch04a] SCHERER, Klaus R.: Which Emotions Can be Induced by Music? What Are the Underlying Mechanisms? And How Can We Measure Them? In: *Journal of New Music Research* 33 (2004), September, Nr. 3, 239–251. <http://dx.doi.org/10.1080/0929821042000317822>. – DOI 10.1080/0929821042000317822. – ISSN 0929–8215
- [Sch04b] SCHUBERT, Emery: Modeling Perceived Emotion With Continuous Musical Features. In: *Music Perception* 21 (2004), Nr. 4, S. 561–585
- [Sch05] SCHERER, Klaus R.: What are emotions? And how can they be measured? In: *Social Science Information* 44 (2005), Dezember, Nr. 4, 695–729. <http://dx.doi.org/10.1177/0539018405058216>. – DOI 10.1177/0539018405058216. – ISSN 0539–0184
- [Sch06] SCHEDL, Markus: The CoMIRVA Toolkit for Visualizing Music-Related Data / Johannes Kepler University Linz. Version: 2006. <http://www.cp.jku.at/people/schedl/Research/Development/CoMIRVA/webpage/CoMIRVA.html>. 2006. – Forschungsbericht. – 00000
- [SCS⁺13] SOLEYMANI, Mohammad ; CARO, Micheal N. ; SCHMIDT, Erik M. ; SHA, Cheng-Ya ; YANG, Yi-Hsuan: 1000 Songs for Emotional Analysis of Music. In: *Proceedings of the 2nd ACM international workshop on Crowdsourcing for multimedia - CrowdMM '13*. Barcelona, Spain : ACM Press, 2013. – ISBN 978–1–4503–2396–3, 1–6
- [SD14] SIGTIA, Siddharth ; DIXON, Simon: Improved Music Feature Learning with Deep Neural Networks. In: *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Florence, Italy : Institute of Electrical and Electronics Engineers (IEEE), Mai 2014. – ISBN 978–1–4799–2893–4, 6959–6963
- [SE03] SHEH, Alexander ; ELLIS, Daniel P W.: Chord Segmentation and Recognition using EM-Trained Hidden Markov Models. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Baltimore, 2003. – 00231
- [Sea02] SEASHORE, Carl E.: A Voice Tonoscope. In: *Studies in Psychology* 3 (1902), S. 18–28
- [Sea38] SEASHORE, Carl E.: *Psychology of Music*. New York : McGraw-Hill, 1938
- [Ser89] SERRA, Xavier: *A System for Sound Analysis / Transformation / Synthesis Based on a Deterministic plus Stochastic Decomposition*, Stanford University, Dissertation, 1989
- [Set05] SETHARES, William A.: *Tuning, timbre, spectrum, scale*. London : Springer, 2005 <http://public.eblib.com/EBLPublic/PublicView.do?ptiID=303730>. – ISBN 978–1–84628–113–6 1–84628–113–X 1–85233–797–4 978–1–85233–797–1

- [Set07] SETHARES, William A.: *Rhythm and transforms*. Berlin; London : Springer, 2007 <http://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&db=nlabk&AN=255666>. – ISBN 978-1-84628-640-7 1-84628-640-9 978-1-84628-639-1 1-84628-639-5
- [SFM⁺14] SMARAGDIS, Paris ; FEVOTTE, Cedric ; MYSORE, Gautham J. ; MOHAMMADIHA, Nasser ; HOFFMAN, Matthew: Static and Dynamic Source Separation Using Nonnegative Factorizations: A unified view. In: *IEEE Signal Processing Magazine* 31 (2014), Mai, Nr. 3, 66–75. <http://dx.doi.org/10.1109/MSP.2013.2297715>. – DOI 10.1109/MSP.2013.2297715. – ISSN 1053-5888
- [SG84] SMITH, Julius O. ; GOSSET, Phil: A flexible sampling-rate conversion method. In: *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* Bd. 2. San Diego, 1984
- [SG15] SCHLÜTER, Jan ; GRILL, Thomas: Exploring Data Augmentation for Improved Singing Voice Detection with Neural Networks. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Malaga, Spain, 2015, S. 6
- [SGER14] SALAMON, Justin ; GOMEZ, Emilia ; ELLIS, Daniel P W. ; RICHARD, Gael: Melody Extraction from Polyphonic Music Signals: Approaches, Applications, and Challenges. In: *IEEE Signal Processing Magazine* 31 (2014), März, Nr. 2, S. 118–134. <http://dx.doi.org/10.1109/MSP.2013.2271648>. – DOI 10.1109/MSP.2013.2271648. – ISSN 1558-0792. – Conference Name: IEEE Signal Processing Magazine
- [SGU14] SCHEDL, Markus ; GÓMEZ, Emilia ; URBANO, Julián: Music Information Retrieval: Recent Developments and Applications. In: *Foundations and Trends® in Information Retrieval* 8 (2014), September, Nr. 2-3, 127–261. <http://dx.doi.org/10.1561/1500000042>. – DOI 10.1561/1500000042. – ISSN 1554-0669, 1554-0677
- [SH13] SPRING, Glenn ; HUTCHESON, Jere: *Musical Form and Analysis: Time, Pattern, Proportion*. Long Grove, IL : Waveland Press, 2013. – ISBN 978-1-4786-0722-9. – OCLC: 882602291
- [Sha84] SHAFFER, L H.: Timing in Solo and Duet Piano Performances. In: *The Quarterly Journal of Experimental Psychology* 36A (1984), S. 577–595
- [Sha98] SHANNON, Claude E.: Communication In The Presence Of Noise. In: *Proceedings of the IEEE* 86 (1998), Februar, Nr. 2, 447–457. <http://dx.doi.org/10.1109/JPROC.1998.659497>. – DOI 10.1109/JPROC.1998.659497. – ISSN 0018-9219
- [She64] SHEPARD, Roger N.: Circularity in Judgments of Relative Pitch. In: *Journal of the Acoustical Society of America (JASA)* 36 (1964), Nr. 12, 2346. <http://dx.doi.org/10.1121/1.1919362>. – DOI 10.1121/1.1919362. – ISSN 00014966
- [Sie65] SIEGEL, Robert J.: A Replication of the Mel Scale of Pitch. In: *The American Journal of Psychology* 78 (1965), Dezember, Nr. 4, 615. <http://dx.doi.org/10.2307/1420924>. – DOI 10.2307/1420924. – ISSN 00029556. – 00019
- [SJ01] SU, Borching ; JENG, Shyh-Kang: Multi-Timbre Chord Classification using Wavelet Transform and Self-Organized Map Neural Networks. In: *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Ieee, 2001. – ISBN 0-7803-7041-4
- [SJL⁺05] SEO, Jin S. ; JIN, Minho ; LEE, Sunil ; JANG, Dalwon ; LEE, Seungjae ; YOO, Chang D.: Audio Fingerprinting Based on Normalized Spectral Subband Centroids. In: *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (2005), 213–216. <http://dx.doi.org/10.1109/ICASSP.2005.1415684>. – DOI 10.1109/ICASSP.2005.1415684

- [SK01] SHIMAMURA, Tetsuya ; KOBAYASHI, Hajime: Weighted Autocorrelation for Pitch Extraction of Noisy Speech. In: *Transactions on Speech and Audio Processing* 9 (2001), Nr. 7, 727–730. <http://dx.doi.org/10.1109/89.952490>. – DOI 10.1109/89.952490. – ISSN 10636676
- [SK04] SCHONBERG, Daniel ; KIROVSKI, Darko: Fingerprinting and Forensic Analysis of Multimedia. In: *Proceedings of the 12th ACM Multimedia Conference*. New York, 2004
- [SK10] SCHÖRKHUBER, Christian ; KЛАPURI, Anssi P.: Constant-Q Transform Toolbox for Music Processing. In: *Proceedings of the 7th Sound and Music Computing Conference (SMC)*. Barcelona, 2010
- [SL01] SLOBODA, John A. ; LEHMANN, Andreas C.: Tracking Performance Correlates of Changes in Perceived Intensity of Emotion During Different Interpretation of a Chopin Piano Prelude. In: *Music Perception* 19 (2001), Nr. 1, S. 87–120
- [Sla93] SLANEY, Malcolm: An Efficient Implementation of the Patterson-Holdsworth Auditory Filter Bank / Apple Perception Group. 1993. – Forschungsbericht
- [Sla98] SLANEY, Malcolm: Auditory Toolbox – Version 2 / Interval Research Corporation. Wien, 1998 (TR-1998-010). – Forschungsbericht
- [Sla11] SLANEY, Malcolm: Web-Scale Multimedia Analysis: Does Content Matter? In: *IEEE Multimedia* 18 (2011), Februar, Nr. 2, 12–15. <http://dx.doi.org/10.1109/MMUL.2011.34>. – DOI 10.1109/MMUL.2011.34. – ISSN 1070–986X
- [Slo82] SLOBODA, John A.: Music Performance. In: DEUTSCH, Diana (Hrsg.): *The Psychology of Music*. New York : Academic Press, 1982
- [Slo85] SLOBODA, John A.: *The Musical Mind – The cognitive psychology of music*. Oxford : Oxford University Press, 1985 (Oxford Psychology Series 5)
- [SM18] SCHREIBER, Hendrik ; MÜLLER, Meinard: A Crowdsourced Experiment for Tempo Estimation of Electronic Dance Music. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Paris, France, 2018, S. 7
- [SM19] SCHREIBER, Hendrik ; MÜLLER, Meinard: Musical Tempo and Key Estimation using Convolutional Neural Networks with Directional Filters. In: *Proceedings of the Sound and Music Computing Conference (SMC)*. Malaga, Spain, März 2019. – arXiv: 1903.10839
- [Smi99] SMITH, Steven W.: *Digital Signal Processing (The Scientist's and Engineer's Guide to)*. 2nd Editio. San Diego : California Technical Publishing, 1999 <http://linkinghub.elsevier.com/retrieve/doi/10.1006/dspr.1993.1020>. – ISBN 978–0–9660176–3–2
- [Smi07] SMITH, Julius O.: Spectral Audio Signal Processing - March 2007 Draft / Stanford University, Center for Computer Research in Music and Acoustics (CCRMA). California, 2007. – Forschungsbericht
- [SMSB17] SCHRAMM, Rodrigo ; MCLEOD, Andrew ; STEEDMAN, Mark ; BENETOS, Emmanouil: Multi-pitch Detection and Voice Assignment for a cappella Recordings of Multiple Singers. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Suzhou, China, 2017. – Accepted: 2017-08-17T10:13:45Z Publisher: ISMIR
- [SN03] SOULODRE, Gilbert A. ; NORCROSS, Scott G.: Objective Measures of Loudness. In: *Proceedings of the 115th Audio Engineering Society Convention (Preprint No. 5896)*. New York : Audio Engineering Society, 2003

- [Soi04] SOILLE, Pierre: Erosion and Dilation. Version: 2004. https://doi.org/10.1007/978-3-662-05088-0_3. In: *Morphological Image Analysis: Principles and Applications*. Berlin, Heidelberg : Springer, 2004. – ISBN 978–3–662–05088–0, 63–103
- [Sol97] SOLTAU, Hagen: *Erkennung von Musikstilen*. Karlsruhe, Universität Karlsruhe, Diploma Thesis, 1997
- [SP03] SHANNON, Ben J. ; PALIWAL, Kuldip K.: A Comparative Study of Filter Bank Spacing for Speech Recognition. In: *Proceedings of Microelectronic engineering research conference*. Brisbane, November 2003, S. 2–4
- [SRS03] SOULEZ, Ferréol ; RODET, Xavier ; SCHWARZ, Diemo: Improving polyphonic and poly-instrumental music to score alignment. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Baltimore, 2003
- [SRSR08] SMARAGDIS, Paris ; RAJ, Bhiksha ; SMARAGDIS, Paris ; RAJ, Bhiksha: *Shift-Invariant Probabilistic Latent Component Analysis*. 2008
- [SS97] SCHEIRER, Eric D. ; SLANEY, Malcolm: Construction and Evaluation of a Robust Multifeature Speech/Music Discriminator. In: *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Munich, April 1997
- [SSKS04] SHALEV-SHWARTZ, Shai ; KESHET, Joseph ; SINGER, Yoram: Learning to Align Polyphonic Music. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Barcelona, 2004
- [SSWW98] SOLTAU, Hagen ; SCHULZ, Tanja ; WESTPHAL, Martin ; WAIBEL, Alex: Recognition of Music Types. In: *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Seattle, 1998
- [Ste38] STEVENS, Stanley S.: *Hearing - Its Psychology and Physiology*. 1983 repri. New York : American Institute of Physics for the Acoustical Society of America, 1938
- [Ste55] STEVENS, Stanley S.: The Measurement of Loudness. In: *Journal of the Acoustical Society of America (JASA)* 27 (1955), Nr. 5, S. 815–829. – 00392
- [Str97] STRATHERN, Marilyn: ‘Improving ratings’: audit in the British University system. In: *European Review* 5 (1997), Juli, Nr. 3, 305–321. [http://dx.doi.org/10.1002/\(SICI\)1234-981X\(199707\)5:3<305::AID-EURO184>3.0.CO;2-4](http://dx.doi.org/10.1002/(SICI)1234-981X(199707)5:3<305::AID-EURO184>3.0.CO;2-4). – DOI 10.1002/(SICI)1234–981X(199707)5:3<305::AID-EURO184>3.0.CO;2–4. – ISSN 1474–0575, 1062–7987. – Publisher: Cambridge University Press
- [Stu90] STUMPF, Carl: *Tonpsychologie II*. Hilversum / Amsterdam : Knuf and Bonset, 1890
- [Stu12] STURM, Bob L.: An Analysis of the GTZAN Music Genre Dataset. In: *Proceedings of the 2nd International ACM Workshop on Music Information Retrieval with User-Centered and Multimodal Strategies (MIRUM)*, Nara, 2012
- [Stu13] STURM, Bob L.: Classification Accuracy is Not Enough: On the Evaluation of Music Genre Recognition Systems. In: *Journal of Intelligent Information Systems* 41 (2013), Dezember, Nr. 3, 371–406. <http://dx.doi.org/10.1007/s10844-013-0250-y>. – DOI 10.1007/s10844–013–0250–y. – ISSN 1573–7675
- [Stu14] STURM, Bob L.: A Survey of Evaluation in Music Genre Recognition. In: NÜRNBERGER, Andreas (Hrsg.) ; STOBER, Sebastian (Hrsg.) ; LARSEN, Birger (Hrsg.) ; DETYNIECKI, Marcin (Hrsg.): *Adaptive Multimedia Retrieval: Semantics, Context, and Adaptation*. Cham : Springer International Publishing, 2014 (Lecture Notes in Computer Science). – ISBN 978–3–319–12093–5, S. 29–66

- [SV40] STEVENS, Stanley S. ; VOLKMANN, John: The Relation of Pitch to Frequency: A Revised Scale. In: *The American Journal of Psychology* 53 (1940), Juli, Nr. 3, 329. <http://dx.doi.org/10.2307/1417526>. – DOI 10.2307/1417526. – ISSN 00029556
- [SVN37] STEVENS, Stanley S. ; VOLKMANN, John ; NEWMAN, E.B.: A Scale for the Measurement of the Psychological Magnitude Pitch. In: *Journal of the Acoustical Society of America (JASA)* 8 (1937), Nr. 3, 185–190. <http://dx.doi.org/10.1121/1.1915893>. – DOI 10.1121/1.1915893. – ISSN 00014966
- [SW65] SHAPIRO, S. S. ; WILK, M. B.: An Analysis of Variance Test for Normality (Complete Samples). In: *Biometrika* 52 (1965), Nr. 3/4, S. 591–611
- [SW05] SHENOY, Arun ; WANG, Ye: Key, Chord, and Rhythm Tracking of Popular Music Recordings. In: *Computer Music Journal* 29 (2005), September, Nr. 3, 75–86. <http://dx.doi.org/10.1162/0148926054798205>. – DOI 10.1162/0148926054798205. – ISSN 0148–9267
- [SWK08] SEYERLEHNER, Klaus ; WIDMER, Gerhard ; KNEES, Peter: Frame Level Audio Similarity - A Codebook Approach. In: *Proceedings of the International Conference on Digital Audio Effects (DAFX)*. Espoo, Finland, 2008, S. 8
- [SWK10] SEYERLEHNER, Klaus ; WIDMER, Gerhard ; KNEES, Peter: A Comparison of Human, Automatic and Collaborative Music Genre Classification and User Centric Evaluation of Genre Classification Systems. In: *Proceedings of the International Workshop on Adaptive Multimedia Retrieval. Context, Exploration, and Fusion*. Berlin, Heidelberg : Springer, 2010 (Lecture Notes in Computer Science). – ISBN 978–3–642–27169–4, S. 118–131
- [SWLH17] SOUTHALL, Carl ; WU, Chih-Wei ; LERCH, Alexander ; HOCKMAN, Jason A.: MDB Drums — An Annotated Subset of MedleyDB for Automatic Drum Transcription. In: *Late Breaking Demo (Extended Abstract), Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Suzhou : International Society for Music Information Retrieval (ISMIR), 2017
- [SWT04] SCHUBERT, Emery ; WOLFE, Joe ; TARNOPOLSKY, Alex: Spectral centroid and timbre in complex, multiple instrumental textures. In: *Proceedings of the 8th International Conference on Music Perception & Cognition (ICMPC)*. Evanston, August 2004
- [SZ11] SANDEN, Chris ; ZHANG, John Z.: Enhancing Multi-Label Music Genre Classification through Ensemble Techniques. In: *Proceedings of the International Conference on Research and Development in Information Retrieval (SIGIR)*. New York, NY, USA : Association for Computing Machinery, Juli 2011 (SIGIR '11). – ISBN 978–1–4503–0757–4, 705–714
- [SZM06] SCARINGELLA, Nicolas ; ZOIA, Giorgio ; MLYNEK, Daniel: Automatic genre classification of music content: a survey. In: *Signal Processing Magazine* 23 (2006), Nr. 2, S. 133–141
- [Tag03] TAGUTI, Tomoyasu: Mapping a Physical Correlate of Loudness into the Velocity Space of MIDI-Controlled Piano Tones. In: *Proceedings of the Stockholm Music Acoustics Conference (SMAC)*. Stockholm, August 2003
- [TC00] TZANETAKIS, George ; COOK, Perry: MARSYAS: A Framework for Audio Analysis. In: *Organised Sound* 4 (2000), Nr. 3
- [TC02] TZANETAKIS, George ; COOK, Perry: Musical genre classification of audio signals. In: *Transactions on Speech and Audio Processing* 10 (2002), Juli, Nr. 5, 293–302. <http://dx.doi.org/10.1109/TSA.2002.800560>. – DOI 10.1109/TSA.2002.800560. – ISSN 1063–6676
- [TE03] TURETSKY, Robert J. ; ELLIS, Daniel P W.: Ground-Truth Transcriptions of Real Music from Force-Aligned MIDI Syntheses. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Baltimore, 2003

- [TE06] TOIVIAINEN, Petri ; EEROLA, Tuomas: Autocorrelation in meter induction: The role of accent structure. In: *Journal of the Acoustical Society of America (JASA)* 119 (2006), Nr. 2, S. 1164–1170
- [TEC02] TZANETAKIS, George ; ERMOLINSKYI, Andrey ; COOK, Perry: Pitch Histograms in Audio and Symbolic Music Information Retrieval. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Paris, 2002
- [Tem07] TEMPERLEY, David: The Tonal Properties of Pitch-Class Sets : Tonal Implication, Tonal Ambiguity, and Tonalness. In: *Computing in Musicology* 15 (2007), S. 24–38
- [Ter79] TERHARDT, Ernst: Calculating Virtual Pitch. In: *Hearing Research* 1 (1979), S. 155–182
- [Ter92] TERHARDT, Ernst: The SPINC Function for Scaling of Frequency in Auditory Models. In: *Acustica* 77 (1992), S. 40–42
- [TH02] TIMMERS, Renee ; HONING, Henkjan: On music performance, theories, measurement and diversity. In: *Cognitive Processing* 1-2 (2002)
- [Tha89] THAYER, Robert E.: *The Biopsychology of Mood and Arousal*. Oxford University Press, 1989. – ISBN 978-0-19-536175-9
- [THK17] THICKSTUN, John ; HARCHAOUI, Zaid ; KAKADE, Sham M.: Learning Features of Music From Scratch. In: *Proceedings of the International Conference on Learning Representations (ICLR)*. Toulon, France, 2017, S. 14
- [Tho02] THODE, Henry C.: *Testing for Normality*. New York : Marcel Dekker, Inc., 2002. – ISBN 0-8247-9613-6
- [Tim05] TIMMERS, Renee: Predicting the Similarity Between Expressive Performances of Music from Measurements of Tempo and Dynamics. In: *Journal of the Acoustical Society of America (JASA)* 117 (2005), Januar, Nr. 1, 391–399. <http://dx.doi.org/10.1121/1.1835504>. – DOI 10.1121/1.1835504. – ISSN 0001-4966
- [TJM07] TZANETAKIS, George ; JONES, Randy ; McNALLY, Kirk: Stereo panning features for classifying recording production style. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Vienna, 2007, 441–444
- [TK00] TOLONEN, Tero ; KARJALAINEN, Matti: A Computationally Efficient Multipitch Analysis Model. In: *Transactions on Speech and Audio Processing* 8 (2000), Nr. 6, S. 708–716
- [TLPG07] TURNBULL, Douglas ; LANCKRIET, Gert ; PAMPALK, Elias ; GOTO, Masataka: A Supervised Approach for Detecting Boundaries in Music Using Difference Features and Boosting. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Vienna, Austria, 2007, S. 5
- [TMCV06] TIMMERS, Renee ; MAROLT, Matija ; CAMURRI, Antonio ; VOLPE, Gualtiero: Listeners' emotional engagement with performances of a Scriabin étude: an explorative case study. In: *Psychology of Music* 34 (2006), Nr. 4, S. 481–510
- [Tod93] TODD, Neil P M.: Vestibular Feedback in Musical Performance. In: *Music Perception* 10 (1993), Nr. 3, S. 379–382
- [Tra90] TRAUNMÜLLER, Hartmut: Analytical Expressions for the Tonotopic Sensory Scale. In: *Journal of the Acoustical Society of America (JASA)* 88 (1990), Nr. 1, 97–100. <http://dx.doi.org/10.1121/1.399849>. – DOI 10.1121/1.399849. – ISSN 00014966

- [TSS82] TERHARDT, Ernst ; STOLL, Gerhard ; SEEWANN, Manfred: Algorithm for extraction of pitch and pitch salience from complex tonal signals. In: *Journal of the Acoustical Society of America (JASA)* 71 (1982), Nr. 3, 679. <http://dx.doi.org/10.1121/1.387544>. – DOI 10.1121/1.387544. – ISSN 00014966
- [TTKV08] TROHIDIS, Konstantinos ; TSOUMAKAS, Grigoris ; KALLIRIS, George ; VLAHAVAS, Ioannis: Multi-Label Classification of Music into Emotions. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Philadelphia, 2008
- [TW03] THOMPSON, Sam ; WILLIAMON, Aaron: Evaluating Evaluation: Musical Performance Assessment as a Research Tool. In: *Music Perception: An Interdisciplinary Journal* 21 (2003), September, Nr. 1, 21–41. <http://dx.doi.org/10.1525/mp.2003.21.1.21>. – DOI 10.1525/mp.2003.21.1.21. – ISSN 0730-7829, 1533-8312
- [Tza05] TZANETAKIS, George: Tempo Extraction using Beat Histograms. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2005, S. 2–3
- [Tza08] TZANETAKIS, George: MARSYAS-0.2: A Case Study in Implementing Music Information Retrieval Systems. In: SHEN, Jialie (Hrsg.) ; SHEPHERD, John (Hrsg.) ; CUI, Bin (Hrsg.) ; LIU, Ling (Hrsg.): *Intelligent Music Information Systems: Tools and Methodologies*. IGI Global, 2008, S. 31–49
- [UCN99] UMESH, S ; COHEN, L ; NELSON, D: Fitting the Mel scale. In: *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing. (ICASSP99)*. Phoenix : IEEE, 1999. – ISBN 0-7803-5041-3, 217–220 vol.1
- [UH03] UHLE, Christian ; HERRE, Jürgen: Estimation of Tempo, Micro Time and Time Signature from Percussive Music. In: *Proceedings of the 6th International Conference on Digital Audio Effects (DAFX)*. London, 2003
- [UPG⁺17] UHLICH, Stefan ; PORCU, Marcello ; GIRON, Franck ; ENENKL, Michael ; KEMP, Thomas ; TAKAHASHI, Naoya ; MITSUFUJI, Yuki: Improving Music Source Separation based on Deep Neural Networks through Data Augmentation and Network Blending. In: *Proceedings of the International Conference on Acoustics Speech and Signal Processing (ICASSP)*, 2017, S. 261–265
- [Van95] VANTOMME, Jason D.: Score Following by Temporal Pattern. In: *Computer Music Journal* 19 (1995), Nr. 3, S. 50–59. – 00023
- [Ver84] VERCOE, Barry L.: The Synthetic Performer in the Context of Live Performance. In: *Proceedings of the International Computer Music Conference (ICMC)*. Paris, 1984, S. 199–200
- [VGW⁺17] VIDWANS, Amruta ; GURURANI, Siddharth ; WU, Chih-Wei ; SUBRAMANIAN, Vinod ; SWAMINATHAN, Rupak V. ; LERCH, Alexander: Objective Descriptors for the Assessment of Student Music Performances. In: *Proceedings of the AES Conference on Semantic Audio*. Erlangen : Audio Engineering Society (AES), 2017
- [Vig04] VIGNOLI, Fabio: Digital Music Interaction Concepts: A User Study. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Barcelona, 2004
- [VIP03] VAN IMMERSEEL, Luc ; PEETERS, Stefaan: Digital implementation of linear gammatone filters: Comparison of design methods. In: *Acoustics Research Letters Online* 4 (2003), Nr. 3, 59. <http://dx.doi.org/10.1121/1.1573131>. – DOI 10.1121/1.1573131. – ISSN 15297853
- [Vir07] VIRTANEN, Tuomas: Monaural Sound Source Separation by Nonnegative Matrix Factorization With Temporal Continuity and Sparseness Criteria. In: *IEEE Transactions on Audio, Speech, and Language Processing* 15 (2007), März, Nr. 3, S. 1066–1074. <http://dx.doi.org/10.1109/TASL.2006.885253>. – DOI 10.1109/TASL.2006.885253. – ISSN 1558-7924. – Conference Name: IEEE Transactions on Audio, Speech, and Language Processing

- [Vit67] VITERBI, Andrew J.: Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. In: *Transactions on Information Theory* 13 (1967), S. 260–269
- [VP85] VERCOE, Barry L. ; PUCKETTE, Miller S.: Synthetic Rehearsal: Training the Synthetic Performer. In: *Proceedings of the International Computer Music Conference (ICMC)*. Vancouver, 1985, S. 275–278
- [VPE18] VIRTANEN, Tuomas (Hrsg.) ; PLUMBLEY, Mark D. (Hrsg.) ; ELLIS, Dan (Hrsg.): *Computational Analysis of Sound Scenes and Events*. Cham : Springer International Publishing, 2018. <http://dx.doi.org/10.1007/978-3-319-63450-0>. <http://dx.doi.org/10.1007/978-3-319-63450-0>. – ISBN 978-3-319-63449-4 978-3-319-63450-0
- [VPM08] VAREWYCK, Matthias ; PAUWELS, Johan ; MARTENS, Jean-Pierre: A Novel Chroma Representation of Polyphonic Music Based on Multiple Pitch Tracking Techniques. In: *Proceedings of the 16th ACM International Conference on Multimedia (MM)* (2008), 667. <http://dx.doi.org/10.1145/1459359.1459455>. – DOI 10.1145/1459359.1459455
- [VVDB18] VANDE VEIRE, Len ; DE BIE, Tijl: From Raw Audio to a Seamless Mix: Creating an Automated DJ System for Drum and Bass. In: *EURASIP Journal on Audio, Speech, and Music Processing* 2018 (2018), September, Nr. 1, 13. <http://dx.doi.org/10.1186/s13636-018-0134-8>. – DOI 10.1186/s13636-018-0134-8. – ISSN 1687-4722
- [Wal02] WALLS, Peter: Historical performance and the modern performer. In: RINK, John (Hrsg.): *Musical Performance – A Guide to Understanding*. Cambridge : Cambridge University Press, 2002
- [Wan03] WANG, Avery: An Industrial Strength Audio Search Algorithm. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Washington, 2003
- [WB06] WANG, DeLiang ; BROWN, Guy J.: *Computational Auditory Scene Analysis: Principles, Algorithms, and Applications*. Wiley-IEEE Press, 2006 <https://ieeexplore.ieee.org/book/5769523>. – ISBN 978-0-470-04338-7
- [WB10] WEISS, Ron J. ; BELLO, Juan P.: Identifying Repeated Patterns in Music Using Sparse Convulsive Non-Negative Matrix Factorization. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Utrecht, August 2010
- [WBKW96] WOLD, Erling ; BLUM, Thom ; KEISLAR, Douglas ; WHEATEN, James: Content-based classification, search, and retrieval of audio. In: *IEEE Multimedia* 3 (1996), Nr. 3, 27–36. <http://dx.doi.org/10.1109/93.556537>. – DOI 10.1109/93.556537. – ISSN 1070986X
- [WD08] WEIL, Jan ; DURRIEU, Jean-Louis: An HMM-based Audio Chord Detection System Attenuating the Main Melody. In: *Proceedings of the Music Information Retrieval EXchange (MIREX) at the 9th International Conference on Music Information Retrieval (ISMIR)*. Philadelphia, September 2008
- [WE99] WILLIAMS, Gethin ; ELLIS, Daniel P W.: Speech/Music Discrimination Based on Posterior Probability Features. In: *Proceedings of the 6th European Conference on Speech Communication and Technology (EUROSPEECH)*. Budapest, September 1999. – 00105
- [Wel12] WELD, Harry P.: An Experimental Study of Musical Enjoyment. In: *The American Journal of Psychology* 23 (1912), Nr. 2, 245–308. <http://www.jstor.org/stable/1412844>
- [Wel67] WELCH, Peter D.: The Use of Fast Fourier Transform for the Estimation of Power Spectra: A Method based on Time Averaging over Short, Modified Periodograms. In: *IEEE Transactions on Audio and Electroacoustics* 15 (1967), Juni, Nr. 2, S. 70–73. <http://dx.doi.org/10.1109/TAU>.

- [1967.1161901](https://doi.org/10.1109/TAU.1967.1161901). – DOI 10.1109/TAU.1967.1161901. – ISSN 1558–2582. – Conference Name: IEEE Transactions on Audio and Electroacoustics
- [WGL⁺16] WU, Chih-Wei ; GURURANI, Siddharth ; LAGUNA, Christopher ; PATI, Ashis ; VIDWANS, Amruta ; LERCH, Alexander: Towards the Objective Assessment of Music Performances. In: *Proceedings of the International Conference on Music Perception and Cognition (ICMPC)*. San Francisco, 2016. – ISBN 1-879346-65-5, 99–103
- [WHB⁺08] WEGENER, Sebastian ; HALLER, Martin ; BURRED, Juan J. ; SIKORA, Thomas ; ESSID, Slim ; RICHARD, Gaël: On the Robustness of Audio Features for Musical Instrument Classification. In: *Proceedings of the 16th European Signal Processing Conference (EUSIPCO)*. Lausanne, 2008
- [Whi29] WHITTAKER, John M.: The 'Fourier' Theory of the Cardinal Function. In: *Proceedings of the Edinburgh Mathematical Society* 1 (1929), Januar, 169–176. <http://dx.doi.org/10.1017/S0013091500013511>. – DOI 10.1017/S0013091500013511. – ISSN 0013–0915
- [WL15] WU, Chih-Wei ; LERCH, Alexander: Drum Transcription using Partially Fixed Non-Negative Matrix Factorization. In: *Proceedings of the European Signal Processing Conference (EUSIPCO)*. Nice : EURASIP, 2015
- [WL18a] WU, Chih-Wei ; LERCH, Alexander: Assessment of Percussive Music Performances with Feature Learning. In: *International Journal of Semantic Computing* 12 (2018), Nr. 3, 315–333. <http://dx.doi.org/10.1142/S1793351X18400147>. – DOI 10.1142/S1793351X18400147. – ISSN 1793–351X
- [WL18b] WU, Chih-Wei ; LERCH, Alexander: From Labeled to Unlabeled Data – On the Data Challenge in Automatic Drum Transcription. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Paris, 2018
- [WL18c] WU, Chih-Wei ; LERCH, Alexander: Learned Features for the Assessment of Percussive Music Performances. In: *Proceedings of the International Conference on Semantic Computing (ICSC)*. Laguna Hills : IEEE, 2018
- [WR12] WÜLFING, Jan ; RIEDMILLER, Martin: Unsupervised Learning of Local Features for Music Classification. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Porto, Portugal, 2012
- [Wri08] WRIGHT, Matthew J.: *The Shape of an Instant: Measuring and Modeling Perceptual Attack Time with Probability Density Functions*. Stanford, Stanford University, Dissertation, 2008. – 00015
- [WT85] WATSON, David ; TELLEGGEN, Auke: Toward a Consensual Structure of Mood. In: *Psychological Bulletin* 98 (1985), Nr. 2, S. 219–235. <http://dx.doi.org/10.1037/0033-2909.98.2.219>. – DOI 10.1037/0033–2909.98.2.219. – ISSN 1939–1455(Electronic),0033–2909(Print). – Place: US Publisher: American Psychological Association
- [WWE16] WESOŁOWSKI, Brian C. ; WIND, Stefanie A. ; ENGELHARD, George: Examining Rater Precision in Music Performance Assessment: An Analysis of Rating Scale Structure Using the Multifaceted Rasch Partial Credit Model. In: *Music Perception: An Interdisciplinary Journal* 33 (2016), Juni, Nr. 5, 662–678. <http://dx.doi.org/10.1525/mp.2016.33.5.662>. – DOI 10.1525/mp.2016.33.5.662. – ISSN 0730–7829, 1533–8312
- [XBP⁺18] XI, Qingsyang ; BITTNER, Rachel M. ; PAUWELS, Johan ; YE, Xuzhou ; BELLO, Juan P.: GUITARSET: A Dataset for Guitar Transcription. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Paris, France, 2018, S. 8
- [XZY05] XU, Yunpeng ; ZHANG, Changshui ; YANG, Jing: Semi-Supervised Classification of Musical Genre using Multiview Features. In: *Proceedings of the International Computer Music Conference (ICMC)*, 2005

- [Yan01] YANG, Cheng: MACS: Music Audio Characteristic Sequence Indexing for Similarity Retrieval. In: *Proceedings of the Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. New Paltz : IEEE, 2001. – ISBN 0-7803-7126-7, 123–126
- [YC12] YANG, Yi-Hsuan ; CHEN, Homer H.: Machine Recognition of Music Emotion: A Review. In: *ACM Transactions on Intelligent Systems and Technology* 3 (2012), Mai, Nr. 3, 40:1–40:30. <http://dx.doi.org/10.1145/2168752.2168754>. – DOI 10.1145/2168752.2168754. – ISSN 2157-6904
- [YD07] YOU, Wei ; DANNENBERG, Roger: Polyphonic Music Note Onset Detection Using Semi-Supervised Learning. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Vienna, Austria, 2007, S. 279–282
- [YD15] YU, Dong ; DENG, Li: *Automatic Speech Recognition - A Deep Learning Approach*. London : Springer, 2015 (Signals and Communication Technology). <https://doi.org/10.1007/978-1-4471-5779-3>. – ISBN 978-1-4471-5779-3
- [YEH⁺02] YOUNG, Steve ; EVERMANN, Gunnar ; HAIN, Thomas ; KERSHAW, Dan ; MOORE, Gareth ; ODELL, Julian ; OLLASON, Dave ; POVEY, Dan ; VALTCHEV, Valtcho ; WOODLAND, Phil: The HTK Book. 2002 (July 2000). – Forschungsbericht
- [YLC06] YANG, Yi-Hsuan ; LIU, Chia-Chu ; CHEN, Homer H.: Music Emotion Classification: a Fuzzy Approach. In: *Proceedings of the 14th annual ACM International Conference on Multimedia*, ACM, 2006. – ISBN 1-59593-447-2
- [YLSC08] YANG, Yi-Hsuan ; LIN, Yu-Ching ; SU, Ya-Fan ; CHEN, Homer H.: A Regression Approach to Music Emotion Recognition. In: *Transactions on Audio, Speech, and Language Processing* 16 (2008), Februar, Nr. 2, 448–457. <http://dx.doi.org/10.1109/TASL.2007.911513>. – DOI 10.1109/TASL.2007.911513. – ISSN 1558-7916
- [YRB99] YIK, Michelle S M. ; RUSSELL, James A. ; BARRETT, Lisa F.: Structure of Self-Reported Current Affect: Integration and Beyond. In: *Journal of Personality and Social Psychology* 77 (1999), Nr. 3, S. 600–619
- [YSLC07] YANG, Yi-Hsuan ; SU, Ya-fan ; LIN, Yu-ching ; CHEN, Homer H.: Music Emotion Recognition: The Role of Individuality. In: *Proceedings of the International Workshop on Human-centered Multimedia (HCM)*. Augsburg : ACM, 2007. – ISBN 978-1-59593-781-0
- [Zö08] ZÖLZER, Udo: *Digital Audio Signal Processing*. 2nd Edition. Stuttgart : John Wiley & Sons Ltd, 2008. – ISBN 978-0-470-99785-7
- [ZF67] ZWICKER, Eberhard ; FELDTKELLER, Richard: *Das Ohr als Nachrichtenempfänger*. 2. Stuttgart : S. Hirzel Verlag, 1967. – ISBN 978-3-7776-0104-5
- [ZF99] ZWICKER, Eberhard ; FASTL, Hugo: *Psychoacoustics. Facts and Models*. 2nd Editio. Springer, 1999. – ISBN 978-3-540-23159-2. – 03460
- [ZGS08] ZENTNER, Marcel ; GRANDJEAN, Didier ; SCHERER, Klaus R.: Emotions Evoked by the Sound of Music: Characterization, Classification, and Measurement. In: *Emotion* 8 (2008), August, Nr. 4, 494–521. <http://dx.doi.org/10.1037/1528-3542.8.4.494>. – DOI 10.1037/1528-3542.8.4.494. – ISSN 1528-3542
- [Zha98] ZHANG, Tong: Hierarchical system for content-based audio classification and retrieval. In: *Proceedings of SPIE* (1998), 398–409. <http://dx.doi.org/10.1117/12.325832>. – DOI 10.1117/12.325832. – ISSN 0277786X
- [ZKG05] ZHU, Yongweil ; KANKANHALLI, Mohan S. ; GAO, Sheng: Music Key Detection for Musical Audio. In: *Proceedings of the 11th International Multimedia Modelling Conference*. Melbourne, 2005

- [ZL15] ZHOU, Xinquan ; LERCH, Alexander: Chord Detection Using Deep Learning. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Malaga : ISMIR, 2015
- [ZR07] ZHOU, Ruohua ; REISS, Joshua D.: Music Onset Detection Combining Energy-Based and Pitch-Based Approaches. In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Wien, September 2007
- [Zwi80] ZWICKER, Eberhard: Analytical Expressions for Critical-Band Rate and Critical Bandwidth as a Function of Frequency. In: *Journal of the Acoustical Society of America (JASA)* 68 (1980), Nr. 5, 1523–1525. <http://dx.doi.org/10.1121/1.385079>. – DOI 10.1121/1.385079. – ISSN 00014966. – 00518

Index

- (Mel) Spectrogram, 77
Zero-R-Classifier, 97
- A Weighting, 158
ACA, 25, 26, 31, 33, 35, 50, 76, 78, 91, 93, 129, 132, 134, 136, 145, 149, 160, 164, 174, 179, 180, 187, 207, 209, 215, 221, 227, 261–264
Accuracy, 99, 130, 144, 171, 176, 211, 218
ACF, 26, 71, 72, 116–119, 122–124, 126, 152, 173–175, 177, 178, 238, 240
ACF Maximum, 71, 72
Acoustic Onset Time, *see* AOT
Adaptive Differential Pulse Code Modulation, *see* AD-PCM
Aeolic Mode, 134
Aliasing, 45, 229, 247
Alignment Path, 188, 189, 193, 194, 196
Alternative Blackman Window, 249
AMDF, 118, 119
AMDF-Weighted Autocorrelation Function, 118
Angular Frequency, 36
ANN, 210, 220
AOT, 164
API, 264
Application Programmer’s Interface, *see* API
Area Under Curve, *see* AUC
Arithmetic Mean, 37–40, 44, 67, 68, 73, 77, 132, 146, 172–174, 257
Articulation, 167, 216
Artificial Neural Network, *see* ANN
Attack, 164
Attack Time, 54, 159, 163, 164, 220
aubio, 264
AUC, 99, 100
Audio Content, 31
Audio Content Analysis, *see* ACA
Audio Fingerprinting, 206
Audio Signal, 35
 Periodic, 35
 Random, 37
Audio-to-Audio Alignment, 187, 193–195, 197
Audio-to-Score Alignment, 187, 194–196
Auditory Filterbank, 50, 51
Autocorrelation Function, *see* ACF
- Average Magnitude Difference Function, *see* AMDF
Bar, 139, 167, 177, 178
Bar Length, 177, 178
Bark, 56, 105, 106, 161
Bark Scale, 106, 107
Bartlett Window, 248
Beat, 139, 146, 165–167, 174–177, 180, 240
Beat Detection, 177
Beat Grid, 174–176
Beat Histogram, 52, 172–175, 209
Beat Period, 174
Beat Phase, 174–177
Beat Spectrum, 172
Beat Strength, 173
Beat Tracking, 175–177
Beats per Minute, *see* BPM
Blackman Window, 249
Blackman-Harris Window, 249
Block Length, 42, 43, 46, 53, 113, 116, 157, 158, 161, 168, 203, 238, 251
Block Overlap Ratio, 42, 43, 161, 169, 203
Block-Based Processing, 42
Box-Cox Transform, 76
BPM, 165, 167, 174
BS.1770, 158, 161
BS.468, 158
- C Weighting, 158
C++ Framework for Audio and Music, *see* CLAM
CAMEL, 263
CASA, 26
CCF, 140, 143, 175–178, 237, 238, 240
CCIR Weighting, 158
CD Quality, 32
cent, 110
Center Clipping, 118
Center of Gravity, *see* COG
Central Moment, 37, 58
Centroid, 141, 220
Cepstrum, 63, 122, 123
Cepstrum-based, 118
Chord, 107, 144–146, 148, 150
Chord Detection, 150

- Chord Inversion, 144
Chord Progression, 148
Chord Recognition, 69, 132, 139, 144–147, 149, 150
Chord Template, 146
Chord Type, 150
Chorus, 178
Chroma Perception, 107
Chromatic Mode, 134
CiCF, 238, 250
Circle of Fifths, 135, 140, 142, 143, 148, 149
Circular Correlation Function, *see* CiCF
CLAM, 262
Classification, 63, 78, 79, 207, 209, 210, 218–220
Clustering, 88
CNN, 77, 87, 168, 210
Coefficient of Determination, 100, 218
COG, 55, 169
Comb Filter, 126
CoMIRVA, 262, 263
Comité Consultatif International des Radiocommunications, *see* CCIR
Compact Disc, *see* CD
Composition, 31, 221
Computational Auditory Scene Analysis, *see* CASA
Computer Audition, 26
Concert Pitch, 131
Confusion Matrix, 98
Constant *Q* Transform, *see* CQT
Content-based Audio and Music Extraction Library, *see* CAMEL
Convolution, 122, 231–233, 240, 242, 243, 247, 248, 250
 Associativity, 232
 Circularity, 233
 Commutativity, 231
 Distributivity, 232
 Identity, 231
Convolutional Neural Network, *see* CNN
Correlation, 79, 148, 165, 203
Correlation Coefficient, 146
Correlation Function, 71, 116, 118, 122, 237, 238
Cosine Distance, 90, 143, 169, 173, 194, 214
Cosine Similarity, 90
Cosine Window, 249, 250
Cost Matrix, 189–192
Covariance Matrix, 255
CQT, 47, 50, 133, 139
Critical Band, 106, 169
Critical Band Rate, 56, 105–107
Cross Correlation Function, *see* CCF
Cross Validation, 92
Cut-Off Frequency, 126, 234, 235
Data, 91
Data Augmentation, 93
Davies-Bouldin-Index, 101
DC Removal, 44
DCT, 63
Decibel, 63, 155, 156, 230
Decision Tree, 86
Deep Neural Network, *see* DNN
Delta Function, 231, 245, 246, 248
Delta Pulse, 126, 175, 176, 178, 246, 247
Descriptor, *see* Feature
DFT, 42, 46, 47, 63, 233, 241, 250, 251, 253
Diatonic Scale, 108
Diatonic Temperament, 110
Differentiator, 44
Digital Signal Processing, *see* DSP
Dimensionality Reduction, 78, 81
Discrete Cosine Transform, *see* DCT
Discrete Fourier Transform, *see* DFT
Disk Jockey, *see* DJ
Distance Matrix, 187–194
Distance Measure, 88, 90
DJ, 27, 174, 179, 213
DNN, 50, 78, 86, 87, 92, 129, 141, 145, 168, 176
Dominant, 145
Dominant Chord, 145
Dorian Mode, 134
Down-Mixing, 43, 44, 115, 202
Down-Sampling, 45, 193, 202
Downbeat, 167, 176–178
Downbeat Detection, 179
DP, 176, 189, 195, 196
DSP, 26
DTW, 149, 176, 187, 188, 191–196
Dynamic Programming, *see* DP
Dynamic Time Warping, *see* DTW, 149
Dynamics, 31, 156, 194, 221
EM, 86
Emotion, 215, 216
Enharmonic Equivalence, 108–110, 135, 143
Envelope, 53, 159, 160, 168, 173, 194, 202, 240
Envelope Extraction, 32
Epoch, 207
Equal Temperament, 109, 110, 133, 139
Equal-Loudness Contour, 158
Equivalent Rectangular Bandwidth, *see* ERB
ERB, 107
Euclidean Distance, 66, 75, 84, 90, 127, 143, 194, 203, 214, 218
European Broadcasting Union, *see* EBU
Expectation Maximization, *see* EM

- F-Measure, 99, 130, 171, 176, 177, 184, 211, 218
 False Negative, *see FN*
 False Negative Rate, *see FNR*
 False Positive, *see FP*
 False Positive Rate, *see FPR*
 Fast Fourier Transform, *see FFT*
 FEAPI, 264
 Feature, 26, 32, 33, 50, 52–54, 63, 71, 72, 75–80, 140, 141, 147, 155, 157, 168, 173, 174, 178, 180, 181, 194–196, 202, 203, 207, 209, 210, 213, 214, 216, 219, 220, 223, 255, 261–265
 Feature Aggregation, 77
 Feature Extraction Application Programmer’s Interface, *see FEAPI*
 Feature Learning, 73, 74, 94, 95
 Feature Matrix, 75, 77
 Feature Representation, 33
 Feature Scaling, 75, 76
 Feature Space, 88, 90, 214
 Feature Space Transformation, 78, 80, 257
 Feature Subset Selection, 78, 257
 FFT, 46, 47, 240, 251
 Filter, 233
 Fingerprint, 201–204
 Fingerprinting, 201–203
 Finite Impulse Response, *see FIR*
 FIR, 126, 233, 235, 242
 Floating Point, 230
 FN, 98, 99, 171, 172, 202
 FNR, 99, 206
 Fourier Series, 36, 111, 241, 250
 Fourier Transform, *see FT*
 FP, 98, 99, 170–172, 184, 202
 FPR, 99, 171, 172, 206
 FT, 46, 48, 63, 125, 237, 240, 241, 244–246, 248, 250
 Convolution, 242
 Frequency Scaling, 244
 Frequency Shift, 243
 Multiplication, 242
 Parseval’s Theorem, 243
 Superposition, 242
 Symmetry, 244
 Time Scaling, 244
 Time Shift, 243
 Full Scale, 155, 157, 160, 230
 Full-Wave Rectification, *see FWR*
 Fundamental Frequency, 26, 35–37, 55, 58, 73, 105, 111, 112, 114, 115, 120, 122–126, 129, 139, 195, 209, 220, 241
 Fundamental Frequency Detection, 73, 111, 112, 115, 125, 252
 FWR, 173
 Gammatone Filter, 51
 Gammatone Filterbank, 51, 124
 Gaussian Distribution, 37
 Gaussian Mixture Model, *see GMM*
 Gaussianity, 58, 76
 Geometric Mean, 39, 68
 Gibbs’ Phenomenon, 36
 GMM, 86, 97, 210, 214, 220
 Ground Truth, *see GT*, 91, 99, 130, 134, 149, 171, 184, 196, 197, 219, 261, 262
 ground truth, 130, 131
 GT, 99
 Half-Wave Rectification, *see HWR*
 Hamming Window, 249
 Harmonic Mean, 39
 Harmonic Mixing, 136
 Harmonic Mode, 134
 Harmonic Product Spectrum, *see HPS*
 Harmonic Sum Spectrum, *see HSS*
 Harmonics, 35, 54, 56, 58, 61, 67, 105, 111, 112, 115, 120, 122, 124, 126, 139, 140, 220
 Harmony, 105, 115, 144, 178, 209
 HFC, 56
 Hidden Markov Model, *see HMM*
 High Frequency Content, *see HFC*
 High-pass Filter, 75
 Hit Rate, 206
 HMM, 86, 146, 147, 195, 196
 HMM Toolkit, *see HTK*
 Homophony, 115
 Hop Size, 42, 43, 157, 203, 253
 HPS, 120–122, 126, 151
 HSS, 122, 126, 151
 HTK, 63, 265
 HWR, 67, 124, 126, 168, 169, 194
 IBI, 165, 174, 176
 ICA, 81
 IDFT, 250
 IFT, 241, 244, 246
 IIR, 75, 233, 235
 Impulse Response, 51, 126, 231, 233–235, 242
 Independent Component Analysis, *see ICA*
 Inference, 83
 Infinite Impulse Response, *see IIR*
 Initial Transient Time, 163
 Input Representation, 33, 168
 Input/Output, *see IO*
 Instantaneous Feature, 43, 52, 53, 136, 157, 209

- Instantaneous Frequency, 71, 114, 133, 169, 252, 253
Instrument Recognition, 219, 220
Instrumentation, 207
Integration Time, 157–159
Intensity, 155–157, 209
Inter-Beat Interval, *see* IBI
Inter-Onset Interval, *see* IOI
International Organization for Standardization, *see* ISO
International Society for Music Information Retrieval,
see ISMIR
International Telecommunication Union, *see* ITU
Interval, 109, 110, 136, 141, 144
Intonation, 31, 111
Inverse Discrete Fourier Transform, *see* IDFT
Inverse Fourier Transform, *see* IFT
Inversion, 150
IOI, 163–165, 175, 177
- jMIR, 262
JNDL, 155
Just Noticeable Difference in Level, *see* JNDL
- K-Means Clustering, 88, 214
K-Nearest Neighbor, *see* KNN
Key, 67, 105, 107, 110, 134–136, 140–145, 148, 213
Key Detection, 32, 69, 132, 134, 139, 141, 144, 146
Key Profile, 141–143, 148
Key Signature, 135
KNN, 88, 220
kNN, 84
Kullback-Leibler Divergence, 90
Kullback-Leibler Divergence, 127
Kurtosis, 58, 76, 174
- L1 Distance, 90
L2 Distance, 90
Laplace Distribution, 37
Latency, 42, 193, 240
LDA, 81, 194
Leave-one-out Cross Validation, 92
Leave-One-Out Crossvalidation, 80
Leptokurtic, 58
libXtract, 263
Linear Discriminant Analysis, *see* LDA
Linear Prediction, 63
Linear Regression, 87
Local Tempo, 165, 167
Log-Mel Spectrogram, 50
Log-mel Spectrogram, 52
Lokrian Mode, 134
Loss Function, 127
- Loudness, 45, 54, 56, 136, 155–158, 160, 161, 196, 209, 214, 215
Loudness Range, 161, 209
Low-Level Feature, 52, 53, 72, 214, 263, 264
Low-pass Filter, 75
Lydian Mode, 134
Lyrics, 207
- MA, 44, 68, 75, 157, 169, 170, 233–235
Maaate, 264
Machine Learning, *see* ML, 207, 261, 262, 264, 265
Machine Listening, 26
macro accuracy, 99
MAE, 100, 131, 172, 218
Magnitude Spectrogram, 137, 138
Magnitude Spectrum, 46, 55, 56, 58, 60, 61, 63, 66–69, 71, 114, 120, 122, 123, 139, 172, 173, 203, 240, 241, 243, 244
Main Lobe, 71, 248, 251
Main Tempo, 165
Major Mode, 108, 134, 135, 141–143
Manhattan Distance, 66, 90, 203, 204
Marsyas, 262
Mean Absolute Error, *see* MAE
Mean Squared Error, *see* MSE
Mean Tempo, 165
Meantone Temperament, 110
Median, 38, 40, 170
Median Filter, 170
Mel, 63, 106
Mel Frequency Cepstral Coefficient, *see* MFCC
Mel Scale, 50, 63, 105–107
Mel Spectrogram, 50
Mel-Spectrogram, 180, 181
Mesokurtic, 58
Meta Data, 25
Meter, 165–167, 176–178
MFCC, 63, 77, 97, 180, 181, 202, 214, 220
Mid-Level Feature, 214
MIDI, 26, 93, 109, 110, 147, 156, 164, 175, 194–197, 261, 262
Min-Max Normalization, 76
Minor Mode, 134, 141–143
MIR, 26, 32, 98, 207, 210, 262, 263
MIRtoolbox, 263
Mixolydian Mode, 134
ML, 83, 98
Mode, 134, 135, 142
Mode Tempo, 165
Modulation, 134, 135, 141
Monophonic, 115
Mood, 215, 216, 218, 219

- Mood Classification, 87, 262
 Mood Recognition, 215, 216, 218
 Motion Picture Experts Group, *see* MPEG
 Moving Average, *see* MA
 MP3, 204
 MPA, 222, 223
 MPEG-1 Layer 3, *see* MP3
 MSE, 87, 98, 100, 131, 172
 Multi-Pitch Detection, 125, 126, 140
 Multi-task Learning, 94
 Music Emotion Recognition, 215
 Music Genre, 32, 213
 Music Genre Classification, 210, 211
 Music Informatics, 26
 Music Information Retrieval, *see* MIR
 Music Information Retrieval Evaluation eXchange, *see* MIREX
 Music Performance, 166, 194, 195, 201, 216, 221–223
 Music Performance Analysis, *see* MPA, 222
 Music Performance Assessment, 221, 223
 Music Similarity, 213, 214, 216, 220
 Musical Communication, 222
 Musical Dynamics, 156
 Musical Form, 166, 207
 Musical Genre, 207, 209, 210, 213, 215
 musical genre, 211
 Musical Genre Classification, 77, 207, 209, 210, 214, 216, 220, 262
 Musical Instrument Digital Interface, *see* MIDI
 Musical Key, 144
 Musical Score, 223
 Nearest Neighbor, *see* NN, 80, 84
 NMF, 126–129, 151
 NN, 97
 Noisiness, 68, 73
 Non-negative Matrix Factorization, *see* NMF
 Normalization, 45, 63, 66, 76, 237, 238
 NOT, 164
 Note, 167, 178, 219, 221
 Note Onset Time, *see* NOT
 Note Value, 167
 Novelty, 180, 182
 Novelty Function, 168–170, 172, 173, 175–178, 180
 Observation, 40, 75
 Octave, 108
 Offset Time, 167
 Onset, 163, 164, 168–172, 174–178, 195–197, 264
 Onset Detection, 66, 84, 164, 165, 168–172, 180
 Onset Time, 163, 164, 167, 168, 170–172, 175, 194–197
 OpenSMILE, 264
 Outlier, 40
 Overfitting, 78, 97
 Overshoot, 36
 Overtone, 111
 P-score, 177
 PAT, 164
 PCA, 80, 81, 255–257
 PDF, 37, 38, 40, 41, 58, 195, 229, 230
 Peak Envelope, 87, 255
 Peak Picking, 168–170, 175
 Peak Program Meter, *see* PPM
 Peak Structure Distance, *see* PSD
 Pearson Correlation Coefficient, 90, 238
 Perceived Tempo, 165
 Perceptual Attack Time, *see* PAT
 Perceptual Onset Time, *see* POT
 Period, 178
 Period Length, 35, 112, 115, 116, 123, 124, 126, 166, 241
 Phase, 114
 Phase Spectrum, 241, 243, 244, 252
 Phrase, 178
 Phrygian Mode, 134
 Pitch, 26, 54, 105, 107, 109–111, 124, 130, 132, 142, 176, 202, 221
 Pitch Chroma, 77, 136–143, 145–149, 178–181, 194, 196
 Pitch Class, 107, 108, 110, 134–137, 140
 Pitch Detection, 111, 150
 Pitch Height, 106, 107, 131
 Pitch Histogram, 209
 Pitch Tracking, 111, 130, 132
 Platykurtic, 58
 Polyphonic, 115, 125, 209, 219
 POT, 164
 Power Spectrum, 55, 56, 60, 63, 68, 118, 241
 PPM, 159, 160
 Pre-dominant Melody Extraction, 125
 Pre-Whitening, 116, 126
 Precision, 99, 100, 170, 171, 211, 218
 Principal Component Analysis, *see* PCA
 Probability Density Function, *see* PDF
 Probe Tone Ratings, 142, 143
 Process Loss, 251
 Production, 31
 PSD, 195
 Pythagorean Temperament, 110
 Quantile, 40
 Quantile Range, 40, 41
 Quantization, 227, 229, 230
 Quantization Error, 229–231

- Quantization Step Size, 229
Quartile, 41
- Radial Basis Function, *see* RBF
Random Forest, 86, 97
Raw Chroma Accuracy, 130
Raw Pitch Accuracy, 130
Real Time, 45
Real-Time, 42, 195, 261, 262, 264
Recall, 99, 100, 170, 171, 211, 218
Receiver Operating Curve, *see* ROC
Rectangular Window, 246, 248–250
Rectilinear Distance, 90
Recurrent Neural Network, *see* RNN
Regression, 87
Relative Frequency Distribution, *see* RFD
Release Time, 159, 160
Rest, 167
Revised Low Frequency B Curve, *see* RLB
RFD, 38
Rhythm, 163, 166, 169, 207, 214, 216
RLB Weighting, 158, 161
RMS, 40, 45, 85, 87, 152, 156–158, 160, 161, 168, 180, 181, 209, 220, 238
RNN, 168
ROC, 99, 100, 171, 172
Root Mean Square, *see* RMS
Root Note, 108, 144, 145, 150
- Sample Rate, 45, 46, 71, 112, 113, 115, 169, 202, 203, 227, 229, 247
Sample Rate Conversion, 45, 115
Sampling, 227
Sampling Theorem, 45, 227, 247
Scale Degree, 107, 110, 111, 134, 145
SCMIR, 263, 264
Score, 26, 31, 108, 109, 111, 133, 135, 166, 167, 194–196, 215, 216, 221–223
Score Following, 194, 195
Selectivity, 99
Self Similarity Matrix, *see* SSM, 52
Self-Organizing Map, *see* SOM
Sensitivity, 99
Sentence, 178
Sequential Backward Elimination, 80
Sequential Forward Selection, 79, 80
Sharpness, 56
Short Time Fourier Transform, *see* STFT, 46
Side Lobe, 248, 251
Signal-to-Noise Ratio, *see* SNR
Silhouette Coefficient, 101
Silhouettes Coefficient, 101
- SIMD, 42
Similarity Matrix, 173, 178, 190
Single Instruction Multiple Data, *see* SIMD
Single Variable Classification, 79
Single-Pole Filter, 157, 235
Singular Value Decomposition, *see* SVD
Skewness, 58, 76, 174
Smoothing, 169, 234, 235
SNR, 230
Software Developer’s Kit, *see* SDK
SOM, 214
Sonic Visualiser, 263
Sparse Coding, 74
Specificity, 99
Spectral Centroid, 43, 55, 56, 58, 59, 87, 203, 209, 255
Spectral Crest Factor, 67, 203
Spectral Decrease, 60, 61
Spectral Envelope, 60, 63, 116, 118, 126, 214
Spectral Flatness, 68, 69, 202
Spectral Flux, 63, 66, 169, 194
Spectral Kurtosis, 59
Spectral Leakage, 248, 251
Spectral Rolloff, 59, 60, 255
Spectral Skewness, 58
Spectral Slope, 61
Spectral Spread, 56, 58, 59, 255
Spectrogram, 46, 125, 127, 129
Spectrum, 45, 63, 68, 69, 120, 122, 123, 125, 126, 241, 243, 245–248, 250
SSM, 179–182, 188
Standard Deviation, 39, 40, 67, 76, 77, 131, 132, 141, 164, 165, 172, 174
Standardization, 76
STFT, 46, 47, 49, 50, 55, 59, 60, 66, 69, 71, 113, 114, 121–123, 126, 133, 136, 137, 139, 168, 169, 173, 194, 203, 204, 247, 250, 252, 253
Structure, 178
Structure Detection, 177, 179, 184
Subdominant Chord, 145
Subfeature, 77, 210, 263
Support Vector Machine, *see* SVM
Support Vector Regression, *see* SVR
SVD, 81
SVM, 86, 87, 97, 210, 220
SVR, 87
- Tactus, 165, 173, 178
Tatum, 165, 166, 177, 178
Temperament, 110, 111
temperament, 131
Tempo, 31, 43, 163, 165–167, 169, 174–178, 193–197, 202, 209, 213, 215, 216, 264

- Tempo Detection, 32, 168, 175–177
 Test Data, 97
 Test Set, 76, 92
 Testing Data, 92
 Texture Window, 43, 77, 141, 173, 210
 Timbre, 31, 54–56, 63, 115, 131, 136, 156, 196, 207, 209, 213, 214
 Timbre Toolbox, 263
 Time Signature, 165–167, 177, 178, 209
 Timing, 31, 164, 166, 193, 194
 TN, 98, 171
 TNR, 99
 Tonal Centroid, 143
 Tonal Power Ratio, 69
 Tonality, 67
 Tonalness, 67–69, 71, 139, 143
 Tonic, 110, 134, 135, 141
 Tonic Chord, 145, 147
 TP, 98, 172, 184, 202
 TPR, 99, 100, 171, 172, 206
 Training Data, 91, 92
 Training Set, 76, 92
 Transfer Learning, 74, 94
 Transient, 36, 56, 68, 157, 164
 Tremolo, 31, 156
 tremolo, 172
 Triad, 144, 146
 True Negative, *see* TN
 True Negative Rate, *see* TNR
 True Peak Meter, 161
 True Positive, *see* TP
 True Positive Rate, *see* TPR
 Tuning Frequency, 110, 131–134, 137
 Tuning Frequency Estimation, 131, 133, 134
 Validation Data, 92
 VAMP, 264
 Variance, 39, 40, 80, 81, 124, 196, 255, 257
 Velocity, 156
 Verse, 178
 Vibrato, 31, 111, 156
 vibrato, 172
 Viterbi, 86
 Viterbi Algorithm, 146–149, 176, 187
 Vocal Activity, 131
 von-Hann Window, 249
 Waikato Environment for Knowledge Analysis, *see*
 WEKA
 Watermarking, 201
 WEKA, 265
 Wholotone Mode, 134
 Wiener-Khinchin Theorem, 240
 Window, 248
 Window Function, 234, 246–248, 250, 251
 Word Length, 203, 229–232
 Workload, 42, 46, 192, 204, 240, 261
 YAAFE, 263
 Yet Another Audio Feature Extractor, *see* YAAFE
 Z-score Normalization, 76
 Zero Crossing Rate, 72, 73, 115, 195, 209, 255
 Zero Phase Filtering, 235
 Zero-Padding, 114