Computational Analysis of Ottoman-Turkish Makam Music for Automatic Content Description and Music Discovery

Sertan Şentürk

TESI DOCTORAL UPF / 2016

Director de la tesi

Dr. Xavier Serra Casals
Information and Communication Technologies

The court's PhD was appointed by the rector of Universitat Pompeu Fabra on ....., 2016.

Chairman

Member

Member

Member

Secretary

The doctoral defense was held on 01 Sep 2016, at the Universitat Pompeu Fabra and scored as .....

PRESIDENT MEMBERS

SECRETARY

To the rhythms in the natural world, from which the rhythms in music emerge...

Abstract

150 words....

Indian art music

Bayesian approaches

Resum

150 palabras.

Resumen

150 palabras.

Preface

- Two or three pages.
- Try to avoid references...

Some of results presented in the dissertation have been published in the following articles:

Acknowledgements

Thank CompMusic

CompMusic partners.

Help with datasets.

Help with code.

Help with music concepts.

Help with conferences.

Contents

List of Symbols

The following is a list of different symbols used in the dissertation along with a short decscription of each symbol.

Symbol	Description
$\{.\}$	Set
$[.]$	Sequence
$< . >$	Tuple
x	An arbitrary "object." Can be a document (a score or an audio recording); an element (a section, phrase, measure, note etc.) in a document; a feature extracted from another document; a model; or also a (sub)set or (sub)sequence of objects.
$\|\boldsymbol{x}\|$	The number of elements in the set or the subsequence (x)
$t(x)$	Time interval or timestamp (for instantaneous eventsXX) of the object (x)
$t_{ini}(x)$	Initial timestamp of the object (x)
$t_{fin}(x)$	Final timestamp of the object (x)
$d(x)$	Duration of the object (x)
$\bar{f}^{(x)}$	An arbitrary fragment selected from a music score or audio recording, e.g. the first instance of the Teslim section in a score, a fragment between 50^{th} and 70^{th} seconds of an audio recording
$\boldsymbol{\mathcal{A}}^{(\boldsymbol{\mathcal{R}})}$	The subjected attribute set (makams, forms or usuls) of the a collection $\boldsymbol{\mathcal{R}}$
$\alpha^{(\boldsymbol{\mathcal{R}})}$	An attribute (makam, form or usul) in the collection $\boldsymbol{\mathcal{R}}$

Symbol	Description
$\mathscr{O}\left(\boldsymbol{\mathcal{A}}^{(\boldsymbol{\mathcal{R}})}\right)$	The overlap of the set of the subjected attribute set $\boldsymbol{\mathcal{A}}^{(\textbf{SymbTr})}$ of the **SymbTr** collection with respect to the set of the subjected attribute set $\boldsymbol{\mathcal{A}}^{(\boldsymbol{\mathcal{R}})}$ of the reference collection $\boldsymbol{\mathcal{R}}$
$o\left(\boldsymbol{\mathcal{A}}_k^{(\boldsymbol{\mathcal{R}})}\right)$	The occurence count of the attribute $\alpha^{(\boldsymbol{\mathcal{R}})} \in \boldsymbol{\mathcal{A}}^{(\boldsymbol{\mathcal{R}})}$
$\hat{o}\left(\boldsymbol{\mathcal{A}}_k^{(\boldsymbol{\mathcal{R}})}\right)$	The occurence count ratio of the attribute XX
$\hat{O}\left(\boldsymbol{\mathcal{A}}_k^{(\boldsymbol{\mathcal{R}})}\right)$	The occurence count of the attribute XX
$\mathscr{O}_k\left(\boldsymbol{\mathcal{A}}^{(\boldsymbol{\mathcal{R}})}\right)$	The overlap against k XX
$\mathscr{C}(\boldsymbol{\mathcal{A}}^{(\boldsymbol{\mathcal{R}})})$	Attribute coverage of the collection $\boldsymbol{\mathcal{R}}$ by **SymbTr** collection
$\mathbf{N}^{(x)}$	Note symbol sequence in the object (x)
$\mathbf{S}^{(\mathbf{x})}$	Section symbol sequence in the object (x)
$n_i^{(x)}$	i^{th} note symbol in the note symbol sequence $\mathbf{N}^{(x)}$
$s_i^{(x)}$	i^{th} section symbol in the note symbol sequence $\mathbf{S}^{(\mathbf{x})}$
$\bar{\mathbf{N}}^{(x)}$	Note sequence in the object (x)
$\bar{\mathbf{M}}^{(x)}$	Measure sequence in the object (x)
$\bar{\mathbf{S}}^{(x)}$	Section sequence in the object (x)
$\bar{\mathbf{P}}^{(x)}$	Phrase sequence in the object (x)
$\bar{n}_i^{(x)}$	i^{th} note in the object (x)
$\bar{m}_i^{(x)}$	i^{th} measure in the object (x)
$\bar{s}_i^{(x)}$	i^{th} section in the object (x)
$\bar{p}_i^{(x)}$	i^{th} phrase in the object (x)
$\mathcal{S}_s{}^{(x)}$	Set of section symbols in the object (x). Equals to $\{\mathbf{S}^{(\mathbf{x})}\}$
$\mathcal{S}^{(x)}$	Set of section symbols in the object (x) plus an $unrelated$ element
$\boldsymbol{\lambda}^{(x)}$	The lyrics (sequence) associated with the object (x)
$\eta\,(x)$	Nodes of an arbitrary graph (x)
$\boldsymbol{\mathcal{U}}$	The set of unique cliques

Symbol	Description
$\mathcal{V}$	The set of similar cliques
$\mathcal{W}$	The set of all intersections between different similar cliques
u	A unique clique
v	A similar clique
w	An intersection between different similar cliques
$\mathscr{L}(\mathbf{x}, \mathbf{y})$	Levenshtein distance between two strings **x** and **y**
$\hat{\mathscr{L}}(\mathbf{x}, \mathbf{y})$	Normalized Levenshtein distance between two strings **x** and **y**
l	Similarity threshold used in symbolic analysis XX
$\Lambda_{mel}^{(x)}$	Semiotic melody label of an arbitrary clique or structural element (x) in a music score.
$\Lambda_{lyr}^{(x)}$	Semiotic lyrics label of an arbitrary clique or structural element (x) in a music score.
$\hat{\boldsymbol{\Psi}}^{(b)}$	Synthetic melody computed from a score fragment (b)
$\hat{\psi}_i^{(b)}$	i^{th} pitch sample in the synthetic melody $\hat{\boldsymbol{\Psi}}^{(b)}$computed from the score fragment (b)
$\hat{\boldsymbol{\Omega}}^{(x)}$	Synthetic harmonic pitch class profiles (HPCPs) (Gómez, 2006) computed from the score fragment (x) in MIDI format
$\hat{\omega}_i^{(x)}$	i^{th} bin of the synthetic HPCPs computed from the score fragment (x) in MIDI format
$\boldsymbol{\varrho}^{(a)}$	Predominant melody extracted from an audio fragment (a)
$\rho_i^{(a)}$	i^{th} pitch sample of the predominant melody $\boldsymbol{\varrho}^{(a)}$extracted from the audio fragment (a)
$\hat{\boldsymbol{\varrho}}^{r,(a)}$	Predominant melody extracted from an audio fragment (a) and normalized with respect to the reference frequency r
$\hat{\rho}_i^{r,(a)}$	i^{th} pitch sample of the predominant melody $\hat{\boldsymbol{\varrho}}^{r,(a)}$, extracted from the audio fragment (a) and normalized with respect to the reference frequency r
$\hat{\boldsymbol{\Gamma}}^{r,(a)}$	Harmonic pitch class profile (Gómez, 2006) (HPCP) extracted from an audio fragment (a). The first is centered around the pitch-class of r.

Symbol	Description
$\hat{\gamma}_i^{r,(a)}$	i^{th} bin of the HPCPs extracted from an audio fragment (a). The first bin is centered around the pitch-class of r.
$\triangle(x, y)$	"Octave-wrapped" cent distance from the frequency (x) to y
$\blacktriangle(x, y)$	Shortest "octave-wrapped" cent distance between the frequencies (x) to y
$\boldsymbol{H}^{(a)}$	Pitch distribution (PD) or pitch-class distribution (PCD), extracted from the audio fragment (a). The bins of the distribution are in Hertz.
$\boldsymbol{H}_P^{(a)}$	Pitch distribution (PD) extracted from the audio fragment (a). The bins of the distribution are in Hertz.
$\boldsymbol{H}_{PC}^{(a)}$	Pitch-class distribution (PCD) extracted from the audio fragment (a). The bins of the distribution are in Hertz.
$h_n^{(a)}$	The value to the n^{th} bin of the pitch distribution (PD) or pitch-class distribution (PCD) $\boldsymbol{H}^{(a)}$, extracted from the audio fragment (a)
$h_{P,n}^{(a)}$	The value to the n^{th} bin of the pitch distribution (PD) $\boldsymbol{H}_P^{(a)}$, extracted from the audio fragment (a)
$h_{PC,n}^{(a)}$	The value to the n^{th} bin of the pitch-class distribution (PCD) $\boldsymbol{H}_{PC}^{(a)}$, extracted from the audio fragment (a)
$\hat{\boldsymbol{H}}^{r,(a)}$	Pitch distribution (PD) or pitch-class distribution (PCD), extracted from the audio fragment (a). The bins of the distribution are in cents with the 0^{th} bin centered around the reference frequency r
$\hat{\boldsymbol{H}}_P^{r,(a)}$	Pitch distribution (PD) extracted from the audio fragment (a). The bins of the distribution are in cents with the 0^{th} bin centered around the reference frequency r
$\hat{\boldsymbol{H}}_{PC}^{r,(a)}$	Pitch-class distribution (PCD) extracted from the audio fragment (a). The bins of the distribution are in cents with the 0^{th} bin centered around the reference frequency r
$\hat{h}_n^{r,(a)}$	The value to the n^{th} bin of the pitch distribution (PD) or pitch-class distribution (PCD) $\hat{\boldsymbol{H}}^{r,(a)}$, extracted from the audio fragment (a) with the distribution having its 0^{th} bin centered around the reference frequency r

Symbol	Description
$\hat{h}_{P,n}^{r,(a)}$	The value to the n^{th} bin of the pitch distribution (PD) $\hat{\boldsymbol{H}}_{P}^{r,(a)}$, extracted from the audio fragment (a) with the distribution having its 0^{th} bin centered around the reference frequency r
$\hat{h}_{PC,n}^{r,(a)}$	The value to the n^{th} bin of the pitch-class distribution (PCD) $\hat{\boldsymbol{H}}_{PC}^{r,(a)}$, extracted from the audio fragment (a) with the distribution having its 0^{th} bin centered around the reference frequency r
ℓ_n	Accumulator function of the n^{th} bin of a pitch distribution (PD) or pitch-class distribution (PCD)
$\ell_{P,n}$	Accumulator function of the n^{th} bin of a pitch distribution (PD)
$\ell_{PC,n}$	Accumulator function of the n^{th} bin of a pitch-class distribution (PCD)
$b\ (\hat{\boldsymbol{H}})$	Bin size of the pitch distribution (PD) or pitch-class distribution (PCD)
$\sigma(\hat{\boldsymbol{H}})$	The width in the standard deviations of the Gaussian kernel used to "smooth" the pitch distribution (PD) or pitch-class distribution (PCD)
$\mu^{(a)}$	(Estimated) makam of an audio fragment (a)
$\mathfrak{m}^{(a)}$	"True" makam of an audio fragment (a)
$\boldsymbol{\mathcal{M}}$	The set of all makams
$\kappa^{(a)}$	Tonic pitch or pitch-class of an audio fragment (a)
$\mathfrak{k}^{(a)}$	"True" tonic pitch or pitch-class of an audio fragment (a)
$\boldsymbol{\Phi}^{(a)}$	Set of stable pitches or pitch classes in an audio fragment (a)
$\phi_i^{(a)}$	A stable pitch or pitch class in an audio fragment (a)
$\boldsymbol{\Phi}_{P}^{(a)}$	Set of stable pitches in an audio fragment (a)
$\phi_{P,i}^{(a)}$	A stable pitch in an audio fragment (a)
$\boldsymbol{\Phi}_{PC}^{(a)}$	Set of stable pitches in an audio fragment (a)
$\phi_{PC,i}^{(a)}$	A stable pitch in an audio fragment (a)

Symbol	Description
$\delta(\boldsymbol{H})$	Minimum ratio between the value of a peak and the highest value in a pitch distribution (PD) or pitch-class distribution (PCD), for the peak to be selected as a stable pitch or pitch-class.
$\boldsymbol{\mathcal{T}}$	The training model obtained for tonic identification and/or makam recognition using the either of the training schemes explained in (Gedik & Bozkurt, 2010) or (Chordia & Şentürk, 2013)
$\diamondsuit(\boldsymbol{x}, \boldsymbol{y})$	Distance or dissimilarity between two pitch distributions (PDs) or pitch-class distributions (PCDs) $\boldsymbol{x}$ and $\boldsymbol{y}$.
$\hat{\boldsymbol{\Phi}}_P^{\kappa^{(a)},(a)}$	Set of performed scale degrees in an audio fragment (a)
$\hat{\phi}_{P,i}^{\kappa^{(a)},(a)}$	A perfomed scale degree in an audio fragment (a)
$\bar{\mathfrak{S}}^{(a)}$	"True" audio section sequence in an audio fragment (a)
$\mathfrak{S}^{(a)}$	"True" audio section symbol sequence in an audio fragment (a)
$\bar{\mathfrak{s}}_i^{(a)}$	i^{th} "true" section in the $\bar{\mathfrak{s}}_i^{(a)}$
$\mathfrak{s}_i^{(a)}$	i^{th} "true" section symbol in the $\mathfrak{S}^{(a)}$
$\boldsymbol{D}^{r,(a,b)}$	Distance matrix between the audio recording (a) and the score fragment (b). The feature extracted from (a) (predominant melody or HPCPs) is normalized with respect to the reference pitch or pitch class r.
$\boldsymbol{B}^{r,(a,b)}$	Binary similarity matrix between the audio recording (a) and the score fragment (b). The feature extracted from (a) (predominant melody or HPCPs) is normalized with respect to the reference pitch or pitch class r.
β	Binarization criteria XX
$\pi^r(a, b)$	A link estimated between the audio fragment (a) and the score fragment (b). The feature extracted from (a) (predominant melody or HPCPs) is normalized with respect to the reference pitch or pitch class r.

Symbol	Description
$\boldsymbol{\varpi}^{r,(a,b)}$	Path followed by the link, $\pi^r(a,b)$, estimated between the audio fragment (a) and the score fragment (b). The feature extracted from (a) (predominant melody or HPCPs) is normalized with respect to the reference pitch or pitch class r.
$\varpi_i^{r,(a,b)}$	i^{th} point in the path followed by the link, $\pi^r(a,b)$, estimated between the audio fragment (a) and the score fragment (b). The feature extracted from (a) (predominant melody or HPCPs) is normalized with respect to the reference pitch or pitch class r.

List of Figures

List of Tables

Chapter 1

Introduction

> ...the most necessary, most difficult and principal thing in music, that is time...
>
> W. A. Mozart from *Mozart: The Man and the Artist, as Revealed in his own Words* by Friedrich Kerst, trans. Henry Edward Krehbiel (1906)

We live in a multicultural world that is replete with rich sources of data and information that keep increasing every passing day. The present day Information and Communication Technologies (ICT) and tools help us to generate, organize, interact, interpret, consume, assimilate the data and information and enhance our experience with the data, information and knowledge of the world. The technology needs in a multicultural context are evolving to cater to the complex sociocultural contexts in which these technologies and tools are being built and used.

Music is an integral part of our lives and is being produced and consumed at an ever increasing rate. The consumption channels and practices of music have changed significantly over the last two decades. With music going digital, there are large collections of music available on demand to users, which provides a great opportunity to enhance our experience interacting with music. The interaction with music has grown beyond just listening into an enriching and engaging experience with the music content. In such a scenario, there are significant efforts to build automatic tools and technologies to enhance our experience with large (and ever increasing in size) music collections. Music being a sociocultural phenomenon necessitates these automatic tools to be aware and adapt to such a context and cater to specific music cultures, music producers (musicians and artists) and the widely diverse audiences.

Music Information Research (MIR) aims to develop tools and applications for representation, understanding, analysis, and synthesis of music. Though a new and interdisciplinary field of research, it has a significant community working on various problems within the purview of MIR. MIR focuses on understanding and modeling what music is and how it functions. Its basic aim is to develop veridical and effective computational models of the whole music understanding chain, from sound and structure perception to the kinds of high-level concepts that humans associate with music, such as melody, rhythm, harmony, structure, mood and other possibly subjective attributes and characteristics. Automatic music analysis in MIR aims to 'make sense' of music and extract useful, musically relevant and semantically meaningful information from music pieces and music collections.

In the last two decades, MIR has received significant attention from the research community and has addressed several relevant research problems advancing the field of sound and music computing. However, the current research in MIR has been largely limited to eurogenetic (popular) music [1] cultures and do not generalize to other music cultures of the world. The approaches have not been developed within a multicultural context and are incapable of extending to the wide variety of music cultures we encounter.

When I say we it means some collaboration. appendixXX lines manipulated by is typically comparable to the principal "coder" of the module because of refactoringXX

Music is a complex phenomenon and there are many types of data sources that can be used to study it, such as audio recordings, scores, videos, lyrics and social tags. At the same time, for a given piece there might be many versions for each type of data, for example we find cover songs, various orchestrations and diverse lyrics in multiple languages. Each type of data source offers different ways to study, experience and appreciate music. If the different information sources of a given piece are *linked* with each other (Thomas, Fremerey, Müller, & Clausen, 2012), we can take advantage of their complementary aspects to study musical phenomena that might be hard or impossible to investigate if we have to

[1] The term eurogenetic music was introduced by Srinivasamurthy, Holzapfel, and Serra (2014) to avoid the misleading dichotomy of Western and non-Western music. The discussed theoretical constructs of western music are motivated by music of the European common practice period. We use the word "genetic" rather with its connotation as "pertaining to origins", coined in 1831 by Carlyle from Gk. genetikos "genitive", from genesis "origin", and not in its biological sense as first applied by Darwin in 1859 (http://www.etymonline.com). The term was proposed by Prof. Robert Reigle (MIAM, Istanbul) in personal communication.

study the various data sources separately.

The linking of the different information sources can be done at different time spans, e.g. linking entire documents (Ellis & Poliner, 2007; Martin, Robine, & Hanna, 2009; Serrà, Serra, & Andrzejak, 2009), structural elements (Müller & Ewert, 2008), musical phrases (Wang, 2003; Pikrakis, Theodoridis, & Kamarotos, 2003), or at note/phoneme level (Niedermayer, 2012; Fujihara & Goto, 2012). Moreover there might be substantial differences between the information sources (even among the ones of the same type) such as the format of the data, level of detail and genre/culture-specific characteristics. Thus, we need content-based (Casey et al., 2008), application-specific and knowledge-driven methodologies to obtain meaningful features and relationships between the information sources. The current state of the art in Music Information Retrieval (MIR) is mainly focussed on Eurogenetic [2] styles of music (Tzanetakis, Kapur, Schloss, & Wright, 2007) and we need to develop methodologies that incorporate culture-related knowledge to understand and analyze the characteristics of other musical traditions (Holzapfel, 2010; Şentürk, 2011; Serra, 2011).

Throughout the text, in the data collection and in the supplementary results, we use MusicBrainz Identifier (MBID) as an unique identifier for the compositions and audio recordings. For more information on MBIDs please refer to `http://musicbrainz.org/doc/MusicBrainz_Identifier`.

[2] We apply this term because we want to avoid the misleading dichotomy of Western and non-Western music.

Chapter 2

Background

2.1 Problems

Refer to Bozkurt review

2.2 Musical Background

Most of the melodic, rhythmic and compositional aspects of OTMM can be explained by the terms *makam, usul* and *form*, respectively. *Makam*s constitute the melodic structure of most of the traditional music repertoires in Turkey. Makams are modal structures, where the melodies typically revolve around an initial tone and a final tone (Ederer, 2011). The final tone is typically used synonymous to tonic. The rhythmic structure of OTMM is described by the *usul*s. A certain *usul* can be described by a group of strokes with different velocities, which imply the beats and downbeats in the rhythmic pattern. OTMM consists of both instrumental and vocal forms. Some of these forms are related with religious music and occasionally performed in religious ceremonies. There are also non-metered improvisational forms such as *taksim* and *gazel*.

OTMM has been predominantly an oral tradition for centuries. For this reason the performance practice is the fundamental unit of OTMM. There are several theories attempting to explain the makam practice. The mainstream theory is the music theory XX (AEU theory) (Arel, 1968). The the music theory XX (AEU theory) divides a whole tone into 9 equidistant intervals, which can be approximated from 53-TET (tone equal tempered) intervals (Tura, 1988).

Since early 20^{th} century, a score representation extending the traditional Western music notation has also been used for OTMM (Popescu-

Judetz, 1996). The extended Western notation typically follows the rules of AEU theory. Most of the scores are transcriptions written sometimes centuries after the pieces were composed. They tend to notate simple melodic lines. Makam musicians follow the scores of compositions as the guideline but they extend them considerably during the performance. These include expressive timings, adding notes and non-notated embellishments. The intonation of some intervals in the performance might differ from the notated intervals as much as a semi-tone (Signell, 1986). Due to these expressive decisions, there may be high degrees of variance between different interpretations of the same piece.

Signell bati notalari bestelerin erisilebilirligini arttirdi, fakat (seyir gib) geleneksel ozellikler kaybedildi diyor (kitap, sayfa 3) XX

Notalarin kaynagi 1) 20. yuzyilda bestecilerin direkt kendi yazdigi notalar, 2) muzisyenlerin zihinlerinden gecirdigi transkripsiyonlar, 3) ebced, harsarsum notalarindan vb. extended Western notation'a gecirilenler. (Signell kitap, sayfa 4)

Chapter 3

Data corpora

For computational studies on a specific type of musics, there is a need for corpora, which constitutes the studied aspects of the specific music. A music corpus may consist of multiple information sources such as audio recordings, music scores, lyrics and editorial metadata. We can also group the corpora into two types: research corpus and test dataset (Serra, 2014). A research corpus is a data collection that represents the ”real world” for a specific research problem. A test dataset is a collection for a specific research task to test, calibrate and evaluate particular methodologies.

Serra (2014) provides such criteria for the design of culture specific corpora, which are specified as *purpose, coverage, completeness, quality* and *re-usability*. To elaborate, the *purpose* of the corpora should be well-defined to facilitate research tasks. The corpora should be of good *coverage* to represent the music tradition and include metadata with a high degree of *completeness* related to studied aspects of the music. The corpora should attain a certain quality and it should be *re-usable* for future research. The Turkish Makam Corpus is designed with these considerations in mind.

In this Chapter, we present a corpus for computational research of OTMM. We explain the corpus with respect to the information sources that are used to populate it, namely audio recordings, machine-readable music scores and editorial metadata. For each type of data, we discuss the *purpose, coverage, completeness, quality* and *re-usability* criteria, when applicable. We also describe the test datasets of OTMM we gathered in the scope of the CompMusic Project.

The rest of the Chapter is as follows: Section 3.1 explains the makam music research corpus and the criteria that we used for creating this corpus. Section 3.2XX. Section 3.3 gives a detailed information about the test datasets and Section 3.4 wraps up the paper with a brief conclusion.

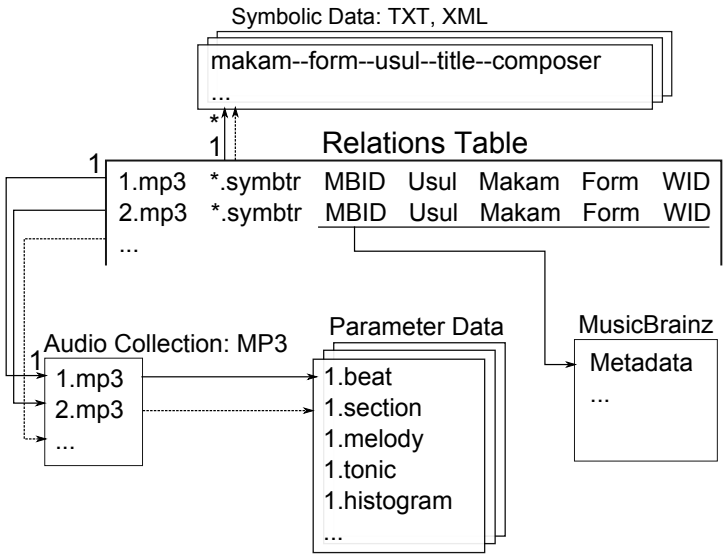


Figure 3.1: Block Diagram of the research corpus. A-MBID and W-MBID refers to the MBID of the audio recording and the MBID of the related work, respectively.Replace with turkey presentationXX

3.1 Research Corpus

In Compmusic project, we mainly focus on the melodic and the rhythmic characteristics of OTMM. To study these aspects of the music tradition, we have been collecting audio recordings and music scores. From the audio recordings we can extract the characteristics of interpretations of compositi ons performed by makam musicians. The music scores, on the other hand, provide an easy-to-access medium to extract the musical elements. We additionally store editorial metadata about OTMM. The metadata includes information related to the audio recordings and music scores as well as additional information such as the birth date of the artists and relevant web sources about the entities. The metadata also include the relationships between each entity so that the connections within the metadata can be exploited to access relevant information in a structured way.

For this purpose, a team with the support of more than 15 collaborators, has been working to collect and classify all the available data. In this section, we explain the audio recordings (Section 3.1.1), the music scores (Section 3.1.2) and the editorial metadata (Section 3.1.3) in the research corpus. The metadata related to audio recordings and the music scores are mainly explained within the corresponding source type. The

corpus is discussed in terms of the *purpose, coverage, completeness, quality* and *reproducibility* of the audio recordings and the music scores(Serra, 2014). In Section 3.1.3 we mostly focus on the overall statistics about the metadata as well as the statistics of inter-relationships.

In our corpus, we use MusicBrainz to store the metadata. MusicBrainz assigns a unique identifier, called MusicBrainz Identifier (MBID) to each entry (e.g. releases, audio recordings, artists).[1]

3.1.1 Audio Collection

While creating the corpus, one of our major efforts has been directed to create an audio collection representative of OTMM. The audio collection consists of almost 6000 stereo recordings, which are sampled at 44.1 kHz and 160 kbps and stored in MP3 audio format. This collection corresponds to 375 hours of play time. The collection includes both solo recordings and ensemble/chorus recordings. They span a time period from the start of the 20^{th} century to nowadays. The collection also covers various forms, some of which are part of the the folk (e.g. türkü) and religious repertoire (e.g. ilahi) (Table 3.2).

Coverage

Historically, TRT has the most representative audio productions of OTMM. However, most of their audio collection is not open to public and only a small part of this collection is commercially available. Apart from TRT, there are numerous labels, which have released recordings of OTMM. For these reasons, it is hard to collect the overall statistics of OTMM recordings.

So far, we have focused our efforts on gathering an audio collection of classical repertoire, including the available commercial recordings from TRT and other important labels. We also include several non-commercial recordings, provided that they have a good overall musical and production quality. In Table 3.1, we present the general statistics of the gathered audio collection.

Completeness

Along with the audio recordings, we also collect editorial metadata given in the album covers. In case an album cover does not provide related

[1] For more information on please refer to http://musicbrainz.org/doc/MusicBrainz_Identifier.

	#
Recordings	5953
Releases	340
Works	2696
Artists	536
Makams	150
Usuls	88
Forms	120

Table 3.1: Number of unique recordings, releases, works, artists, makams, usuls and forms in the CompMusic OTMM corpus

Form	#	Form	#
Şarkı	2378	Türkü	157
Taksim	1141	Ağır Semai	146
Peşrev	437	Beste	146
Sazsemaisi	390	İlahi	118
Yürük Semai	164	Other (92 forms)	836

Table 3.2: The most represented forms in the audio corpus and the corresponding number of audio recordings. Note that multiple compositions and improvisations might be performed in an audio recording. Hence an audio recording may have multiple forms associated with it.

metadata (e.g. related work, makam) we attempt to fill the missing metadata by accessing other information sources available. The procedure is as follows: if the makam, usul, form, composer information is missing, a search with the name of the recording is performed in the online score collections (such as the ones explained in Section 3.1.2), and the missing information is obtained from the matched score. For recording names made of form information (such as *"Hicaz Peşrev*"), since there can be many "Hicaz Peşrev"s, other "Hicaz Peşrev"s in the audio collection are listened and checked if there exists a match. If a match is found, its makam, form, usul and work information are copied.

The completeness of the audio related metadata is shown in Table 3.3. While checking the completeness of the artist metadata in the audio recordings, we assumed a recording is complete if it has at least one artist associated with it. Note that this does not imply a strict completeness with respect to the artist metadata since a lot of recordings (esp. ensemble recordings) do not have the complete information about the members in the album covers.

	# Recordings	% of total
Releases	5953	100%
Works	4626	78%
Artists	5953	100%
Makams	5544	93%
Usuls	5349	90%
Forms	5770	97%

Table 3.3: The number of audio recordings for which the corresponding metadata is available. Note that 1141 audio recordings are improvisations, which do not have a work.

Quality

The audio recordings are stored in MP3 format. This format is chosen due to its small storage size compared to other audio formats. In the selection process, we did not include releases with low production quality (except the historical recordings of the grand masters) or with performances with low musical quality (e.g. MIDI accompaniment).

Re-Usability

The non-commercial recordings in our research corpus are freely-available. Most of these non-commercial recordings can be downloaded from *Internet Archive*[2] or the respective websites where they were originally fetched from.[3]

Due to copyright restrictions, we cannot distribute the commercial audio recordings and their cover arts in our collection. On the other hand, they are available for browsing and listening through Dunya-makam (Porter, Sordo, & Serra, 2013b; Şentürk, Ferraro, Porter, & Serra, 2015). Moreover the annotations on all the audio recordings and the various features extracted from them are freely distributed and can be used for computational research purposes.

3.1.2 Score-Collection

XX from previous writing

In the analysis, we use the release v2.4.1 of the **SymbTr** score collection K. Karaosmanoğlu (2012).[4] This release includes 2200 scores

[2] http://tinyurl.com/n9omoue

[3] e.g. http://neyzen.com/ney_den_saz_eserleri.html

[4] https://github.com/MTG/SymbTr/releases/tag/v2.4.1

from the folk and classical repertoires. It is currently the largest and the most representative machine-readable score collection of OTMM aimed at research purposes Uyar, Atlı, Şentürk, Bozkurt, and Serra (2014). The scores typically notate the basic melody of the composition devoid of the performance aspects such as intonation deviations and embellishments. The scores also include editorial metadata such as the composer, the makam, the form, the usul (rhythmic structure) of the composition. We use the scores in txt format in our analysis, as they are the reference format from which the other formats are generated.

The content in the **SymbTr**-txt scores are stored as "tab separated values," where each row is a note or an editorial annotation (such as usul change) and each column represent an attribute such as the note symbol, the duration, the measure marking and the lyrics. The pitch intervals are given according to both the 24 tone-equal-tempered (TET) system defined in the Arel-Ezgi-Uzdilek theory and the 53-TET system.[5] The lyrics are synchronous to the note onsets on the syllable level. The final syllable of each word ends with a single space and the final syllable of each poetic line ends with double spaces K. Karaosmanoğlu (2012). Some columns may be overloaded with additional types of information. For example the lyrics row also includes editorial annotations such as the section names, instrumentation and tempo changes, entered in capital letters.

explain the **SymbTr**-txt structure explain the phrase codes, explain the flavors

explain the mu2-headers

explain note convention Symbol-Cotave-Accidental-Comma

Figure showing symbtr format XX

MIDI having tuning information (K. Karaosmanoğlu, 2012).

The existing music scores of OTMM are mostly in physical formats, such as hand-written scores and books. There are also scores available in digital formats like JPEG and Portable Document Format (PDF). Typically, these types of scores are not very useful in computational research[6] since the musical elements (e.g. notes, durations, tempo, melodic structure, measure info) cannot be directly read by the machines.

In the scope of the CompMusic Project, a music score collection has been created called **SymbTr** (K. Karaosmanoğlu, 2012). The **SymbTr** collection consists of score-files in text format as well as the corresponding PDF and MIDI files. The naming of the text-scores follows a convention, which provides some of the key information to identify and cat-

[5] The unit interval of the 53-TET, which is simply the $1/53$th of an octave, is called a Holderian comma (Hc).

[6] An obvious exception is optical music recognition.

egorize the compositions. This structure includes the makam, form, usul, *title* (for vocal compositions) and *composer name* of the music piece. Apart from the music scores, the collection also consists of makam, usul and form libraries, which provide additional structured musical information about these attributes. The symbols of each note in the music score are both given according to Arel-Ezgi-Uzdilek and 53-TET theory, as well as the corresponding pitch intervals in Holderian comma (Section 2.2). In addition to the note information, the nominal tempo, section information, beat information are given. In vocal compositions the lyrics are aligned to the notes in the syllable-level.

Recently, we released a second version of this score collection[7]. In this version, there are 2200 unique compositions. The general statistics are given in Table 3.4. In addition to the text-scores we provide the scores in MusicXML 3.0 format.[8] MusicXML is a format which can be imported/exported by the well-known score editing programs such as MuseScore, Finale and Sibelius.

Now we present the coverage, completeness and quality and re-usability of the updated **SymbTr** collection.

Coverage

To the best of our knowledge there are only two machine-readable score collections of OTMM other than **SymbTr**, which can be used for computational research. First is the Uzun Hava Humdrum Database (**UHHD**) (Şentürk, 2011). This repository features the 77 music scores of *uzun hava*s, a non-metered improvisational form of Turkish folk music. Due to its specialized nature, this collection is not considered for comparison. A more relevant collection is the Türk Sanat Müziği Derlemi (English: Turkish Art Music Corpus) (**TSMD**) (Atalay & Yöre, 2011), which includes 600 compositions equally divided into 20 makams (i.e. 30 pieces per makam). It is smaller than the **SymbTr** collection.

To get a better means of comparison, we focus on online music score collections, in which the music scores are stored in various image formats. Although this repositories are not machine-readable, hence insuitable for computational research, they contain a much greater amount of music scores with respect to the machine-readable collections. This leads us to accept these collections as our references while measuring the coverage of our score collection.

[7] https://github.com/MTG/SymbTr/releases/tag/v2.0.0
[8] http://www.musicxml.com/

Among these online collections, we selected **TRT-TTMA**[9] and the **TMKH** collections.[10] Similar to the case explained in Section 3.1.1, **TRT-TTMA** is arguably the most reliable resource. Currently, **TRT-TTMA** includes ~ 17000 scores in total, all of which are manually scanned from physical scores. The scores in **TRT-TTMA** are sold online. **TMKH** is a collection created by funds from the *Istanbul 2010 European Capital of Culture Organization*[11] through the *European Union*. The **TMKH** collection includes ~ 45000 scanned scores (where multiple versions are available for almost each work) of the personal collections of 3 professional makam musicians/scholars. The collection is free, however there are several restrictions on the site navigation and the number of daily downloads. From these collections, we crawled the metadata of the music scores to obtain the statistics (Table 3.4). **TRT-TTMA** has some duplicate entries and some compositions which are not in the context of OTMM, *(e.g. church chants, operettas etc.)*. When these compositions are removed from comparison, the number of compositions are reduced to ~ 12000.

Some of the names of the makam, usul and form in the collections slightly differ from each other. To match the names, we use an automatic string matching method. The algorithm we chose uses a weighted measure which consists of two edit distance measures:[12] longest common subsequence, and Damerau–Levenshtein distance, which have 0.7 and 0.3 weights respectively. The weights are determined empirically by varying them to find a configuration that results in satisfactory matches. Finally we do a manual check and remove any remaining erroneous matches.

To assess how well the **SymbTr** collection covers the OTMM, we compare our collection against these music score collections. From each collection we report the number of compositions, composers, makams, forms and usuls. We also check how much the makams, forms and usuls in the **SymbTr** collection overlap with the corresponding type of metadata in other collections. We define overlap as:

$$\mathcal{O}\left(\mathcal{A}^{(\mathcal{R})}\right) = \frac{\left|\mathcal{A}^{(\textbf{SymbTr})} \cap \mathcal{A}^{(\mathcal{R})}\right|}{\left|\mathcal{A}^{(\mathcal{R})}\right|} \tag{3.1}$$

[9] http://www.trtkulliyat.com/

[10] http://www.sanatmuziginotalari.com/; accesible through http://turkmusikisivakfi.org/.

[11] http://istanbul2010.org/

[12] The implementation is here: https://github.com/gopalkoduri/string-matching/

	SymbTr	TSMD	TRT-TTMA	TMKH
Compositions	2,200	600	12,035	45,368
Composers	455	230	1,447	2,674
Makams	157	20 (100%)	293 (49%)	317 (45%)
Usuls	84	46 (89%)	N/A	382 (22%)
Forms	62	6 (100%)	110 (35%)	90 (31%)

Table 3.4: Coverage of the score collection in the corpus. The number in paranthesis is the overlap measure Equation 3.1 in percentage. N/A indicates that data is not available.

where $\mathcal{A}^{(\mathbf{SymbTr})}$ is the set of the subjected attribute (makam, usul or form) from our score collection **SymbTr**, $\mathcal{A}^{(\mathcal{R})}$ is the set of the subjected attribute from the referance collection $\mathcal{R}$, against which we want to measure our collection's coverage and $\mathscr{O}\left(\mathcal{A}^{(\mathcal{R})}\right)$ is the overlap, which demonstrates how much the subjected attribute of $\mathcal{R}$ is represented in **SymbTr**.

Table 3.4 shows the overlap of the makams, usuls, forms between our collection and the three music score collections explained above. We can observe that the **SymbTr** collection covers almost all of the makams, usuls and forms in the **TSMD**. While the number of compositions are much less than **TRT-TTMA** and **TMKH**, there is a fair number of overlapping makams, usuls and forms the the **SymbTr** collection with respect to **TRT-TTMA** and **TMKH**. Note that in OTMM, it is common to have different titles for the scores of the same composition (first line of lyrics, the chorus etc.) and a composer might have various names (e.g. aliases, titles, added surname etc.). It is hard to obtain an accurate overlap for these attributes. Hence, the overlap of the composers and the compositions are not computed.

Note that the makams, usuls and forms listed in the score collections are not evenly distributed, some of these attributes are much more represented than the others. Hence we should also consider the coverage of these attributes with respect to their rate of presence in the reference collection. Taking these circumstances into consideration, we have modified the overlap function by adding some measure parameters as explained in detail below:

- The attribute set $\mathcal{A}^{(\mathcal{R})}$ of the collection $\mathcal{R}$ has $|\mathcal{A}^{(\mathcal{R})}|$ elements.
- Each element $\alpha_k^{(\mathcal{R})}$ has an occurrence count $o\left(\alpha_k^{(\mathcal{R})}\right)$ in the collection such that:

$$\sum_{k=1}^{|\mathcal{A}^{(\mathcal{R})}|} o\left(\alpha_k^{(\mathcal{R})}\right) = |\mathcal{R}| \tag{3.2}$$

where $|\mathcal{R}|$ indicates the number of scores in the reference collection, $\mathcal{R}$. All $\alpha_k^{(\mathcal{R})}$'s are already ranked with respect to their occurances $o\left(\alpha_k^{(\mathcal{R})}\right)$ in $\mathcal{A}^{(\mathcal{R})}$ (i.e. $\mathcal{A}^{(\mathcal{R})}$ is enumerated) such that $o\left(\alpha_k^{(\mathcal{R})}\right) \geq o(\alpha_{k+1}^{(\mathcal{R})}), \forall k \in \left[1 : \left|\mathcal{A}^{(\mathcal{R})}\right| - 1\right]$.

- The occurence ratio $\hat{o}\left(\alpha_k^{(\mathcal{R})}\right)$ is defined as:

$$\hat{o}\left(\alpha_k^{(\mathcal{R})}\right) = \frac{o\left(\alpha_k^{(\mathcal{R})}\right)}{|\mathcal{R}|} \tag{3.3}$$

- Then we cumulatively add the ratios to find the total occurrence ratio $\hat{O}\left(\alpha_k^{(\mathcal{R})}\right)$ up to $\alpha_k^{(\mathcal{R})}$ as,

$$\hat{O}\left(\alpha_k^{(\mathcal{R})}\right) = \frac{\sum_k^{|\mathcal{A}^{(\mathcal{R})}|} o\left(\alpha_k^{(\mathcal{R})}\right)}{|\mathcal{R}|} \tag{3.4}$$

Notice that, $\hat{O}\left(\alpha_{|\mathcal{A}^{(\mathcal{R})}|}^{(\mathcal{R})}\right) = 1$, as it includes all the score collection.

- We measure the overlap of **SymbTr** against $\hat{O}\left(\alpha_k^{(\mathcal{R})}\right)$'s.

$$\mathscr{O}_k\left(\mathcal{A}^{(\mathcal{R})}\right) = \frac{\left|\mathcal{A}^{(\textbf{SymbTr})} \cap \mathcal{A}^{(\mathcal{R})}[1:k]\right|}{\left|\mathcal{A}^{(\mathcal{R})}[1:k]\right|} \tag{3.5}$$

- Finally we define the attribute coverage $\mathscr{C}(\mathcal{A}^{(R)})$ of **SymbTr** against $\mathcal{R}$.

$$\mathscr{C}(\mathcal{A}^{(R)}) = max\left(\hat{O}(\alpha_k^{(\mathcal{R})})\right) \quad | \quad \mathscr{O}_k\left(\mathcal{A}^{(\mathcal{R})}\right) = 1 \tag{3.6}$$

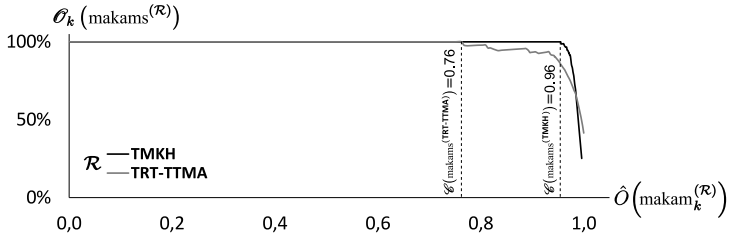


Figure 3.2: Overlap with respect to the makam, our corpus vs **TMKH** and **TRT-TTMA**. The dashed lines indicate the coverage values for **TRT-TTMA** (0.76) and **TMKH** (0.96).

By applying this modified procedure, we have reached detailed results specifically different for the entities with different occurence ratios. In Figure 3.2, two functions for $\mathscr{O}\left(makams^{(\mathcal{R})}\right)$ are provided with respect to $\alpha^{(\mathcal{R})} = makams$. In the Figure, **SymbTr** corresponds to our collection and $\mathcal{R}$ corresponds to either **TMKH** or **TRT-TTMA**. The overlap of **TMKH** by **SymbTr** is 100%, i.e. $\mathscr{O}\left(makams_k^{(\textbf{TMKH})}\right) = 1$, when the makams which contribute less then 0.1% to the **TMKH** are excluded, i.e. $\hat{o}\left(makam_k^{(\textbf{TMKH})}\right) < 0.001$. According to this specific k value, **SymbTr** covers **TMKH** by 96%; $\mathscr{C}\left(makams^{(\textbf{TMKH})}\right) = 0.96$. The overlap rate of **TRT-TTMA** by **SymbTr** reaches to 1, i.e. $\mathscr{O}\left(makams_k^{(\textbf{TRT-TTMA})}\right) = 1$ at the point where the makams which constitute less than 0.6%, i.e. $\hat{o}\left(makam_k^{(\textbf{TRT-TTMA})}\right) < 0.006$ are excluded, providing the coverage value of 76%, i.e. $\mathscr{C}\left(makams^{(\textbf{TRT-TTMA})}\right) = 0.76$.

For the form attribute, the coverage $\mathscr{C}\left(forms^{(\textbf{TMKH})}\right)$ is 98% with $\hat{o}\left(form_k^{(\textbf{TMKH})}\right) < 0.002$ excluded for **TMKH**. For **TRT-TTMA**, the form coverage $\mathscr{C}\left(forms^{(\textbf{TRT-TTMA})}\right)$ is 86%, when the forms with $\hat{o}\left(form_k^{(\textbf{TRT-TTMA})}\right) < 0.01$ excluded. **TRT-TTMA** does not provide the usul information. For **TMKH**, **SymbTr** has a usul coverage $\mathscr{C}\left(usuls^{(\textbf{TMKH})}\right)$ of 94%, when $\hat{o}\left(usul_k^{(\textbf{TMKH})}\right) < 0.003$ are excluded.

Completeness

As explained above, the **SymbTr**-scores have the makam, usul, form and *composer* information, as well as the beat information, nominal tempo, lyrics and section note boundaries and labels. The section boundaries in the vocal compositions indicated by single space and double spaces but

the section names are not given in these cases. Therefore we can argue that **SymbTr**-scores are editorially complete except the section labels.

Paraphrase XX we use the explicit section names along with the poetic line ends mentioned above in the section extraction step. However, this set of editorial annotations does not convey the complete information about the section boundaries and the section names. First, the section name (and hence the first note of a section) is only given for the instrumental sections and the final note of these sections are not marked at all. Moreover, the section name does not indicate if there are any differences between the renditions of the same section. Regarding the vocal sections, only the last syllable of a poetic line is marked as explained above. This mark does not typically coincide with the actual ending of the vocal section since a syllable can be sung for longer than one note or there might be a short instrumental movement in the end of the vocal section. Out of 2200, 1771 txt-scores in the **SymbTr** collection has some editorial section information. The remaining 429 scores either lack the editorial section information or they are very short such that they do not have any sections.

Now they are stored XX Note that the score metadata are not currently stored in MusicBrainz. However, we have been organizing the relationships between **SymbTr** scores and corresponding work MBIDs, which will also be available in MusicBrainz. There is currently an intersection of 741 pieces, with 2046 corresponding recordings, between 2200 scores in our corpus and 2696 works in MusicBrainz. Detailed statistics of relation between the works in MusicBrainz and **SymbTr** scores can be seen in Table 3.5.

Quality and Re-usability

The aim of the MusicXML Makam Music Score Collection is to make this score library reachable for the interested community by providing it in a format that they can use in the software they are familiar with. In this way the community can make necessary changes on a certain work or create their own scores and share, by providing them in a machine-readable format which serves for widening the spectrum of the corpus for research. In this way, the corpus would represent the OTMM tradition better and facilitate the research done on OTMM.

Turkish Makam Music MusicXML Score Collection contains and supports all the accidental symbols in the notation of this music tradition. This provides users and researchers to work with the correct information of the music tradition.

n	#works with **SymbTr**, related to *n* #recordings	total #audio recordings
1	259	259
2	164	328
3	113	339
4	81	324
5	48	240
6	32	192
7	19	133
8	10	80
9	6	54
10	3	30
11	5	55
12	1	12
Total	741	2046

Table 3.5: Number of works with **SymbTr**-scores distributed with respect to the related number of audio recordings

The idea of having this score collection in MusicXML format solves multiple problems including the community content generation, community content examination, portability and re-usability. MusicXML is a format that can be used as a simple text file to be interpreted for research or that it can be used as a score file for the sheet music software. In both manners, the library can be used as a resource for infinitely many times as soon as the files are not changed on purpose. This is important because while organizing a community generated collection, we should use representations that facilitate users familiarity.

3.1.3 Metadata

The metadata includes the general information about our corpus. It is mainly related to the audio recordings in the releases and the relevant compositions. The metadata also include other related information such as artist information, URLs to the related web pages. The metadata is interlinked with each other through relationships such that the instrument an artist played in an audio recording or the lyricist of a composition is known. We can use these relationships to navigate the concepts of OTMM with ease.

So far we have collected more than 27000 entries of metadata. They

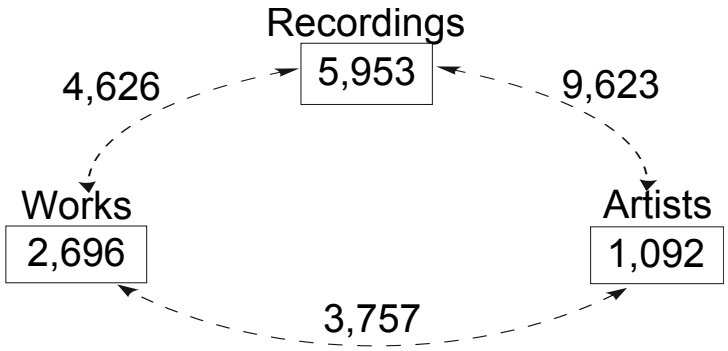


Figure 3.3: Number of audio recording, work and artist entities in the metadata and the number of relationships between each type of entity

are stored in MusicBrainz.[13] Figure 3.3 shows the number of entries related to recordings, works and artists and also the number of relationships between each entity. Among the metadata entered to MusicBrainz related to 7000 audio recordings from Turkey (pop, rock etc...), 6000 are metadata entered within the score of CompMusic OTMM research corpus.

3.2 Music Theory

Extracted from

Makam, usul, form dictionaries:

`https://github.com/sertansenturk/otmm_corpus_stats/blob/master/data/makamFormUsulDicts/makam_extended.json`

3.3 Test Datasets

Test datasets are collections arranged for the specific research problems. These datasets are typically used as the ground-truth to evaluate methodologies applied to certain problems. They can be composed of different types of data such as synthetic data or data with manual annotations.

Bozkurt et al. (Bozkurt, Ayangil, & Holzapfel, 2014) made a review of computational analysis literature for OTMM. The datasets that we mention in this section are useful for some of the research tasks discussed in this paper such as structure analysis, automatic tonic identification, automatic ornamentation segmentation, melodic phrase segmentation.

[13] `http://musicbrainz.org/collection/544f7aec-dba6-440c-943f-103cf344efbb`

In our test datasets we have manual annotations by the experts of OTMM tradition. The details of test datasets are discussed in Section 3.3.1, 3.3.7, 3.3.8 and 3.3.3.

3.3.1 Melodic Segmentation Test Dataset

Karaosmanoğlu and Bozkurt have studied the problem of usul and makam driven automatic melodic segmentation for Turkish Music (Bozkurt, Karaosmanoğlu, Karaçalı, & Ünal, 2014). For this research, 899 **SymbTr**-scores were manually annotated into melodic segments by 3 experts. There are a total of 31362 phrase annotations in this dataset.[14].

Score analysisde kullaniliyor. XX

3.3.2 Symbolic Section Dataset

Used in (Şentürk & Serra, 2016b)

3.3.3 Tonic Test Dataset

For score-informed tonic identification, we annotated the tonic frequency of 257 audio recordings associated with 57 compositions in total (Şentürk, Gulati, & Serra, 2013). The **SymbTr**-score of the related composition for each audio recordings is given in the metadata.

tonic last note dataset

otmm makam recognition dataset

XX Additionally, tonic frequencies of 3400 audio recordings have been manually annotated by the musicians.[15]

3.3.4 Makam Recognition Dataset

(Karakurt, Şentürk, & Serra, 2016)'de kullanildi, tonic de var.

3.3.5 Section Linking Dataset

For section linking experiments, we have also annotated the start and end of each section (as given in the corresponding **SymbTr**-score) in the same 257 audio recordings mentioned above (Şentürk et al., 2013) (Section 3.3.3) (Şentürk, Holzapfel, & Serra, 2014). The number of section annotations in the test dataset is 2095.

[14] http://akademik.bahcesehir.edu.tr/~bbozkurt/112E162_en.html

[15] http://compmusic.upf.edu/node/230

3.3.6 Partial Audio-Score Alignment Dataset

in github. part of it is used in evaluation the variant of the predominant melody extraction procedure described in (?, ?) using the post filtering method described in (Bozkurt, 2008) ($\texttt{ATL-MEL}_f$) (Atlı, Uyar, Şentürk, Bozkurt, & Serra, 2014) by Atlı (2016)

3.3.7 Audio-Score Alignment Test Dataset

For the initial experiments in audio-score alignment of OTMM, we collected 6 audio recordings of 4 peşrev compositions (Şentürk, Gulati, & Serra, 2014). The audio recordings in the dataset have the annotated tonic frequencies, 51 section annotations and 3896 note annotations in total. The note annotations follow the note sequences in the corresponding **SymbTr**-scores.

Newer version in github.

3.3.8 Audio-Lyrics Aligment Test Dataset

Dzhambazov has worked on automatic lyrics-to-audio alignment in OTMM (?, ?). 10 şarkı were manually divided into sections and aligned to the recordings[16].

Newer version in SMCXX

3.4 Conclusion

In this paper a research corpus of OTMM is presented. The corpus is created under the considerations to meet some criteria: purpose, quality, completeness, coverage and re-usability. We also present some test datasets, which have been used to test and calibrate some computational tasks (Bozkurt, Karaosmanoğlu, et al., 2014; ?, ?; Şentürk et al., 2013; Şentürk, Gulati, & Serra, 2014; Şentürk, Holzapfel, & Serra, 2014). We hope that this research corpus and the test datasets will facilitate academic studies in several fields such as music information retrieval and computational musicology.

[16] http://compmusic.upf.edu/node/226

Chapter 4

Score Analysis

In analyzing a music piece, scores provide an easily accessible symbolic description of many relevant musical components. Moreover they typically include editorial annotations such as the nominal tempo, the rhythmic changes and structural markings. These aspects render the music score a practical source to extract and analyze the melodic, rhythmic and structural properties of the studied music.

In this chapter, automatic content description procedure applied to the music scores of OTMM is explained. First, the editorial information in the scores (such as composer, makam and tempo) is parsed. This information is fetched from three different sources, namely the **SymbTr** scores in the txt format, in the mu2 format and MusicBrainz. Next, the information is validated against each other. The main contribution in this chapter is the structural analysis methodology (Section 4.3), which aims to extract and label the melodic and lyrical organization both on phrase-level and section-level, using symbolic information available in the music scores of OTMM. The method labels the extracted sections and phrases semiotically according their relations with each other using basic string similarity and graph analysis. Our contributions are:

- Parsing and
- Automatic content validation
- An automatic structural analysis method applied on Ottoman-Turkish makam music scores
- A novel semiotic labeling method based on network analysis
- An open implementation of the methodology extending our existing score parser
- A dataset of sections and phrases automatically extracted from more than 1300 and 1750 music scores, respectively

The editorial information is used extensively in the joint analysis of music scores and audio recordings (Chapter sec:) as a reliable XX, which

The structure of the rest of the chapter is as follows: XX

4.1 Metadata

Music scores, by their nature, contain curated editoral metadata relevant to the composition or the recording (in case of transcriptions of performances). Since the scores we process are written, edited and curated by experts (Section 3.1.2), this information is typically incorporates reliable information about the score and relevant entities.

Restricting ourself to machine-readable scores (e.g. not considering scores stored in image or some vector formats such as PDF, which would require optical music recognition (OMR)) XX. Nevertheless, extracting such metadata may not be a straightforward task due to the format of the music score (e.g.text file in tab separated values (TSV) vs. MusicXML format), the organization within the score (e.g. a score consisting of a metadata header vs. a score with the metadata stored as "text fields.") and how explicit the metadata is presented (e.g. having the composer information stored as a tuple such as "Composer: [name]" vs. specifying the print location of this information as top right corner in the engraved score.).

Getting the information from multiple sources and aggregating them.

The sources are MusicBrainz, **SymbTr**-txt, headers of the **SymbTr**-mu2 files and the **SymbTr**-slug

Show the header of a the kurdilihicazkar mu2 file

Obtained structured and linked data

`https://github.com/MTG/SymbTr/blob/master/symbTr_mbid.json`

makam, usul, form are cross validated accross MusicBrainz, txt files, mu2 headers and supplied symbtrnames.

4.2 Lyrics and Melody

The lyrics and synthetic melody extracted from each structural element is used to compute the relationships between the structures (Section 4.3). The synthetic melody is also the score feature used in audio-score alignment in Section 6.2. In Section 6.2, we additionally compute synthetic HPCPs and compare its "representativeness" with the synthetic melody. Figure 4.1 shows the lyrics and the synthetic melody (according to the theoretical in-

Table 4.1: Summary of the metadata related to a glsSymbTr score. The sources named as "txt, mu2, MB" and "slug" refer to the contents of the **SymbTr**-txt score, header of the **SymbTr**-mu2 score, MusicBrainz and **SymbTr**-slug ("makam–form–usul–name–composer").

Key	txt	mu2	MB	slug	Explanation
symbtr				✓	The symbtr-slug, if supplied or the filename
url			✓		The URL of the MusicBrainz work (https://musicbrainz.org/work/[mbid])
work		✓	✓	✓	Related work
composer		✓	✓	✓	Composer(s) related to the work
lyricist		✓	✓	✓	Lyricist(s) related to the work
makam	✓	✓	✓	✓	Makam of the work
form	✓	✓	✓	✓	Form of the work
usul	✓	✓	✓	✓	Usul of the work
number_of_notes	✓				Number of notes and rests in the **SymbTr**-txt score (annotation rows etc. are not counted)
rhythmic_structure	✓				List of rhythmic (e.g. tempo, usul) changes in the score
tempo	✓	✓			Nominal tempo of the piece. Read from the mu2 header, validated with the note durations in the **SymbTr**-txt
notation		✓			Shows whether a score is written in the classical accidentals ("TSM") or folk accidentals ("THM").
genre		✓			Indicates whether the composition belongs to the classical or the folk repertoire
key_signature		✓			Validated with the key signature of the dictionary, *symbtrdataextractor/makam_data/makam.json*
language			✓		The lyrics language, if the piece
recordings			✓		Recordings related to the work in MusicBrainz (optional)
scores					A list with a single **SymbTr**-score slug related to the work as there is a single score version per work in the current collection. The slug is obtained by referring to the *symbTr_mbid.json* dictionary in the **SymbTr** collection.
tonic					Symbol of the tonic note. Obtained by referring to the karar symbol of the makam of the composition in the dictionary, *symbtrdataextractor/makam_data/makam.json*

tervals defined in AEU theory) and the synthetic HPCPs extracted from an except of the **SymbTr**-score of the composition "Gel Güzelim."[1]

Table 4.2 summarizes the lyrics and melody-related features extracted from the score and their usage in different computational tasks.

[1] `https://github.com/MTG/SymbTr/blob/master/txt/nihavent--sarki--aksak--gel_guzelim--faiz_kapanci.txt`

Table 4.2: The features extracted from score fragments and their summary for each computational task they are used as input

Feature	Source	Task	Frame Rate	Explanation
Lyrics	**SymbTr**-txt	Lyrical relationship computation (Section 4.3.2)	N/A	
Synthetic Melody I	**SymbTr**-txt	Melodic relationship computation (Section 4.3.2)	Least common multiplier of the symbolic durations	
Synthetic Melody II	**SymbTr**-txt	Preliminary section linking experiments (Appendix B)	$44100/2048 \approx 46$ms	The tuning extracted from the audio recording to be aligned is used to generate the synthetic melody.
Synthetic Melody III	**SymbTr**-txt	All audio-score alignment tasks (Chapter 6)	$44100/2048 \approx 46$ms	The rest durations are added to the duration of the previous note (Figure 4.1d). The theoretical tuning defined in AEU theory are used to generate the synthetic melody.
Synthetic HPCPs	**SymbTr**-MIDI	Section Linking (Section 6.7.2)	$44100/2048 \approx 46$ms	The frame size is chosen as 4096 samples. They are computed with different number of bins per octave in the section linking experiments.

4.2.1 Lyrics

We use the information in the lyrics column to determine the boundaries of the vocal sections in Section 4.3.2. The lyrics associated with a sequence or an element x is a string denoted as $\boldsymbol{\lambda}^{(x)}$, simply obtained by contatenating the syllables of the note sequence $[\bar{n}_1^{(x)}, \ldots, \bar{n}_{|\bar{\mathbf{N}}(x)|}^{(x)}]$ of x. The editorial annotations (Section 3.1.2) and the whitespaces in the lyrics column are ignored in the process. Then the characters in the obtained string are all converted to lower case. Trivially, $\boldsymbol{\lambda}^{(\bar{n}_i)}$ of a note $\bar{n}_i$ is the syllable associated with the note $\bar{n}_i$ in the lyrics column.

4.2.2 Synthetic Melody

Given a fragment (x) in the **SymbTr**-txt score, the corresponding $\left\langle n_i^{(x)}, d\big(\bar{n}_i^{(x)}\big)\right\rangle$ tuples in the note sequence $\bar{\mathbf{N}}^{(x)}$ is selected. Here, the sum of the durations in the tuples $\sum_i d\big(\bar{n}_i^{(x)}\big)$ is equal to the duration of the score fragment $d(x)$. Then we note the makam of the composition, which is given in the score, and obtain the karar-symbol of the piece by checking the makam in the $\langle\, makam,\ karar\rangle$ dictionary (Section 3.2).

Then the note-symbols $n_i^{(x)}$ are mapped to the Holderian comma (Hc) distances. The mapping can be done according to the AEU theory (or any other music theory) with reference to the karar note. As an example see Figure 4.1: here the karar note is G4 (Nihavent makam) and all the notes take on values in relation to that karar, as for instance 13 Hc for the

B4♭. The intervals can also be mapped according to a tuning (Section 5.9) extracted from audio recording(s) (Şentürk, Holzapfel, & Serra, 2012). Finally, a synthetic predominant melody $\hat{\mathbf{\Psi}}^{(x)}$ is calculated by sampling the mapped notes relative to the their duration $d(\bar{n}_i^{(x)})$ at a certain frame rate and concatenating all samples (Figure 4.1c). In the score structure analysis (Section 4.3.2), the synthetic melody is sampled with a frame rate equal to the least common multiplier (LCM) of the symbolic score durations (e.g. if there are only dotted eighth and fourth notes in a score, the frame rate is a sixteenth note). In the audio-score alignment experiments presented in Chapter 6, the frame rate is selected as $2048/44100 = 46$ms (21.5 samples per second), which is equal to the frame rate of the predominant melody extracted from the audio recordings (Section 6.2).

In makam music practice, the notes preceding rests may be sustained for the duration of the rest.[2] For this reason, the rests in the score are ignored and their duration is added to the previous note (Figure 4.1d) in the synthetic melody computation step in the audio-score alignment experiments (Section 6.2).

4.2.3 Synthetic Harmonic Pitch Class Profiles

In the section-linking experiments that will be described in Section 6.7.2, **SymbTr** scores in MIDI format are used to obtain the synthetic HPCPs. Given a fragment (x) selected from a MIDI-score, audio is generated from the fragment. We use TiMidity++[3] with the default parameters for the audio synthesis. Since there are no standard SoundFonts of makam music instruments, we select the default SoundFont.[4] Nevertheless the SoundFont selection should not affect the HPCP computation greatly since HPCPs were reported to be robust to changes in timbre (Gómez, 2006). Then, the synthetic HPCPs, $\hat{\mathbf{\Omega}}^{(x)} = [\hat{\omega}_1^{(x)}, \ldots, \hat{\omega}^{(x)}_{|\hat{\mathbf{\Omega}}^{(x)}|}]$, are computed for the score fragment (x) (Figure 6.1e). We use Essentia in the computation (Bogdanov et al., 2013). We use the default parameters given in (Gómez, 2006). The hop size and the frame size are chosen to be 2048 (e.g. ~ 21.5 frames per second) and 4096 samples respectively. The first bin of the HPCPs is assigned to the tonic symbol, κ, (e.g.G4 for Nihavent makam). Note that the HPCPs contain microtonal information as well, since this information is encoded into the MIDI-scores (K. Karaosmanoğlu, 2012). HPCPs,

[2] Figure 6.1 in Chapter 6 shows the same score fragment in Figure 4.1 with a linked audio fragment. Notice that the notes in 6.1a are sustained in the performance as seen in the audio waveform in Figure 6.1b.

[3] http://timidity.sourceforge.net/

[4] grand acoustic piano: http://freepats.zenvoid.org/sf2/

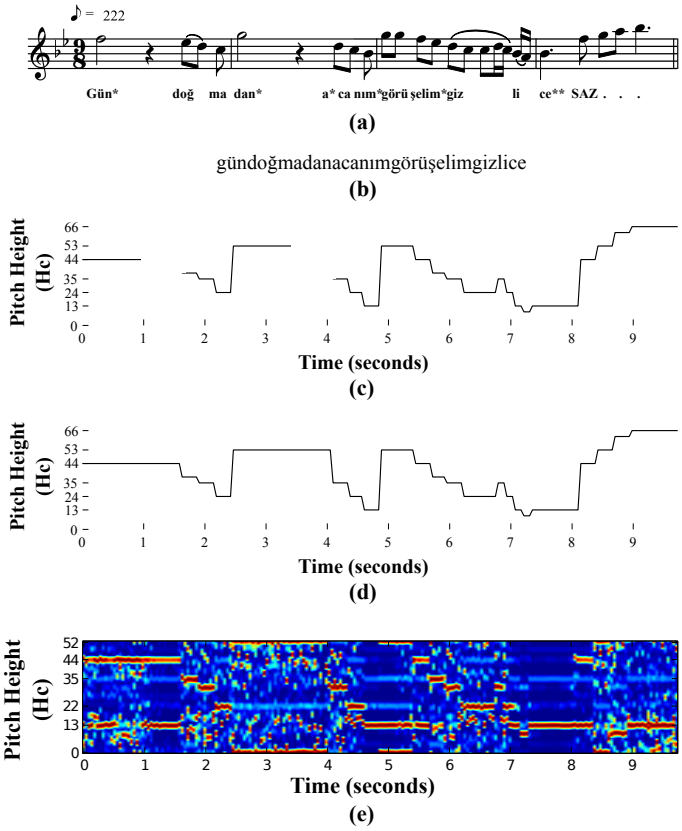


Figure 4.1: A short excerpt from the score of the composition, *Gel Güzelim*. **a)** The score, **b)** the lyrics, **c)** the synthetic melody computed from the note symbols and durations. The spaces in the end of the syllables are displayed as *s. **d)** the synthetic melody with the rest duration added to the duration of the previous note. **e)** the synthetic HPCPs

$\hat{\mathbf{\Omega}}^{(x)}$, are computed with different number of bins per octave in the section linking experiments (see Section 6.7.2).

4.3 Structure

Analyzing the structure of a music piece is integral in understanding how the musical events progress along with their functionality within the piece. Automatic extraction of the melodic and lyrical structures, as well as their roles within the composition, might be used to facilitate and enhance tasks such as digital music engraving, automatic form identification and analysis, audio-score and audio-lyrics alignment, music prediction and gener-

ation.

Structural analysis is a complex problem, which can be approached in different granularities such as sections, phrases and motifs (Pearce, Müllensiefen, & Wiggins, 2010). To find such groupings there has been many approaches based on music theory (Jackendoff, 1985), psychological findings and computational models (Cambouropoulos, 2001; Pearce et al., 2010). On the other hand, there are a few studies that has investigated automatic structural analysis of makam musics. Lartillot and Ayari (2009) has used computational models to segment Tunisian modal music and compared the segmentations with the annotations of the experts. Lartillot, Yazıcı, and Mungan (2013) has proposed a similar segmentation model for OTMM and also conducted comparative experiments between the automatic segmentations and human annotations. Due to the lack of musicological agreement on how to segment makam music scores, Bozkurt, Karaosmanoğlu, et al. (2014) focused on learning a model from a dataset of music scores annotated by experts and segmenting larger score datasets automatically using the learned model. They propose two novel culture-specific features based on the melodic and rhythmic properties of OTMM and conduct comparative studies with the features used in the state-of-the-art methods (Bod, 2002; Cambouropoulos, 2001; Temperley, 2004; Tenney & Polansky, 1980) and show that the proposed features improve the phrase segmentation performance.[5] These methods typically focus on finding the segment boundaries and do not study the inter-relations between the extracted segments.

4.3.1 Problem Definition

Given the note sequence $\bar{\mathbf{N}}^{(b)} := \left[\bar{n}_1^{(b)}, \bar{n}_2^{(b)}, \dots\right]$ and the measure sequence $\bar{\mathbf{M}}^{(b)} := \left[\bar{m}_1^{(b)}, \bar{m}_2^{(b)}, \dots\right]$ in the score (b), the aim is to extract the sections $\bar{\mathbf{S}}^{(b)} := \left[\bar{s}_1^{(b)}, \bar{s}_2^{(b)}, \dots\right]$ and the phrases $\bar{\mathbf{P}}^{(b)} := \left[\bar{p}_1^{(b)}, \bar{p}_2^{(b)}, \dots\right]$ along with their boundaries, and the melodic and lyrical relationship with other structural elements of the same type. Note that the sections and phrases will be collectively called as "structural elements" in the score throughout the Chapter. Moreover, the superscript (b) is omitted in the rest of the Chapter for the sake of simplicity.

It is assumed that the structural elements of the same type are non-overlapping and consecutive (e.g. the last note of a section is always adjacent to the first note of the next section). Consecutiveness restriction

[5] For a detailed review of structural analysis applied to OTMM and relevant state of the art the readers are referred to (Bozkurt, Karaosmanoğlu, et al., 2014) and (Pearce et al., 2010), respectively.

also implies that transitive interactions between two consecutive structural elements are not permitted. In the scores of the vocal compositions, each poetic line is considered as a section.

Remark that each subsequence[6] might cover or overlap with subsequences of different types, e.g. the note sequence in a section would be a subsequence of $\bar{\mathbf{N}}$ or a phrase might start in the middle of a measure and end in another. The index of the first note and the index of the last note in the note sequence $\bar{\mathbf{N}}^{(x)}$ of a score element x are denoted as $\bar{n}_1^{(x)}$ and $\bar{n}_{|\bar{\mathbf{N}}^{(x)}|}^{(x)}$ (where $|\bar{\mathbf{N}}^{(x)}|$ is the number of notes in $\bar{\mathbf{N}}^{(x)}$), respectively. For example, the first note of an arbitrary section $\bar{s}_i$, phrase $\bar{p}_j$ and measure $\bar{m}_k$ are denoted as $\bar{n}_1^{(\bar{s}_i)}$, $\bar{n}_1^{(\bar{p}_j)}$ and $\bar{n}_1^{(\bar{m}_k)}$, respectively. The lyrics associated with an arbitrary element x is denoted as $\boldsymbol{\lambda}^{(x)}$. Each $\bar{n}_j^{(b)}$ (where $j \in \left[1 : |\bar{\mathbf{N}}^{(b)}|\right]$) consists of a $\left\langle n_j^{(b)}, d\left(n_j^{(b)}\right), \boldsymbol{\lambda}^{\left(n_j^{(b)}\right)} \right\rangle$ tuple, the elements of which represent the note symbol, note duration and the syllable in the lyrics associated with the note.

4.3.2 Methodology

We first extract the section boundaries from the score using a *ad hoc*, heuristic process taking the editorial structure labels in the score as an initial reference. In parallel, the score is automatically segmented phrases according to a model learned from the phrases annotated by an expert. Next, the synthetic melody and the lyrics are extracted from each section and phrase. Then, a melodic and a lyrical similarity matrix are computed between the extracted phrases and the sections separately. A graph is formed from each similarity matrix and the relation between the structural elements in the context of the similarity (melodic or lyrical) is obtained. Finally semiotic labeling is applied to the computed relations (Şentürk & Serra, 2016b).

Section Extraction

The section boundaries are inferred using the explicit and implicit boundaries given in the lyrics column of the **SymbTr**-txt scores (Section 3.1.2). As a preprocessing step to distinguish the instrumental section labels from other editorial annotations in the lyrics column, the unique strings are extracted in the lyrics column of all **SymbTr** scores. We only keep the

[6] or element, which can also be regarded as a subsequence composed of a single element.

strings, which are written in capital letters and obtain the set of all editorial annotations in the **SymbTr**-scores. Then, the section annotations are picked manually.[7]

Given the score (b), the set of instrumental section names are searched in the lyrics column. The matched note indices mark the actual beginning $\bar{n}_1^{(\bar{s}_i)}$s of the instrumental sections $\bar{s}_i \in \bar{\mathbf{S}} \,|\, \boldsymbol{\lambda}^{(\bar{s}_i)} = \varnothing$. Next, the lyrics column is searched for syllables ending with double spaces. The index of the matched notes are assigned to the final note $\bar{n}^{(\bar{s}_i)}_{\left|\bar{\mathbf{N}}^{(\bar{s}_i)}\right|}$s of the vocal sections; $\bar{s}_i \in \bar{\mathbf{S}} \,|\, \boldsymbol{\lambda}^{(\bar{s}_i)} \neq \varnothing$. As explained in Section 3.1.2, the indices $\bar{n}^{(\bar{s}_i)}_{\left|\bar{\mathbf{N}}^{(\bar{s}_i)}\right|}$s may not coincide with the actual ending and it may be moved to a subsequent note.

Up to here, the section sequence $\bar{\mathbf{S}} := [\bar{s}_1, \bar{s}_2, \ldots, \bar{s}_{|\bar{\mathbf{S}}|}]$ has been found, where $|\bar{\mathbf{S}}|$ is the total number of sections in the music score (b). The first note of the vocal sections and the last note of the instrumental sections are unassigned at this stage. The section boundaries are located using a rule-based scheme iterating though all sections starting from the last one.

If a section $\bar{s}_i$ is instrumental, then $\bar{n}_1^{(\bar{s}_i)}$ is already assigned. If a section $\bar{s}_i$ is vocal and the previous section $\bar{s}_{i-1}$ is instrumental, the last instrumental measure is found, $\bar{m}_k \in \bar{\mathbf{M}} \,|\, \boldsymbol{\lambda}^{(\bar{m}_k)} = \varnothing$, before the last note $\bar{n}^{(\bar{s}_i)}_{\left|\bar{\mathbf{N}}^{(\bar{s}_i)}\right|}$ of the section $\bar{s}_i$. We then assign the first note $\bar{n}_1^{(\bar{s}_i)}$ to the first note $\bar{n}_1^{(\bar{m}_{k+1})}$ of the next measure $\bar{m}_{k+1}$. If both the current section $\bar{s}_i$ and the previous section $\bar{s}_{i-1}$ are vocal, $\bar{n}_1^{(\bar{s}_i)}$ is assigned to the index of the first note with lyrics after the last note $\bar{n}^{(\bar{s}_{i-1})}_{\left|\bar{\mathbf{N}}^{(\bar{s}_{i-1})}\right|}$ of $\bar{s}_{i-1}$. If $\bar{n}_1^{(\bar{s}_i)}$ and $\bar{n}^{(\bar{s}_{i-1})}_{\left|\bar{\mathbf{N}}(\bar{s}_{i-1})\right|}$ are not in the same measure, $\bar{n}_1^{(\bar{s}_i)}$ is reassigned to the first note of its measure, i.e. $\bar{n}_1^{(\bar{m}_k)} \mid \bar{n}_1^{(\bar{s}_i)} \in \bar{m}_k$. Finally the last note $\bar{n}^{(\bar{s}_i)}_{\left|\bar{\mathbf{N}}^{(\bar{s}_i)}\right|}$ of the section is moved to the index of the first note $\bar{n}_1^{(\bar{s}_{i+1})}$ of the next section $\bar{s}_{i+1}$ minus one.

The pseudocode of the procedure is given in Algorithm 1. Note that the start of the first section and the end of the final section are assigned to 1 and $|N|$, respectively, where $|N|$ is the number of notes in the score. This detail omitted from the pseudocode for the sake of brevity.

Having located the boundaries, the sections are extracted by simply taking all information (i.e. rows in the **SymbTr**-txt score) between these

[7] https://github.com/sertansenturk/symbtrdataextractor/blob/master/symbtrdataextractor/makam_data/symbTrLabels.json

Algorithm 1 Locate section boundaries

for $i := |\bar{\mathbf{S}}| \rightarrow 1$ **do** ▷ from the last index to the first
  if $\boldsymbol{\lambda}^{(\bar{s}_i)} \neq \varnothing$ **then** ▷ find the start of the vocal section
    if $\boldsymbol{\lambda}^{(\bar{s}_{i-1})} = \varnothing$ **then** ▷ previous section is instrumental
      $k \leftarrow arg_k min\big(\bar{n}_1^{(\bar{m}_k)}$ **is after** $\bar{n}_1^{(\bar{s}_{i-1})} \wedge$
        $\boldsymbol{\lambda}^{(\bar{m}_k)} \neq \varnothing\big)$
      $\bar{n}_1^{(\bar{s}_i)} \leftarrow \bar{n}_1^{(\bar{m}_k)}$
    else ▷ previous section is vocal
      $k \leftarrow arg_k min\big(n_k$ **is after** $\bar{n}^{(\bar{s}_{i-1})}_{|\bar{\mathbf{N}}^{(\bar{s}_{i-1})}|} \wedge$
        $\boldsymbol{\lambda}^{(\bar{n}_k)} \neq \varnothing\big)$
      if $\bar{n}_1(\bar{s}_i) \in \bar{m}_k \ \wedge \ \bar{n}^{(\bar{s}_{i-1})}_{|\bar{\mathbf{N}}^{(\bar{s}_{i-1})}|} \notin \bar{m}_k$ **then**
        $\bar{n}_1^{(\bar{s}_i)} \leftarrow \bar{n}_1^{(\bar{m}_k)}$
      else
        $\bar{n}_1^{(\bar{s}_i)} \leftarrow n_k$
  $\bar{n}^{(\bar{s}_i)}_{|\bar{\mathbf{N}}^{(\bar{s}_i)}|} \leftarrow \bar{n}_1^{(\bar{s}_{i+1})} - 1$ ▷ sections are consecutive

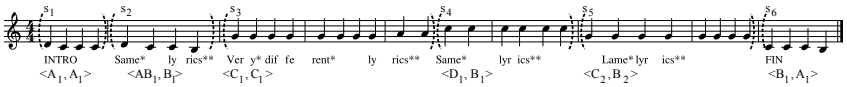


Figure 4.2: Section analysis applied to a mock example. The section labels ("INTRO" and "FIN") are given in the lyrics written in capital letters, The spaces in the end of the syllables are visualized as *. The semiotic $<$ *Melody*, *Lyrics* $>$ label tuples of each section are shown below the lyrics. The similarity threshold in the similar clique computation step is selected as 0.7 for both melody and lyrics.

note boundaries. Figure 4.2 shows the section boundaries obtained on a mock example.

Automatic Phrase Segmentation

We use the automatic phrase segmentation methodology has been proposed by Bozkurt, Karaosmanoğlu, et al. (2014), why XX. The source code and the training dataset (M. K. Karaosmanoğlu, Bozkurt, Holzapfel, & Doğrusöz Dişiaçık, 2014) are open and available online.[8]

In order to train the segmentation model, the annotations of Expert 1, who annotated all the 488 scores in the training dataset, are used (M. K. Karaos-

[8] http://www.rhythmos.org/shareddata/turkishphrases.html

manoğlu et al., 2014). Moreover, the authors of (M. K. Karaosmanoğlu et al., 2014) commented through personal communication that the first expert's annotations are more consistent with each other. There are a total of 20801 training phrases annotated by the first expert. In the training dataset, the variants of some usuls are combined (e.g. the scores in Kapalı curcuna usul is treated as Curcuna. Using the trained model, automatic phrase segmentation is applied to the score collection (Section 3.1.2) and the phrase boundaries $\bar{n}_1^{(\bar{p}_k)}$ and $\bar{n}_{|\bar{\mathbf{N}}^{(\bar{p}_k)}|}^{(\bar{p}_k)}$ for each phrase $\bar{p}_k \in P := [\bar{p}_1, \bar{p}_2, \dots]$ is obtained, where P is the automatically extracted phrase sequence. In Figure 4.4, the vertical red and purple lines shows the phrase boundaries extracted from the score "Kimseye Etmem Şikayet."[9]

Melodic and Lyrical relationship computation

Given the structure sequence $F := [f_1, f_2, \dots]$ (which is either the section sequence $\bar{\mathbf{S}}$ or the phrase sequence $\bar{\mathbf{P}}$) extracted from the score, first the synthetic melody is generated and the lyrics of each structural element is extracted (Section 4.2.1). The sampling rate of the synthetic melody is taken as the `LCM` of the symbolic score durations (e.g. if there are only dotted eighth and fourth notes in a score, the sampling rate is a sixteenth note). This sampling rate allows to represent the melody discretely without introducing any ambiguity in time using the least number of samples possible. Then, a melodic similarity and lyrical similarity between each element is computed using a similarity measure based on Levenshtein distance (Levenshtein, 1966). The similarity measure $\hat{\mathscr{L}}(x, y)$ is defined as:

$$\hat{\mathscr{L}}(x, y) := 1 - \frac{\mathscr{L}(x, y)}{max\left(|x|, |y|\right)} \tag{4.1}$$

where $\mathscr{L}(x, y)$ is the Levenshtein distance between the two "strings" x and y with the lengths $|x|$ and $|y|$, respectively and $max()$ denotes the maximum operation. In our case, x and y are the synthetic melody or the lyrics of two structural elements. The similarity yields a result between 0 and 1. If the strings of the compared structural elements are exactly the same, the similarity will be one. Similar strings (e.g. the melodies of two instances of the same section with volta brackets) will also output a high similarity.

[9] https://github.com/MTG/SymbTr/blob/a50a16ab4aa2f30a278611f333ac446737c5a877/txt/nihavent--sarki--kapali_curcuna--kimseye_etmem--kemani_sarkis_efendi.txt

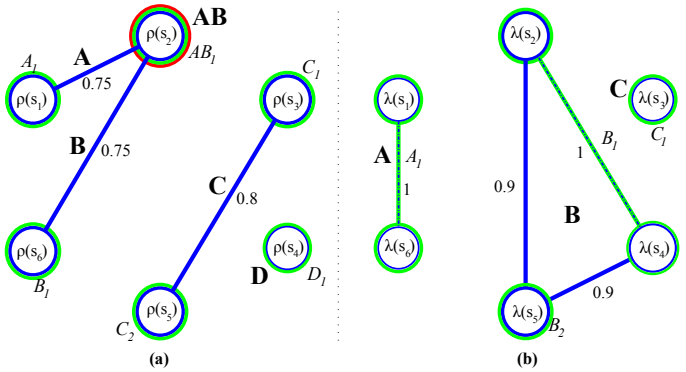


Figure 4.3: The graphs, the cliques and the semiotic labels obtained from the mock example (Figure 4.2) using an edge weight threshold of 0.7 for both melody and lyrics. The circles represent the nodes and the lines represent the edges of the graphs, respectively. The edge weights are shown next to the lines. Green, blue and red colors represent the unique cliques, the similar cliques and the intersection of similar cliques, respectively. The semiotic label of each similar clique and each intersection is shown in bold and the semiotic label of each unique clique is shown in italic, respectively.

From the melodic and lyrical similarities, two separate graphs are built, in which the nodes are the structural elements and the elements are connected to each other with undirected edges. The weight of an edge connecting two structural elements f_i and f_j is equal to $\hat{\mathscr{L}}\left(\hat{\mathbf{\Psi}}^{(f_i)}, \hat{\mathbf{\Psi}}^{(f_j)}\right)$ in the melodic relation graph and $\hat{\mathscr{L}}\left(\boldsymbol{\lambda}^{(f_i)}, \boldsymbol{\lambda}^{(f_j)}\right)$ in the lyrics relation graph, respectively. Next, the edges are removed with a weight less than a constant similarity threshold $\mathscr{L} \in [0, 1]$. In Section 4.3.3, the effect of using different $\mathscr{L}$ values will be experimented.

Given the graph, the groups of structural elements having similar strings are obtained by finding the maximal cliques in the graph (Tomita, Tanaka, & Takahashi, 2006). A maximal clique is a subgraph, which has its each node connected to each other and it cannot be extended by including another node. These cliques are denoted as $v_j \in \boldsymbol{\mathcal{V}}$, where $\boldsymbol{\mathcal{V}}$ is the set of "similar cliques." The maximal cliques of the graph are additionally computed only considering the edges with zero weight. These cliques show us the groups of structural elements, which have exactly the same string. Each of these cliques are called as "unique clique" $u_k \in \boldsymbol{\mathcal{U}}$, where $\boldsymbol{\mathcal{U}}$ is the set of the 'unique clique. Note that two or more similar cliques can intersect with each other. Such an intersection resembles all the relevant similar cliques. These "intersections" are denoted as $w_l \in \boldsymbol{\mathcal{W}}$, where

$\boldsymbol{\mathcal{W}}$ is the set of intersections between different similar cliques. Also, $\eta(x)$ denotes the nodes of an arbitrary graph x. Remark that:

- A unique clique is a subgraph of at least one similar clique, i.e. $\forall u_k \in \boldsymbol{\mathcal{U}}, \exists v_j \in \boldsymbol{\mathcal{V}} \mid \eta(u_k) \subseteq \eta(v_j)$.
- A unique clique cannot be a subgraph of more than one intersection, i.e. $\forall u_k \in \boldsymbol{\mathcal{U}}, \nexists \{w_l, w_m\} \subseteq \boldsymbol{\mathcal{W}} \mid \eta(u_k) \subseteq \eta(w_l) \wedge \eta(u_k) \subseteq \eta(w_m)$.
- A structural element belongs to only a single unique clique, i.e. $\forall f_i \in F, \exists! u_k \in \boldsymbol{\mathcal{U}} \mid \eta(f_i) \subseteq \eta(u_k)$.

Figure 4.3 shows the graphs computed from the sections of the mock example introduced in Figure 4.2. In the melodic relations graph, each section forms a unique clique since the melody of each section is not exactly the same with each other. Using a similiarty threshold of 0.7, four similar cliques are obtained formed by $\{\bar{s}_1, \bar{s}_2\}$, $\{\bar{s}_2, \bar{s}_6\}$, $\{\bar{s}_3, \bar{s}_5\}$, $\{\bar{s}_4\}$. Notice that $\{\bar{s}_4\}$ is not connected to any clique. so it forms both a unique and a similar clique. Moreover, $\bar{s}_2$ is a member of both the first and the second similar cliques and hence it is the intersection of these two cliques. For the lyrics, there are four unique cliques, formed by the sections $\{\bar{s}_1, \bar{s}_6\}$ (aka. instrumental sections), $\{\bar{s}_2, \bar{s}_4\}$, $\{\bar{s}_3\}$ and $\{\bar{s}_5\}$. The lyrics of $\bar{s}_5$ is very similar to $\{\bar{s}_2, \bar{s}_4\}$ and they form a similar clique composed of these three nodes and the relevant edges.

Semiotic Labeling

After forming the cliques, semiotic labeling explained in (Bimbot, Deruty, Sargent, & Vincent, 2012) is used to describe the structural elements. First similar cliques are labeled with a base letter ("A", "B", "C", ...). Then the intersections are labeled by concetanating the base letters of the relevant similar cliques (e.g. "AB", "BDE", ...). Each unique clique is finally labeled with the label of the relevant intersection, if exists and with respect to the relevant similar clique otherwise, plus a number according to the occurence order of the clique in the score. Right now, only the simple labels termed by Bimbot et al. (2012) (e.g. "A_1", "A_2", "AB_2") are used to label the unique cliques.

The pseudocode of the process is given in Algorithm 2. During labeling, $\boldsymbol{\mathcal{U}}$, $\boldsymbol{\mathcal{V}}$ and $\boldsymbol{\mathcal{W}}$ are enumerated by sorting the elements with respect to the index of the first occurence each element in the score. The semiotic melody and lyrics label of an arbitrary element x are denoted as $\Lambda_{mel}^{(x)}$ and $\Lambda_{lyr}^{(x)}$, respectively. In the algorithm, the iterators $\#(v_j)$ for each similar clique v_j and $\#(w_l)$ for each each intersection w_l are used to assign the

numerical index to each unique clique $u_k \in \boldsymbol{\mathcal{U}}$ according its relation with the relevant similar clique or intersection.

Algorithm 2 Semiotic labeling

$@ \leftarrow$"A" ▷ Start the base letter iterator from "A"
$\#(v_j) \leftarrow 1, \forall v_j \in \boldsymbol{\mathcal{V}}$ ▷ Initialize the num. iterators for all v_j
$\#(w_l) \leftarrow 1, \forall w_l \in \boldsymbol{\mathcal{W}}$ ▷ Initialize the num. iterators for all w_l
for $v_j \in sort(\boldsymbol{\mathcal{V}})$ **do** ▷ Label similar cliques
  $\Lambda^{(v_j)} \leftarrow @$
  increment $@$ ▷ "A" $\Rightarrow$ "B"
for $w_l \in sort(\boldsymbol{\mathcal{W}})$ **do** ▷ Label intersections
  $\Lambda^{(w_l)} \leftarrow$ concatenate $\Lambda^{(v_j)}, \forall (v_j) \mid \eta(w_l) \subseteq \eta(v_j)$
for $u_k \in sort(\boldsymbol{\mathcal{U}})$ **do** ▷ Label unique cliques
  if $\exists w_l \mid \eta(u_k) \subseteq \eta(w_l)$ **then**
    $\Lambda^{(u_k)} \leftarrow \Lambda^{(w_l)}_{\#(w_l)}$ ▷ e.g. "ACD_1"
    $\#(w_l) \leftarrow \#(w_l) + 1$
  else
    $\Lambda^{(u_k)} \leftarrow \Lambda^{(v_j)}_{\#(v_j)} \mid \eta(u_k) \subseteq \eta(v_j)$ ▷ e.g. "C_2"
    $\#(v_j) \leftarrow \#(v_j) + 1$
for $f_i \in F$ **do** ▷ Label structural elements
  $\Lambda^{(f_i)} \leftarrow \Lambda^{(u_k)} \mid \eta(f_i) \subseteq \eta(u_k)$

The label of each section of the mock example is shown below the staff in Figure 4.2. The same semiotic labels are also shown on the computed graphs in Figure 4.3. Notice that the melodic semiotic label of $\bar{s}_6$ is B_1 because the first occurence of the relevant similar clique is at $\bar{s}_2$.

By extracting the relations in the graphs computed from the melodic and lyrics similarity matrices (Section 4.3.2) and then applying semiotic labeling to each section and phrase according to its relation, a $< \Lambda_{mel}, \Lambda_{lyr} >$ tuple is obtained for each section and phrase (Section 4.3.2). For each phrase the sections are also marked, which enclose and/or overlap with the phrase.

Figure 4.4 shows the results of the structural analysis applied to the score "Kimseye Etmem Şikayet." The sections are displayed in colored boxes with the volta brackets colored with a darker shade of the same color. The section labels and their semiotic $< \Lambda_{mel}, \Lambda_{lyr} >$ label tuple is shown on the left. The phrase boundaries are shown as red lines for the first and as purple for the second pass. The phrases and their semiotic labels are shown on top of the relevant interval and on the bottom, when there are differences in the boundaries in the second pass. Note that

$\bar{s}_5, \bar{s}_6, \bar{s}_9$ and $\bar{s}_{10}$ are the repetitive poetic lines (tr: "Nakarat"). "[Son]" in the end of the "Nakarat" marks the end of the piece. The similarity threshold is taken as 0.7 for both melody and lyrics. The usul of the score is Kapalı curcuna, which is treated as Curcuna in the phrase segmentation step (Section 4.3.2). Further examination of the analysis is left to the readers as an exercise.

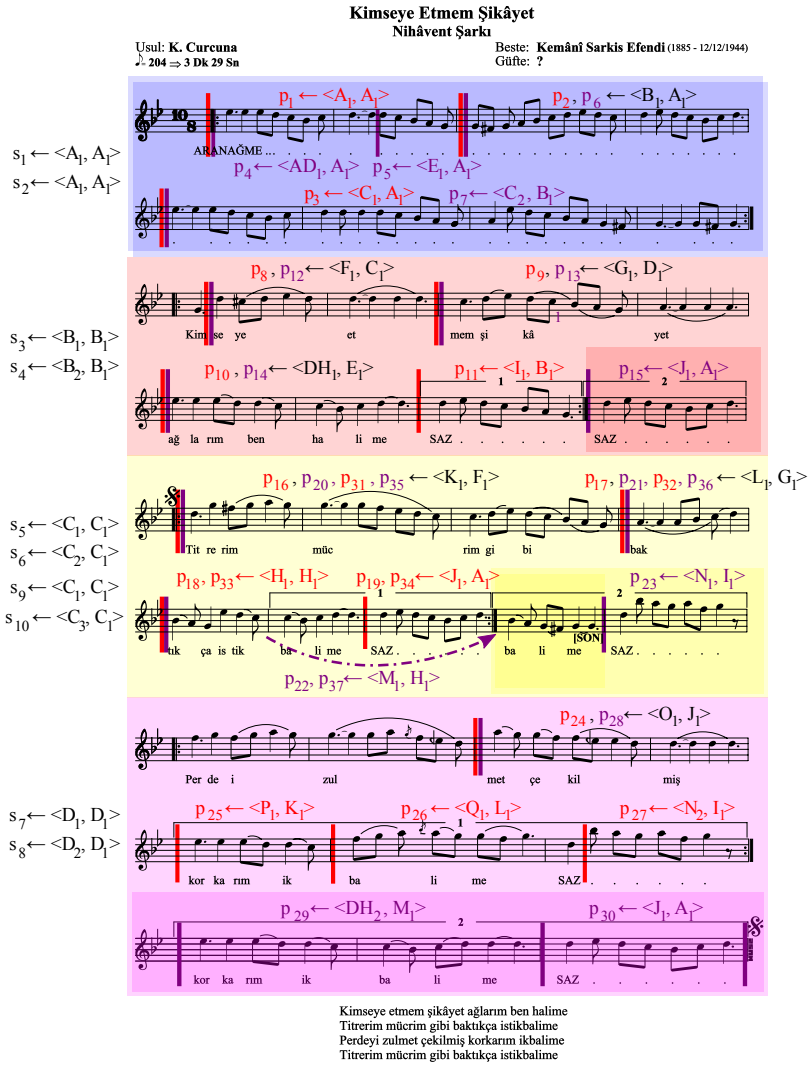


Figure 4.4: The results of the automatic structural analysis of the score "Kimseye Etmem Şikayet."

4.3.3 Experiments

Bozkurt, Karaosmanoğlu, et al. (2014) report the evaluation of the phrase segmentation method (Section 4.3.2) on an earlier and slightly smaller version of the annotations that is used to compute the segmentation model. The readers are referred to Bozkurt, Karaosmanoğlu, et al. (2014) for the evaluation of the training data. Furthermore, the labels of the automatic phrase segmentations need to be validated by musicologists parallel to the discussions brought by (Bozkurt, Karaosmanoğlu, et al., 2014). For this reason, investigation of the effects of the similarity threshold $\mathscr{L}$ in phrase analysis is left as future research.

To observe the effect of the similarity threshold in the melodic and lyrical relationship extraction (Section 4.3.2), a small dataset from the **SymbTr** collection is collected. The test dataset consists of 23 vocal compositions in the şarkı form and 42 instrumental compositions in peşrev and sazsemaisi forms. These three forms are the most common forms of the classical OTMM repertoire. Moreover their sections are well-defined within the music theory; the two instrumental forms typically consists of four distinct "hane"s and a "teslim" section, which follow a verse-refrain-like structure; the sections of the şarkıs typically coincide with the poetic lines. The experiments on şarkıs are focused on the ones with the poetic organization "zemin, nakarat, meyan, nakarat," which is one of the most common poetic organization observed in the şarkı form. Using the automatically extracted section boundaries (Section 4.3.2) as the ground-truth, I have manually labeled the sections in the scores with the same naming convention explained in Section 4.3.2.[10] Due to lack of data and concerns regarding subjectivity, the evaluation of section boundaries is left as future research.

We have conducted section analysis experiments on the test dataset by varying the similarity threshold from 0 to 1 with a step size of 0.05. After the section labels are obtained, the semiotic melody and lyrics labels are compared with the annotated labels. An automatic label is considered as "True," if it is exactly the same with the annotated label and "False," otherwise. For each score, the labeling accuracy is computed for the melody and the lyrics separately by dividing number of correctly identified (melody or lyrics) labels with the total number of sections. Additionally, the number of similar cliques and its ratio to the unique cliques obtained for each score is noted. For each experiment, the average accuracy for the similarity threshold $\mathscr{L}$ is computed by taking the mean of the

[10] The experiments and results are available at https://github.com/sertansenturk/otmm-score-structure-experiments/releases/tag/fma_2016

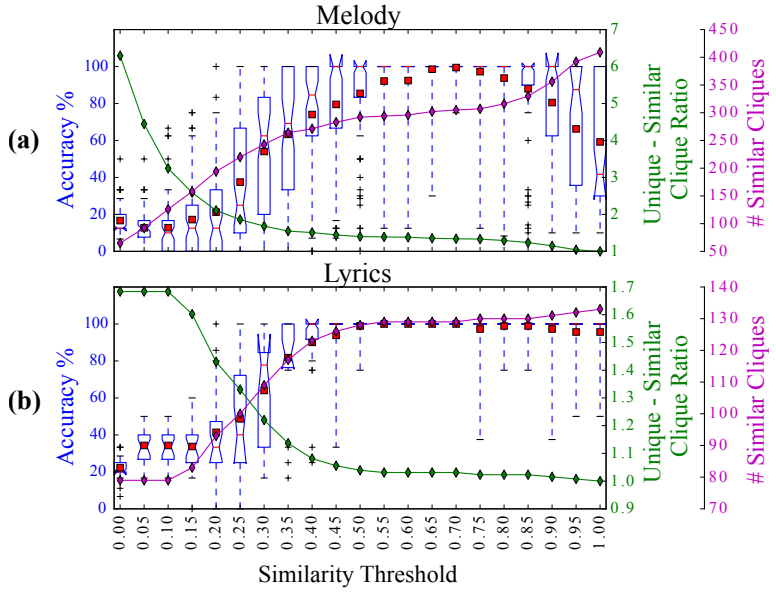


Figure 4.5: The notched boxplots of the accuracies, number of similar cliques and the ratio between the number of unique cliques and similar cliques obtained for **a)** the melody labels and **b)** the lyrics labels (only for vocal compositions) using different similarity thresholds. The squares in the boxplots denote the mean accuracy.

accuracies obtained from each score.

Figure 4.5 shows the notched boxplots of the accuracies, the total number of similar cliques and the ratio between the number of unique cliques and the number of similar cliques obtained for each similarity threshold. For the melody labels, the best results are obtained for the similarity threshold values between 0.55 and 0.80 and the best accuracy is 99%, when l is selected as 0.70. For lyrics labeling, any similarity value above 0.35 yields near perfect results and 100% accuracy is obtained for all the values of l between 0.55 and 0.70. In parallel, the number of similar cliques and the ratio between the unique cliques and the similar cliques gets flat in these regions. From these results, the optimal l as 0.70 is selected for both melodic and lyrical similarity.

4.3.4 Discussion

As shown in Section 4.3.3, the similarity threshold l has a direct impact on the structure labels. A high threshold might cause most of the similar structural elements regarded as different, whereas a low threshold would result in many differences in the structure disregarded. In this sense the

extreme values of l (around 0 or 1), would not provide any meaningful information as $l = 0$ would result in all the structures being labeled similar and $l = 1$ would be output all the structures as unique. We also observe that the melodic similarity is more sensitive to value of l than lyrics similarity. This is expected as the strings that make up the lyrics are typically more diverse than the note symbols used to generate the synthetic melody. In our experiments the optimal value of l is found as 0.7 for the small score dataset of compositions in the peşrev, sazsemaisi and şarkı forms. Moreover, it is observed that the curves representing the number of similar cliques and the ratio between the unique cliques and the similar cliques are relatively flat around the same l value, where obtain the best results are obtained (Figure 4.5). This implies that there is a correlation between decisions of the annotator and our methodology.

Nevertheless, the optimal l value presented above should not be considered as a general optimal. First of all, the sections were annotated by a single person and therefore our evaluation does not factor in the subjectivity between different annotators. Second, the section divisions in different forms are much different from the studied forms, which might influence the structure similarity. For example, many vocal compositions of OTMM with terennüms (repeated words with or without meaning such as "dost," "aman," "ey") are expected to have a lower optimal similarity threshold in the lyrics relationship computation step. Moreover the poetic lines might not coincide with melodic sections in many vocal compositions especially in folk music genre. Third, the threshold can be different in different granularities. For example, the phrases are much shorter than the sections as can be seen in Figure 4.4. The human annotators might perceive the intra-similarity between sections and phrases differently.

4.3.5 Summary

We proposed a method to automatically analyze the melodic and lyrical organization of the music score of OTMM. We applied the method on the latest release of the **SymbTr** collection. We extracted 49259 phrases from 1345 scores and 21569 sections from 1771 scores. We are also using the extracted structural information in automatic score validation, score engraving and audio-score alignment tasks.

In the future, I would like to test other string matching and dynamic programming algorithms (Serrà et al., 2009; Şentürk, Holzapfel, & Serra, 2014) in general, for similarity measures with different constraints and select the optimal similarity threshold l automatically according to the melodic and lyrical characteristics of the data. We would also like to solidify our findings by working on a bigger dataset annotated by multi-

ple experts and cross-comparing the annotated and the automatically extracted boundaries as done in (Bozkurt, Karaosmanoğlu, et al., 2014). Our ultimate aim is to develop methodologies, which are able to describe the musical structure of many music scores and audio recordings semantically and on different levels.

XX

4.4 Music Score Format Conversion

We have developed tools in Python to convert the **SymbTr**-txt scores to the MusicXML format[11] and then to the LilyPond format[12] to improve the accesibility of the collection from popular music notation and engraving software.[13] The converters use the information obtained from *symbtrdataextractor* to add the metadata and the section names into the converted scores.

Moreover, the row index of each note (and rest) in the **SymbTr**-txt files are stored in the converted files as inline comments. The scores in the LilyPond format are later converted to Scalable Vector Graphics Format (SVG) using the LilyPond software itself. The software divides the score into multiple files during the conversion. As a postprocessing step, these files are joined together. Similar to the previous steps in the score conversion, the note indices are also stored. These indices are used later to visualize the audio-score alignment results (Section 6.8) synbchronous to the audio playback in Dunya (Section 7.1.1).

The files in the MusicXML format are hosted in the **SymbTr** github repository and updated with each release.[14] The scores in the latest release is also hosted in Dunya-makam along with the score file in the SVG format. Independently, the txt files in the **SymbTr** releases are converted to abc notation[15] by Seymour Shilen.[16]

[11] `https://github.com/burakuyar/MusicXMLConverter`

[12] `https://github.com/hsercanatli/makam-musicxml2lilypond`

[13] MusicXMLConverter and makam-musicxml2lilypond packages are mainly realized by Burak Uyar and Hasan Sercan Atlı, respectively, within their masters research under the advisorship of Barış Bozkurt. My contributions to these packages are mostly related to bug fixing, refactoring, documentation and deployment. As of August 2016, the number of lines manipulated by me is approximately the same with the main developers in both packages.

[14] `https://github.com/MTG/SymbTr/tree/master/MusicXML`

[15] `http://abcnotation.com/`

[16] `http://ifdo.ca/~seymour/runabc/makams/index.html`

4.5 Implementation and Applications

The metadata extraction and score analysis is implemented in Python and hosted in the *symbtrdataextractor* package[17]. The package is able to parse the headers of the **SymbTr**-mu2 files, process the contents of the **SymbTr**-txt files and also crawl the relevant metadata from MusicBrainz. To crawl MusicBrainz, a wrapper around the open-source *Musicbrainz NGS bindings*[18] is created, which is specialized to fetch the OTMM related metadata.[19] We have also forked the open automatic phrase segmentation package by (Bozkurt, Karaosmanoğlu, et al., 2014),[20] which is written in MATLAB scripting language. The fork modularizes the code and packages it into a standalone binary so it can be integrated to other tools without the need of a MATLAB proprietary license. Moreover, the code is optimized such that it performs considerably faster than the original code.

tomatoXX

Block diagram of tomato XX

For a further details about the implementation, please refer to Appendix D.1.

We have been using the extracted information in several applications as summarized below.

4.5.1 Score collection analysis

Using the optimal similarity threshold ($l = 0.7$), structural analysis (Section 4.3) is applied on the latest release of the **SymbTr** collection (Section 3.1.2). 49259 phrases are extracted and labeled from 1345 scores, which have both their makam and usul covered in the phrase segmentation training model. Because there is no training data for the usul variants "Yürüksemai II", "Devrihindi II", "Müsemmen II", "Raksaksağı II", "Devrituran II" and "Kapalı Curcuna," they are treated as the most common variant of the same usul, namely "Yürüksemai", "Devrihindi", "Müsemmen", "Raksaksağı", "Devrituran" and Curcuna, respectively. In parallel, 21569 sections are extracted from 1771 scores.[21] The data can be further used to study the structure of musical forms of OTMM.

[17] https://github.com/sertansenturk/symbtrdataextractor/

[18] https://github.com/alastair/python-musicbrainzngs

[19] https://github.com/sertansenturk/makammusicbrainz

[20] https://github.com/MTG/makam-symbolic-phrase-segmentation

[21] The data is available at https://github.com/sertansenturk/turkish_makam_corpus_stats/tree/66248231e4835138379ddeac970eabf7dad2c7f8/data/SymbTrData

4.5.2 Automatic score validation

Structural analysis, along with the other functionalities of the *symbtr-dataextractor* package are used in unittests applied to **SymbTr** collection in a continuous integration scheme to automatically validate the contents of the scores.[22]

encoding and line-breaks differences in version, e.g. missing usul rows content validity, e.g. grace notes have a duration of 0 seconds and the offset follows the durations other??

SymbTr-extras can be used to fix these issues, automate attribute name changes and also to convert the score to MusicXML with all relevant metadata semi-automatically.

4.5.3 Audio-score alignment

In the performances of OTMM compositions, the musicians occasionally insert, repeat and omit sections. Moreover they may introduce musical passages, which are not related to the composition (e.g. improvisations). In Şentürk, Holzapfel, and Serra (2014), we have proposed a section-level audio-score alignment methodology proposed for OTMM, which considers such structural differences. In the original methodology the sections in the score are manually annotated with respect to the melodic structure. Next, the candidate time intervals in the audio recording are found for each section using partial subsequence alignment. The manual section annotation step is replaced with the automatic section analysis part of our alignmet method, where we use the melody labels to align relevant audio recordings and music scores. Using the modified method we have aligned the related audio and score pairs in the CompMusic Turkish makam music corpus (Uyar et al., 2014) and linked 18.770 sections performed in 1767 pairs of audio recordings and music scores. The aligned audio-score pairs are accessible via *Dunya makam*, our prototype web application for the discovery of OTMM (Şentürk et al., 2015).[23] In the application, the audio can be listened synchronous to the related music score(s) on the note-level and the sections are displayed on the audio timeline.

We have additionally conducted experiments using the melodic relations of the extracted phrases. Our preliminary results suggest that phrase-level alignment may provide better results than section-level alignment.

[22] https://travis-ci.org/MTG/SymbTr/

[23] http://dunya.compmusic.upf.edu/makam

4.6 Conclusion

XX

Chapter 5

Audio Analysis

The audio recordings can provide information about the characteristics (e.g. in terms of dynamics or timing) of an interpretation of a particular piece. Automatic analysis XX would facilitate the curation and description of large audio corpora not only by greatly reducing the time and effort spent on manual annotations but also providing automatically extracted, reliable information, which would be too difficult or time-consuming for human annotators.

Given an audio fragment (a), we fetch the structured metadata available in MusicBrainz (Section 5.1). We extract the predominant melody (Section 5.2), pitch distribution, pitch-class distribution (Section 5.5), stable pitches, stable pitch-classes (Section 5.6), makam (Section 5.7), karar (Section 5.7), transposition (Section 5.8), tuning (Section 5.9) and melodic progression (Section 5.10) of the audio recording.[1]

Explain implementation and overall analysis

There exists several audio analysis libraries. Explain Marsyas, Essentia, Tarsos DSPXX. However, they do not have most of the implementations of the culture-specific methodologies applied to OTMM. They also don't work optimal for OTMM. This necesitates tools specialized to analyse and XX OTMM. Another aim of such a tool is to adapt solutions of relevant problems applied to other music cultures, improve existing state-of-the-art and integrate the aforamentioned methodologies into a workflow, which is able to analyse the audio recordings of OTMM in a flexible and completely automatic way.

Aims:XX

ContributionsXX:

[1] Note that the superscript (a) will be omitted throughout this chapter (e.g. the predominant melody of (a), $\boldsymbol{\varrho}^{(a)}$ will be printed as $\boldsymbol{\varrho}$) for the sake of simplicity except the cases in which multiple audio recordings are processed (e.g. Section 5.7.2).

Table 5.1: Summary of the metadata obtained for an audio recording

Source	Key	Explanation
Audio file	path	Location of the audio file
	duration	Duration of the audio file (in seconds)
	bit_rate	Bit rate of the audio file
	sampling_frequency	Sampling frequency of the audio file
	mbid	MusicBrainz Identifier
	url	URL in MusicBrainz (basically, http://musicbrainz.org/recording/[mbid])
MusicBrainz	title	Title of the recording in MusicBrainz
	releases	List of releases which the recording is part of
	works	List of works performed in the recording
	artist-credits	List of main credited artist(s)
	artists	List of artist relationships; vocalists, intrument performers, conductors, recording enginners, etc. who took part in realizing the recording
	makam	List of makams associated with the recording. The information is fetched both from the related works and also the tags in MusicBrainz associated with the recording ("makam: [makam_name]")
	form	List of forms associated with the recording. The information is fetched both from the related works and also the tags in MusicBrainz associated with the recording ("form: [form_name]")
	usul	List of usuls associated with the recording. The information is fetched both from the related works and also the tags in MusicBrainz associated with the recording ("usul: [usul_name]")

5.1 Metadata

Given an audio recording, basic information about the file such as the sampling frequency, bit rate and duration are fetched using *eyeD3* [2] The recording MBID is also read from the metadata contained within the file (i.e. from the ID3 metadata in MP3 files). Next MusicBrainz is queried to obtain relevant metadata. Table 5.1 makes a summary of the extracted metadata. To crawl MusicBrainz, a wrapper around the open-source *MusicBrainz NGS bindings*[3] is created, which is specialized to fetch the OTMM related metadata in a structured manner.[4]

[2] http://eyed3.nicfit.net/

[3] `https://github.com/alastair/python-musicbrainzngs`

[4] https://github.com/sertansenturk/makammusicbrainz

5.2 Predominant Melody

In analyzing the tonal characteristics of music traditions involving harmony, features capturing harmonic content such as chroma features (Müller, Ewert, & Kreuzer, 2009; Gómez, 2006) are typically used. Chroma features are the state of the art features used in structure analysis of Eurogenetic musics (Paulus, Müller, & Klapuri, 2010) and also in relevant tasks such as version identification (Serrà et al., 2009) and audio-score alignment (Thomas et al., 2012). On the other hand, predominant melody is a more representative, musically meaningful and interpretable feature for analyzing melody-dominant musics (Chordia & Şentürk, 2013; Bozkurt, Ayangil, & Holzapfel, 2014; Şentürk, Holzapfel, & Serra, 2014; Koduri, Ishwar, Serrà, & Serra, 2014).

The fundamental pitch extraction method proposed by (De Cheveigné & Kawahara, 2002) (`YIN`) is used in (Bozkurt, 2008). `YIN` has been shown to be highly reliable to estimate the fundamental frequency over time in monophonic music. However, `YIN` (and melody extraction algorithms in general) introduce octave errors and spurious estimations, when heterophonic or noisy recordings are analyzed. To overcome these problems, (Bozkurt, 2008) has proposed a post-filtering method to eliminate contours with short duration, remove pitch estimations with low confidence and correct octave errors by shifting octave of the contours closer to the neighboring contours. Recently, Şimşek, Bozkurt, and Akan (2016) proposed a method based on variational mode decomposition (`VMD`). the predominant melody extraction method proposed by (Şimşek et al., 2016) (`VMD-SIM`) is reported to output comparable results to `YIN` and our own adaptation of MELODIA, which will be explained more in Section 5.2.1.

The predominant melody extracted from an audio fragment (a) is denoted as $\boldsymbol{\varrho} := \left[\rho_1 \dots \rho_{|\boldsymbol{\varrho}|}\right]$, where $\rho_i \in \boldsymbol{\varrho}$ is a pitch sample and $i \in [1 : |\boldsymbol{\varrho}|]$, where $|\boldsymbol{\varrho}|$ is the length of the predominant melody.

5.2.1 Adaptations of the Existing State-of-the-Art

Since predominant melody is the primary feature extracted from the audio recordings, its quality greatly determines the quality of the subsequent analysis steps. Nevertheless, developing a novel methodology would require a exhaustive effort and this task has not been addressed within the scope of this thesis. Instead, I and Hasan Sercan Atlı have worked on improving the predominant melody extraction by adapting state-of-the-art methodologies to OTMM recordings. Except the evaluation in (Şimşek et al., 2016) (which is addressed later in this Section), these adaptations are not evaluated quantitatively due to lack of ground-truth annotations, but

evaluated qualitatively by comparing the output predominant melodies visually and also by audio synthesis. Some of the algorithms are also evaluated extrinsically by comparing the results obtained from audio-score alignment methods (Section 6.7).

YIN

In the preliminary experiments on audio score alignment (Şentürk et al., 2012) (explained in Appendix B), we use YIN to extract the predominant melody of the audio recordings. We used the mex implementation[5] of YIN included in the Makam Toolbox. The hop size is taken as 10ms. Next, Makam Toolbox applies the post-processing explained in (Bozkurt, 2008). The toolbox has an additional option to quantize the pitch values in the predominant melody. The advantage of the quantized predominant melody is that it takes out minor pitch variations such as vibratos. Afterwards, a median filter with a window length of 41 frames (410ms) is applied to fix spurious jumps in the predominant melody. The predominant melody extraction method extracted in (Şentürk et al., 2012) is named as SEN-YIN$_f$.

The results of the preliminary experiments of section linking (Appendix B) shows that as the instrumentation of a recording gets more complex (i.e. the tendency of observing more heterophonic interactions and expressive elements in an audio recording increases), the section linking performance decreases almost monotonically. This suggests that an improvement in the extraction of audio pitch contour is necessary. Through inspecting errors in the audio recording level, it is seen that the current bottleneck of the system is the pitch estimation. Since YIN is designed for monophonic sounds, lots of confusions arise in the predominant melody extraction due to the heterophonic nature of OTMM, especially in ensemble performances. Moreover, YIN is found to lose its robustness, where there are substantial usage of expressive elements such as legatos, slides and tremolos.

MELODIA

Based on the observations in our preliminary section-linking experiments using YIN, we started optimizing MELODIA, a predominant melody extraction method proposed for polyphonic music signals by (Salamon & Gómez, 2012), in our following experiments on section-level audio-score

[5] http://es.mathworks.com/help/matlab/call-mex-files-1.html?s_cid=wiki_mex_1

alignment (Section 6.7.2).[6] The predominant melody extraction procedure explained in (Şentürk, Holzapfel, & Serra, 2014) is named as SEN--MEL.

The methodology proposed by Salamon and Gómez (2012) assumes that there is no predominant melody in time intervals where the peaks of the pitch saliences are below a certain magnitude with respect to the mean of all the peaks. Moreover, it eliminates pitch contours, which are considered as belonging to the accompaniment. However, as OTMM is heterophonic (Section 2.2), unvoiced intervals are very rare. Applied on OTMM recordings, the contour selection step in the methodology (Salamon & Gómez, 2012, Section D) treats a substantial amount of melody candidates as non-salient (due to the embellishments and wide dynamic range), and dismisses a significant portion of pitch contours as unvoiced. Hence, we include all the non-salient candidates to guess predominant melody and obtain the audio predominant melody $\boldsymbol{\varrho}$.

Next the predominant melody is downsampled from MELODIA's default frame rate of ~ 344.5 frames per second (hop size of 128 samples) to ~ 21.5 frames per second or a period of ~ 46 ms, which is sufficient to track the note changes. In our alignment experiments, melody extraction is performed using various pitch resolutions (Section 6.7.2).

We also compare the extracted predominant melody and HPCPs as input features for the audio-score alignment method. The results (explained in Section 6.7.2) show that SEN-MEL performs better compared to both SEN-YIN$_f$ and HPCPs. Moreover, a pitch precision of 50 cents is adequate for section-level audio-score alignment. Nevertheless, the precision can be increased further to use the extracted predominant melody later, in more precision-demanding computational tasks such as tonic identification (Section 5.7), tuning analysis (Section 5.9) and note-level audio-score alignment (Section 6.8). For this reason, we select the bin resolution as 1 cents instead of the default (10 cents) in the computation of the pitch salience function.[7]

Additional adaptations on MELODIA

The predominant melody computed by SEN-MEL still produces substantial amount of errors, especially when the music is played softer than the rest of the audio. This becomes a noticeable problem in the end of the melodic phrases, where musicians choose to play softer. For this reason we decided to optimize the methodology of (Salamon & Gómez, 2012) step by

[6] We use the Essentia implementation of the algorithm (Bogdanov et al., 2013).

[7] http://essentia.upf.edu/documentation/reference/streaming_PitchSalienceFunction.html

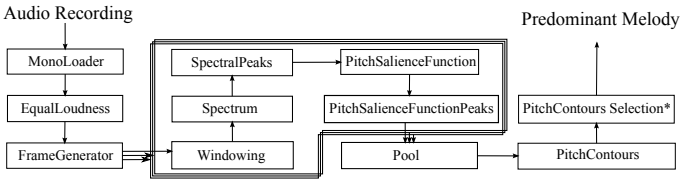


Figure 5.1: The block diagram of ATL-MEL

step (Atlı et al., 2014).[8] This method is termed as ATL-MEL throughout the text.

Figure 5.1 shows the steps followed to compute the predominant melody. All the steps shown (inside boxes) in Figure are named the same with the respective Essentia class, except the "PitchContours Selection" step modified by us (explained below). After "FrameGenerator" step, the audio is divided into frames and these frames are processed in parallel between the "Windowing" and "PitchSalienceFunctionPeaks" steps. For more information about the parameters, we refer the reader to the Essentia documentation[9] and (Salamon & Gómez, 2012).[10]

In the computation of pitch contours,[11] we experimented on different values of the peak distribution threshold parameter to get a satisfactory pitch contour length. We emprically set the "peakDistributionTreshold" parameter as 1.4. This setting provides longer pitch contours compared to the default value of 0.9. Originally, the contour selection step in the method is trained using Eurogenetic musics (Salamon & Gómez, 2012), which is main reason for erroneous unvoiced estimations (explained in SEN-MEL). However, lacking the ground truth for training during the time of development, we decided to replace the contour selection step with a simple and generic heuristics. Once the pitch contours are obtained, we order the pitch contours according to their length and start with selecting the longest one. Then, we remove all portions of pitch contours which overlap with the selected pitch contour. We carry the same process for the next longest pitch contour, and so forth. By repeating the process for all pitch contours, we obtain the predominant melody of the audio recording

[8] This work was mainly conducted by Hasan Sercan Atlı under my guidance during his Erasmus+ stay in Music Technology Group, UPF in Summer 2014. Apart from proposing methodology, I have also contributed to the code with bug fixing, refactoring, documentation and deployment. Andrés Ferraro has also contributed to the code deployment and refactoring.

[9] http://essentia.upf.edu/documentation/algorithms_reference.html

[10] The implementation is openly avaliable at https://github.com/sertansenturk/predominantmelodymakam

[11] http://essentia.upf.edu/documentation/reference/streaming_PitchContours.html

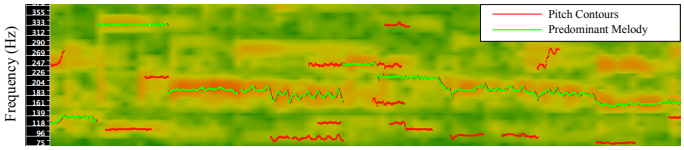


Figure 5.2: Pitch contours and the resultant predominant melody extracted from an audio fragment using ATL-MEL, plotted on top of the spectrogram of the fragment. Sonic Visualizer is used to compute the spectrogram and to display the features.

(Figure 5.2).

Due to lack of ground truth in the time of implementation, we made a qualitative comparison between the predominant melodies obtained from SEN-MEL and ATL-MEL by synthesizing and listening the resultant predominant melodies synchronous to the audio playback. We observed that ATL-MEL is able to capture the melody better when the music is played softer in the expense of obtaining more spurious pitch estimations and octave errors.

To get rid of the spurious estimations and octave errors, we re-introduced the post-processing method proposed by Bozkurt (2008). We first tried the *PitchFilter* function[12] in Essentia, an open implementation of this method. However, this implementation was not good enough in removing spurious estimations, which affected the tonic identification accuracy using the last note detection method Section 5.7.2. Therefore, we made our own open source implementation.[13] The filtered variant of the predominant melody extracted using the procedure described by Atlı et al. (2014) is named as ATL-MEL$_f$.[14]

In his masters thesis, Atlı (2016) extracted the predominant melody of 18 audio recordings in the OTMM partial audio-score alignment dataset (Section 3.3.6) using ATL-MEL$_f$ and recorded the number of the predominant melody samples that are within 1 Hc vicinity of the annotated notes (Figure 5.3). He reports that 143308 out of 157231 samples (91.14%) coincide with the note annotations, implying ATL-MEL$_f$ outputs reliable predominant melody estimations. Nevertheless, these findings should not be regarded as an intrinsic evaluation of the method, since the note annotations include information about neither the intonation devations nor the embellishments.

[12] http://essentia.upf.edu/documentation/reference/std_PitchFilter.html

[13] https://github.com/hsercanatli/pitchfilter

[14] Reimplementation of the pitch filter has been done by Hasan Sercan Atlı within his masters research under the advisorship of Barış Bozkurt. Later, I have contributed to the code with bug fixing, refactoring, documentation and deployment.

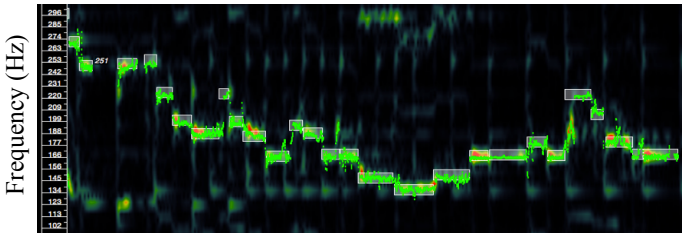


Figure 5.3: The predomiant melody (in green), the note annotations (white transparent boxes) and the melodic range spectrogram (background) of an audio fragment in the OTMM partial audio-score alignment dataset. Sonic Visualizer is used to compute the melodic range spectrogram and to display the features. The Figure is reproduced from (Atlı, 2016) courtesy of Hasan Sercan Atlı.

In parallel, Şimşek et al. (2016) compared their method `VMD-SIM` with $\texttt{ATL-MEL}_f$ and `YIN` on four heterophonic OTMM recordings. They report the evaluation measures used in Music Information Retrieval EXchange (MIREX) Audio Melody Extraction task.[15] The results (Şimşek et al., 2016, Table 1) indicate that $\texttt{ATL-MEL}_f$ is superior to `YIN`, and it outputs either comparable or better results than `VMD-SIM` over all evaluation measures.

5.3 Harmonic Pitch Class Profiles

As mentioned in the Section 5.2, HPCPs (Gómez, 2006) are compared to predominant melody in section linking experiments and predominany melody is intrinsically observed to be a more representative feature to describe the melodic characteristics of OTMM. The experiments will be introduced in Section 6.7.2.

We use the default parameters given in (Gómez, 2006) to compute the HPCPs. The hop size and the frame size are chosen to be 2048 (e.g. ~ 21.5 frames per second) and 4096 samples respectively. The first bin of the HPCPs is assigned to the karar pitch or pitch-class κ identified automatically using distribution matching (Section 5.7.2) or score-informed tonic identification (Section 6.4) and HPCPs, $\hat{\mathbf{\Gamma}}^{(a)} = [\hat{\gamma}_1^{(a)}, \ldots, \hat{\gamma}_{|\hat{\mathbf{\Gamma}}^{(a)}|}^{(a)}]$ are obtained, where $|\hat{\mathbf{\Gamma}}^{(a)}|$ is the number of frames in time. Each frame consists of a constant number of bins. The number of bins denoted as n_{HPCP} determines the pitch resolution of the HPCPs, i.e. the width of each

[15] http://www.music-ir.org/mirex/wiki/2016:Audio_Melody_Extraction#Evaluation_Procedures

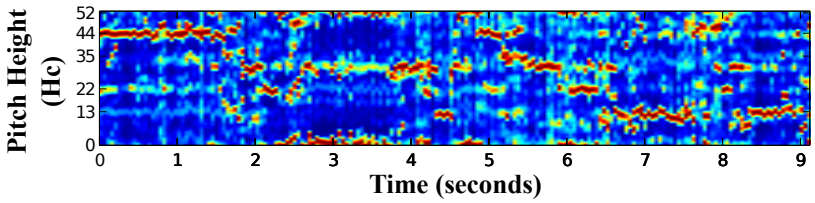


Figure 5.4: HPCPs extracted for a short audio fragment. The first bin of the HPCPs is centered at its annotated tonic frequency

bin equals to $1200/n_{\text{HPCP}}$. Figure 5.4 shows the tonic normalized HPCPs extracted from a short audio fragment.[16]

5.4 Pitch Intervals

In order to process the pitch content independent of the absolute frequency (and as a result, independent of the octave), the pitch values should be converted to intervals. This is achieved by converting the pitch values in Hertz unit to cents. A pitch value ρ in Hertz is converted to the cent scale by taking a frequency value f as the reference in the equation below:

$$\hat{\rho}^f := 1200 \log(\rho / f) \tag{5.1}$$

Hence, the cent value shows the ratio between two frequencies. This operation can be applied to any feature representing the pitch content in Hertz such as the predominant melody (Section 5.2) and the bins of the pitch(-class) distributions (Section 5.5), and the stable pitches (Section 5.6). For example, given the predominant melody $\boldsymbol{\varrho}$ extracted from an audio fragment (a), $\hat{\boldsymbol{\varrho}}^f := \left[\hat{\rho}_1^f \ldots \hat{\rho}_{|\hat{\boldsymbol{\varrho}}^f|}^f\right]$ denotes the predominant melody converted to cent scale with respect to the frequency value f. Here $\hat{\rho}_i^f \in \hat{\boldsymbol{\varrho}}^f$ is a pitch sample in cents and $i \in \left[1 : |\hat{\boldsymbol{\varrho}}^f|\right]$, where $|\hat{\boldsymbol{\varrho}}^f| = |\boldsymbol{\varrho}|$ is the length of the predominant melody.

Given a pitch value ρ, the cent distance of its pitch class to the reference frequency f (also termed as "octave-wrapped" cent distance throughout the text) is computed as:

$$\triangle(\rho, f) := \hat{\rho}^{(f)} \mod 1200 \tag{5.2}$$

[16] http://musicbrainz.org/work/9aaf5c0b-4642-40fd-97ba-c861265872ce

where mod is the modulo operation and $\hat{\rho}^{(f)}$ is the cent-distance of ρ with respect to f (Equation 5.1). Notice that $\triangle\,(f,\rho) = -\ \triangle\,(\rho,f) \mod 1200 = 1200 - \triangle(\rho,f)$.

The shortest "octave-wrapped" cent-distance $\blacktriangle(\rho,f)$ between ρ and f is computed as:

$$\blacktriangle(\rho,f) := min\big(\triangle(\rho,f), \triangle(f,\rho)\big) \tag{5.3}$$

5.5 Pitch and Pitch-Class Distributions

Pitch distributions (PDs) and pitch-class distributions (PCDs) show the relative occurrence of the pitch and pitch class values with respect to each other, respectively. PDs and PCDs are commonly used for analysis of tonic and pitch organisation. Krumhansl and Shepard (1979) used 12-dimensional PCDs to study the tonal organisation of euro-genetic musics. PCDs are also used for relevant tasks such as key detection and chord recognition (Gómez, 2006; Temperley & Marvin, 2008) for Eurogenetic musics.

For musical styles involving microtonality, the pitch space must be extended beyond 12-dimensions to model, analyze and predict the melodic properties of the studied music (Bozkurt, Yarman, Karaosmanoğlu, & Akkoç, 2009; Gedik & Bozkurt, 2010; Şentürk, 2011; Şentürk, Holzapfel, & Serra, 2014). Pitch distributions are used in many tasks such as tonic identification (Bozkurt, 2008; Atlı et al., 2015), makam recognition (Gedik & Bozkurt, 2010) and tuning analysis (Bozkurt et al., 2009). Likewise these features will be used extensively throughout the thesis in stable pitch and pitch-class extraction (Section 5.6), tonic identification (Sections 5.7 and 6.4), tempo estimation (Section 6.5), tuning analysis (Section 5.9), audio melodic progression analysis (Section 5.10) and note modeling (Section 6.11).

These distributions can be computed from any time-series representation of pitch such as spectrograms, chroma features and predominant melody. Since Ottoman-Turkish makam music is a melody-dominant tradition (Section 2.2), we use the predominant melody to compute these distributions. First, the predominant melody $\boldsymbol{\varrho}$ is converted from Hz-scale to cent scale by taking a frequency value $*$ as the reference and $\hat{\boldsymbol{\varrho}}^{*}$ is obtained. $*$ equals to the tonic pitch κ, if available; otherwise $*$ is arbitrarily assigned to 440 Hz. The conversion to cents allows to compute the distribution with constant bin width in the cents scale . The values in both distributions are computed as:

$$\hat{h}_n^* := \frac{\sum_{i=1}^{|\hat{\boldsymbol{\varrho}}^*|} \ell_n\left(\hat{\rho}^*\right)}{|\hat{\boldsymbol{\varrho}}_i^*|} \tag{5.4}$$

where $\hat{h}_n^*$ is the relative occurence computed for the n^{th} bin of the distribution $\hat{\boldsymbol{H}}^*$, computed from the samples $\hat{\rho}^* \in \hat{\boldsymbol{\varrho}}^*$. As a convention, 0^{th} bin is centered around the reference frequency; or in other words, the reference pitch is mapped to 0 cents.

The accumulator function $\ell_{P,n}$ for PDs is defined as:

$$\ell_{P,n}(\hat{\rho}) := \begin{cases} 1, & c_n \leq \hat{\rho} \leq c_{n+1} \\ 0, & \text{otherwise} \end{cases} \tag{5.5}$$

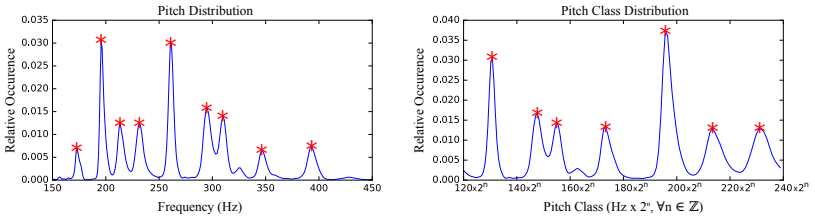
where $\hat{\rho}$ is a pitch sample in cents and (c_n, c_{n+1}) are the boundaries of the n-th bin. Similarly the accumular function for PCDs is defined as:

$$\ell_{PC,n}(\hat{\rho}) := \begin{cases} 1, & c_n \leq (\hat{\rho} \mod 1200) \leq c_{n+1} \\ 0, & \text{otherwise} \end{cases} \tag{5.6}$$

Note that the PCD is a "circular" feature, e.g. the first and the last bins are next to each other. On the other hand, PD would span to multiple octaves. Therefore a PD would typically have bins with negative indices, which represent frequencies below the reference. Also notice that both PD and PCD are normalized such that the resultant distribution can be treated as a probability density function.

The bin size $b\ (\hat{\boldsymbol{H}})$ of a distribution $\hat{\boldsymbol{H}}$ determines how precise the distribution is (to the extend allowed by the cent-precision of the predominant melody) in representing the pitch space, the tuning of the stable pitches and the microtonal characteristics in a lower-level. The computed distributions might need to have a small bin size, e.g. less than a quarter tone (50 cents) for many music cultures (Gedik & Bozkurt, 2010; Chordia & Şentürk, 2013). We select a constant bin size for the computed distributions, i.e. $b\ (\hat{\boldsymbol{H}}) = c_{n+1} - c_n, \forall n$. The bin centers of both PDs and PCDs are selected such that the reference frequency r is represented as a bin centered around 0 cents. We denote the number of bins in a distribution $\hat{h}$ as $|\hat{\boldsymbol{H}}|$. Note that $|\hat{\boldsymbol{H}}_{PC}|$ equals to $\lfloor 1200/\ b\ (\hat{\boldsymbol{H}}) \rfloor$ in a PCD.

To remove the spurious peaks in the distribution, the distribution is convolved with a Gaussian kernel and a "smoothed" distribution is obtained (Chordia & Şentürk, 2013). The standard deviation of the Gaussian kernel, termed as the kernel width $\sigma(\hat{\boldsymbol{H}})$, determines how smooth the resulting distribution will get. The kernel width should be comparable to the bin size $b\ (\hat{\boldsymbol{H}})$ since a value lower than one third of the bin



(a) (b)

Figure 5.5: The pitch distribution and pitch-class distribution computed a predominant melody. The stable pitch and stable pitch classes are also marked on the pitch distribution and the pitch-class distribution, respectively.

size would not contribute much to smoothing[17] and a high value would "blur" the distribution too much. Moreover, this parameter has a direct impact on the number and the location of peaks in distribution, which are later used in tonic identification (Section 5.7) and tuning analysis (Section 5.9). Finally, bin values are converted from cents back to Hz using the inverse of the Equation 5.1 and $\boldsymbol{H}$ is obtained (Figure 5.5).

Unless stated otherwise, the default value of the bin size b ($\hat{\boldsymbol{H}}$) is selected as 7.5 cents $\approx 1/3$ Hc, which is reported as an empirical optimal for this feature to capture tuning differences (Bozkurt, 2008). The standard deviation of the Gaussian kernel $\sigma(\hat{\boldsymbol{H}})$ is selected as 7.5 cents such that it practically affects an area of six standard deviations (22.5 cents $\approx$ 2 Hc peak to tail) and does not mask quarter tone intervals (50 cents) (Şentürk et al., 2013). In our implementation, we select the overall width of the Gaussian kernel as 37.5 cents from peak to tail (i.e the kernel has 11 samples) for performance reasons.[18]

5.6 Stable Pitches and Pitch-Classes

To extract the performed notes and their intervallic relations the first step is to extract the stable pitches and the stable pitch-classes performed in the audio fragment. A simple method is to detect the peaks in the computed pitch distributions and pitch-class distributions to extract these two features, respectively.

[17] The values of the bins in a Gaussian kernel, which are more than three standard deviations away from the mean are greatly diminished.

[18] The implementation of pitch distribution and pitch-class distribution is available at `https://github.com/altugkarakurt/morty/blob/556828039c781d4723ae8f7f3acd97cad1566c88/morty/pitchdistribution.py`.

We detect the peaks in the distribution using the method explained in (Smith III & Serra, 1987). We only consider the peaks, which have a ratio between its height and the maxima of the distribution above a constant threshold. As will be explained in Section 5.7.5, we empirically set the ratio $\delta(\boldsymbol{H})$ to 0.15. The peaks indicate the stable pitches or stable pitch-classes performed in the fragment depending on the distribution input (Figure 5.5). We denote the set of stable pitches extracted from the pitch distribution $\boldsymbol{H}_P$ as $\boldsymbol{\Phi}_P := \{\phi_{P,1}, \phi_{P,2}, \dots, \}$. Similarly the set of stable pitch classes extracted from the pitch class distribution $\boldsymbol{H}_{PC}$ is denoted as $\boldsymbol{\Phi}_{PC} := \{\phi_{PC,1}, \ \phi_{PC,2}, \dots, \}$.

This procedure will not be able to extract stable pitches/pitch-classes, which are performed considerably less than maxima of the distribution due to peak selection threshold or are masked by other peaks. Therefore the extracted set of pitch-classes are limited to "prominent" stable notes, which typically correspond to the pitches that correspond to the scale of the makam.

5.7 Karar and Makam

In many music cultures, the melodies adhere to a particular melodic framework, which specifies the melodic characteristics of the music. While the function and the understanding of these frameworks are distinct from a culture-specific perspective, in a broader sense they may be considered as the "modes" of the studied music culture. Some of the music traditions that can be considered as "modal" are Indian art musics, the makam traditions and medieval church chants (Powers, et al., n.d.). mode recognition is an important complementary task in computational musicology, music discovery, music similarity and recommendation. In the context of Ottoman-Turkish makam music, mode is synonymous to makam.

Tonic is another important musical concept. It acts as the reference frequency for the melodic progression in a performance. In many music cultures there is no standard reference tuning frequency, which makes it crucial to identify the tonic frequency to study melodic interactions. Estimating the tonic of a recording is the first step for various computational tasks such as tuning analysis (Bozkurt, 2012), automatic transcription (Benetos & Holzapfel, 2015) and melodic motif discovery (Gulati, Serrà, Ishwar, Şentürk, & Serra, 2016). In the context of Ottoman-Turkish makam music, tonic is synonymous to karar.

There has been a extensive interest on mode recognition in the last decade (Koduri, Gulati, Rao, & Serra, 2012). Most of these work focus on culture-specific approaches for music traditions such as Ottoman-

Turkish makam music (Gedik & Bozkurt, 2010), Carnatic music (Dighe, Karnick, & Raj, 2013; Gulati, Serrà, Ishwar, et al., 2016), Hindustani music (Chordia & Rae, 2007; Chordia & Şentürk, 2013; Gulati, Serrà, Ganguli, Şentürk, & Serra, 2016) and Dastgah music (Abdoli, 2011). A considerable portion of these studies are based on comparing pitch distributions (Chordia & Rae, 2007; Chordia & Şentürk, 2013; Dighe et al., 2013; Gedik & Bozkurt, 2010), which are shown to be reliable in the mode recognition task. There also exists recent approaches that are based on characteristic melodic motif mining using network analysis (Gulati, Serrà, Ishwar, et al., 2016), aggregating note models using automatic transcription (Koduri et al., 2014) or audio-score alignment (Şentürk, Koduri, & Serra, 2016) and neural network based classification (Suma & Koolagudi, 2015; Shetty & Achary, 2009), all of which are designed specific to the studied music culture and are not generalizable to other music cultures without considerable effort. Similarly, several studies on tonic identification use pitch distribution based methods (Bozkurt, 2008; Chordia & Şentürk, 2013). More recently there has been an interest in culture specific methods for this task (Şentürk et al., 2013; Gulati, 2011; Atlı et al., 2015) that make use of heuristics and the musical characteristics of the studied tradition.

As stated earlier in Section 2.2, there is not agreed reference frequency in OTMM. This requires identification of the tonic frequency for many tasks such as tuning analysis (Bozkurt et al., 2009), automatic transcription (Benetos & Holzapfel, 2015) and audio-score alignment (Şentürk, Holzapfel, & Serra, 2014).

5.7.1 Problem Definition

Mode recognition is defined as classifying the mode $\mu^{(a)}$ of an audio fragment (a) from a discrete set of modes $\boldsymbol{\mathcal{M}} := \{\mu_1, \ldots, \mu_V\}$, where $\mu^{(a)} \in \boldsymbol{\mathcal{M}}$ and $|\boldsymbol{\mathcal{M}}|$ is the total number of modes. In mode recognition, we assume that the true tonic frequency (or pitch class) $\mathfrak{t}^{(a)}$ of the audio recording is available.

Tonic identification is defined as estimating the frequency or the pitch class (if the octave information of the tonic is not well-defined for the music culture or the performance) of the performance tonic. We denote the estimated tonic of an audio fragment as $\kappa^{(a)}$. Tonic is a continuous variable. However, in practice, the tonic is typically constrained to be one of the stable pitches or pitch classes performed in the audio fragment (Chordia & Şentürk, 2013; Gedik & Bozkurt, 2010). With this assumption, tonic identification can be reformulated as estimating the tonic frequency or the pitch class $\kappa^{(a)}$ from a finite set of stable pitches/pitch-

classes $\mathbf{\Phi}^{(a)} := \{\phi_1^{(a)}, \ldots, \phi_{|\mathbf{\Phi}^{(a)}|}^{(a)}\}$ performed in an audio fragment (a), where $\kappa^{(a)} \in \mathbf{\Phi}^{(a)}$ and $|\mathbf{\Phi}^{(a)}|$ is the number of the stable pitches/pitch-classes in the audio fragment. In the context of OTMM, we focus on identifying the pitch class of the tonic, since the frequency of the tonic is ambiguos in heterophonic recordings, as explained in Section 2.2. In tonic identification, we assume that the true mode $\mathfrak{m}^{(a)}$ of the audio recording is known.

A third scenario arises when both the tonic $\kappa^{(a)}$ and the mode $\mu^{(a)}$ of the recording (a) are unknown. In this case, we identify the tonic and recognize the mode together, which we term as *joint estimation of mode and tonic*.

Note that these scenarios are actually multi-class problems since the mode and the tonic may change (and not necessarily simultaneously) throughout the performance. This is a more challenging problem, where we would not only like to obtain the set of the modes and tonics in the performance but also mark the intervals, where these musical "attributes" are observed.[19] Later in Section 6.7, audio-score alignment will be used to identify the tonic and mode with their time intervals.

In the context of OTMM, mode recognition and tonic identification are termed more specifically as *makam recognition* and *karar* identification.

5.7.2 Methodologies

For makam recognition and joint recognition tasks, we generalize two state-of-the-art methods on distribution matching (Gedik & Bozkurt, 2010; Chordia & Şentürk, 2013). The generalized method is explained in (Karakurt et al., 2016).[20]

For karar identification, in addition to the generalized distribution matching method explained above, we also use the last-note detection method proposed in (Atlı et al., 2015).[21]

[19] A manually annotated example for OTMM is given in `http://musicbrainz.org/recording/37dd6a6a-4c19-4a86-886a-882840d59518`

[20] This work was mainly conducted by Altuğ Karakurt within his Erasmus+ internship under my guidance. I was responsible to design the methodology, dataset and experiments. I have also contributed in developing the library, bug fixing, refactoring, documentation and deployment.

[21] This work was mainly conducted by Hasan Sercan Atlı within his masters research under the advisorship of Barış Bozkurt. I have assisted Hasan Sercan Atlı in implementing the methodology and improving its performance. I have also contributed with bug fixing, refactoring, documentation and deployment.

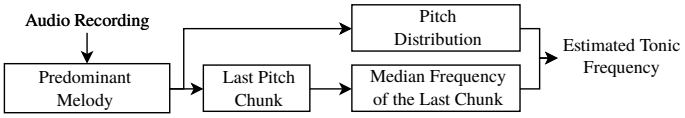


Figure 5.6: The block diagram of ATL-TON

Last note detection

ATL-TON uses the musical knowledge that a makam music performance ends in the karar note (Section 2.2). The method first extracts the predominant melody from the audio recording (Section 5.2). The end of the predominant melody is divided into chunks according to the pitch jumps on the predomiant melody. Initially the karar frequency is estimated as the median of all the frequencies in the last chunk. Next, a pitch distribution is computed (Section 5.5) using only the frequency values in the predominant melody, which are close to the initial estimation (± a semitone). Then the karar estimation is refined as the frequency of the closest peak in the histogram. The flow diagram of ATL-TON is shown in Figure 5.6.

This method provides a simple and generalizable solution without requiring neither training nor additional information such as the makam of the audio recording or the music score (as needed by the distribution matching methodology; Section 5.7.2). Note that the method will fail for any audio fragment, which does not end with the karar note and the accuracy of this method is susceptible to the quality of the predominant melody in the end of the fragment.[22]

Distribution matching

In (Karakurt et al., 2016), we combine and generalize the two state of the art methods, originally proposed for audio recordings of Ottoman-Turkish makam music (Bozkurt, 2008; Gedik & Bozkurt, 2010)[23] and short audio fragments of Hindustani music (Chordia & Şentürk, 2013). The generalized methods are supervised and use k nearest neighbors (kNN) estimation for classification. Our implementation is generic such that the parameters selected in the feature extraction, training and testing steps can be optimized for the properties of the studied music tradition. We also allow the user to classify either short audio fragments or complete audio recordings

[22] The open implementation of this method is available in https://github.com/hsercanatli/tonicidentifier_makam

[23] (Bozkurt, 2008) introduces karar identification methodology, which is later extended to makam recognition in (Gedik & Bozkurt, 2010).

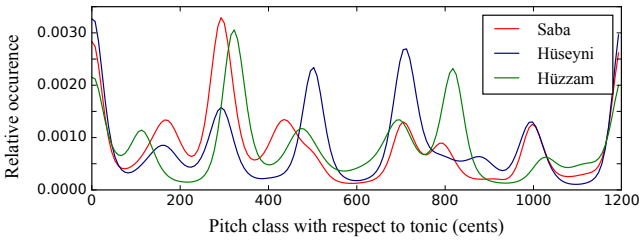


Figure 5.7: An example model with a single PCD per makam trained for three makams

and switch between different features, training schemes and tasks as introduced in (Bozkurt, 2008; Gedik & Bozkurt, 2010; Chordia & Şentürk, 2013). Later in Section 5.7.3, we demonstrate the experiments for the parameter selection and optimization on a test dataset of audio recordings of OTMM (Section 5.7.3) and the results of (Chordia & Şentürk, 2013) on a Hindustani and a Carnatic music dataset using the optimal parameters reported in (Chordia & Şentürk, 2013) (Appendix E.2).

In the training step, we use audio fragments with annotated makam and karar. We first extract a predominant melody for each audio fragment. These are used to compute either pitch distribution or pitch-class distribution (Section 5.5). Next, we create makam models from these computed distributions.

Given an audio recording with an unknown makam and/or karar, we extract its predominant melody and compute the distribution. Then, we compute a distance or dissimilarity between the distribution of the test audio and the selected distributions in the training models and compute the k nearest neighbors according to the computed measure. Finally, we estimate the unknown makam and/or karar as the most common candidate among the k nearest neighbors.

Now we proceed to explain the generalized methodology in detail.

Training model: The generalized method is supervised and hence require training data, i.e. audio fragments with annotated makam and karar. From a training audio fragment (x), we first extract the predominant melody $\boldsymbol{\varrho}^{(x)}$ and normalize with respect to the annotated karar frequency $\kappa^{(x)}$ (Equation 5.1). Next, the normalized predominant melodies $\hat{\boldsymbol{\varrho}}^{\kappa,(x)}$ are grouped according to the annotated makam $\mu^{(x)}$ of each individual fragment.

The fundamental difference between the methodologies proposed in (Bozkurt, 2008; Gedik & Bozkurt, 2010) and (Chordia & Şentürk, 2013) is the training model $\boldsymbol{\mathcal{T}}$. The methodology proposed in (Bozkurt, 2008; Gedik & Bozkurt, 2010) joins all the normalized predominant melodies and

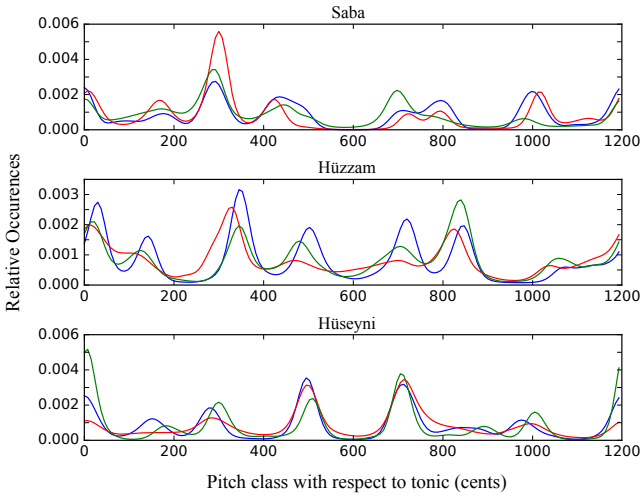


Figure 5.8: An example model with three PCDs per makam trained for three makams

compute a single distribution per mode. On the other hand, (Chordia & Şentürk, 2013) creates a separate distribution from each annotated audio fragment. From a machine learning perspective (Bozkurt, 2008; Gedik & Bozkurt, 2010) represents each makam with a single data point (Figure 5.7), whereas (Chordia & Şentürk, 2013) represents them with many (Figure 5.8) in an $|\boldsymbol{H}|$-dimensional space, where $|\boldsymbol{H}|$ is the number of bins in the distributions. From now on, we term the training models using the training step in (Bozkurt, 2008; Gedik & Bozkurt, 2010) and (Chordia & Şentürk, 2013) as "single distribution per mode" and "multi-distributions per mode", respectively. We denote the obtained model as $\boldsymbol{\mathcal{T}} := \left\{ \langle \hat{\boldsymbol{H}}_1, \mu_1 \rangle, \langle \hat{\boldsymbol{H}}_2, \mu_2 \rangle, \dots \right\}$, where $\langle \hat{\boldsymbol{H}}_j, \mu_j \rangle$ is a tuple. $\hat{\boldsymbol{H}}_j$ and μ_j denotes the trained distribution and the makam label of the j^{th} data point, respectively.[24] The model $\boldsymbol{\mathcal{T}}$ consists of the distribution representations for $|\boldsymbol{\mathcal{M}}|$ makams, where $|\boldsymbol{\mathcal{M}}|$ is the number of unique makam labels μ_i (where $i \in [1 : |\boldsymbol{\mathcal{M}}|]$) in the training fragments.

Nearest Neighbor Selection: In mode recognition, karar identification and joint estimation tasks, the common step is to find the nearest neighbor(s) of a selected distribution among a set of distributions to be compared against. To find these nearest neighbors, we compute a distance or a dissimilarity between the test distribution and each distribution in the comparison set selected from the training model (Cha & Srihari, 2002).

[24] The annotated karar and the source are not used later in the classification step (Section 5.7.2). Therefore these labels are omitted from the representation in $\boldsymbol{\mathcal{T}}$.

We have implemented the distance and the similarity metrics in (Gedik & Bozkurt, 2010; Chordia & Şentürk, 2013), namely, City-Block (L_1 Norm) distance, Euclidean (L_2 Norm) distance, L_3 Norm, Bhattacharyya distance, intersection and cross correlation. Note that intersection and cross correlation are similarity metrics, hence we convert them to dissimilarities (i.e. $1-$similarity) instead. The choice of the distance or dissimilarity measure plays a crucial role in the neighbor selection.

Given a distribution $\hat{\boldsymbol{H}}_j$ in the model $\boldsymbol{\mathcal{T}}$ and the distribution $\hat{\boldsymbol{H}}^{*,(a)}$ extracted from an audio fragment (a) by taking the frequency $*$ as the reference, the implemented metrics between these distributions are given below:

- City-Block (L_1 Norm) Distance:

$$\diamondsuit_{L1}\left(\hat{\boldsymbol{H}}_j, \hat{\boldsymbol{H}}^{*,(a)}\right) = \frac{1}{|\hat{\boldsymbol{H}}_j|}\sum_n |\hat{h}_{j,n} - \hat{h}_n^{*,(a)}| \tag{5.7}$$

- Euclidean (L_2 Norm) Distance:

$$\diamondsuit_{L2}\left(\hat{\boldsymbol{H}}_j, \hat{\boldsymbol{H}}^{*,(a)}\right) = \sqrt{\sum_n \left(\hat{h}_{j,n} - \hat{h}_n^{*,(a)}\right)^2} \tag{5.8}$$

- L_3 Norm:

$$\diamondsuit_{L3}\left(\hat{\boldsymbol{H}}_j, \hat{\boldsymbol{H}}^{*,(a)}\right) = \left(\sum_n \left|\hat{h}_{j,n} - \hat{h}_n^{*,(a)}\right|^3\right)^{1/3} \tag{5.9}$$

- Bhattacharyya Distance:

$$\diamondsuit_{bhat}\left(\hat{\boldsymbol{H}}_j, \hat{\boldsymbol{H}}^{*,(a)}\right) = -\log\sum_n \sqrt{\hat{h}_{j,n}\,\hat{h}_n^{*,(a)}} \tag{5.10}$$

- Inverse of Intersection:

$$\diamondsuit_{intr^{-1}}\left(\hat{\boldsymbol{H}}_j, \hat{\boldsymbol{H}}^{*,(a)}\right) = 1 - \frac{1}{|\hat{\boldsymbol{H}}_j|}\sum_n min\left(\hat{h}_{j,n}, \hat{h}_n^{*,(a)}\right) \tag{5.11}$$

- Negative of Cross-Correlation:

$$\diamondsuit_{1\text{ - }ccor}\left(\hat{\boldsymbol{H}}_j, \hat{\boldsymbol{H}}^{*,(a)}\right) = 1 - \frac{1}{|\hat{\boldsymbol{H}}_j|}\sum_n \hat{h}_{j,n}\,\hat{h}_n^{*,(a)} \tag{5.12}$$

Remember that the reference is always the 0^{th} bin. Also remark that the PDs would not typically have the same length. During the distance or

dissimilarity computation, the values of the "missing" bins in the distributions are treated as 0.

After the distances or the dissimilarities are computed, the compared distributions are ranked and the k nearest neighbors are selected. We then estimate the test sample as the most common label of the neighbors. In case of a tie between two or more groups, we select label of the group, which accumulates the lowest distance or dissimilarity. Note that if a single-distribution is computed for each mode as explained in (Gedik & Bozkurt, 2010), the k value is always 1, since each mode is only represented by one sample.

Now we proceed to explain the procedure for each task in detail.

Makam Recognition: Given an audio fragment (a) with an unknown mode, we compute the distribution $\hat{\boldsymbol{H}}^{\mathfrak{k},(a)}$ by taking the annotated karar $\mathfrak{k}^{(a)}$ as the reference (Section 5.5). Next we compute the distance or the dissimilarity between $\hat{\boldsymbol{H}}^{\mathfrak{k},(a)}$ and the trained distribution $\hat{\boldsymbol{H}}_j$ of each tuple $\in \boldsymbol{\mathcal{T}}$, where $\boldsymbol{\mathcal{T}}$ is the trained model, and obtain the k nearest neighbors to (a). We estimate the makam$\mu^{(a)}$ of (a) as the most common makam label within the nearest neighbors.

Tonic Identification: Given an audio fragment (a) with the annotated makam $\mathfrak{m}^{(a)}$, we first extract the predominant melody $\boldsymbol{\varrho}^{(a)}$. Then we compute a distribution $\boldsymbol{H}^{(a)}$ (Section 5.5). We detect the peaks in the distribution and obtain the set of stable pitches/pitch-classes as $\boldsymbol{\Phi}^{(a)} := \left\{\phi_1^{(a)}, \ldots, \phi_{|\boldsymbol{\Phi}^{(a)}|}^{(a)}\right\}$, where $|\boldsymbol{\Phi}^{(a)}|$ is the number of detected peaks (Section 5.6). Assuming each of the peaks $\phi_i^{(a)}$ as the karar candidate, we compute $\hat{\boldsymbol{H}}^{\phi_i,(a)}$ such that the index of the bin representing $\phi_i^{(a)}$ becomes 0 in the shifted distribution.

From the training model $\boldsymbol{\mathcal{T}}$, we select all the $\langle \hat{\boldsymbol{H}}_j, \mu_j \rangle \in \boldsymbol{\mathcal{T}}$ such that the label $\mu_j = \mathfrak{m}^{(a)}$. Next we compute the distance or the dissimilarity between each shifted distribution $\hat{\boldsymbol{H}}^{\phi_i^{(a)}(a)}$ and the selected $\langle \hat{\boldsymbol{H}}_j, \mu_j \rangle$s. We obtain the k pairs with the lowest distance or dissimilarity and select the most occurring peak $\phi_i^{(a)}$ in the neighbors as the estimated karar $\kappa^{(a)}$.

Joint Estimation of Makam and karar: Given an audio fragment (a) with unknown makam and karar, we compute the stable pitches/pitch-classes, $\boldsymbol{\Phi}^{(a)} := \left\{\phi_1^{(a)}, \ldots, \phi_{|\boldsymbol{\Phi}^{(a)}|}^{(a)}\right\}$ and then the distributions $\hat{\boldsymbol{H}}^{\phi_i^{(a)},(a)}$ assuming each $\phi_i^{(a)} \in \boldsymbol{\Phi}^{(a)}$ as the karar candidate, as explained in the karar identification procedure explained above. Next, we compute the distance or the dissimilarity between each pair of shifted distribution $\hat{\boldsymbol{H}}^{\phi_i^{(a)},(a)}$ and the distributions $\hat{\boldsymbol{H}}_j$ in all the training samples $\langle \hat{\boldsymbol{H}}_j, \mu_j \rangle \in \boldsymbol{\mathcal{T}}$. We

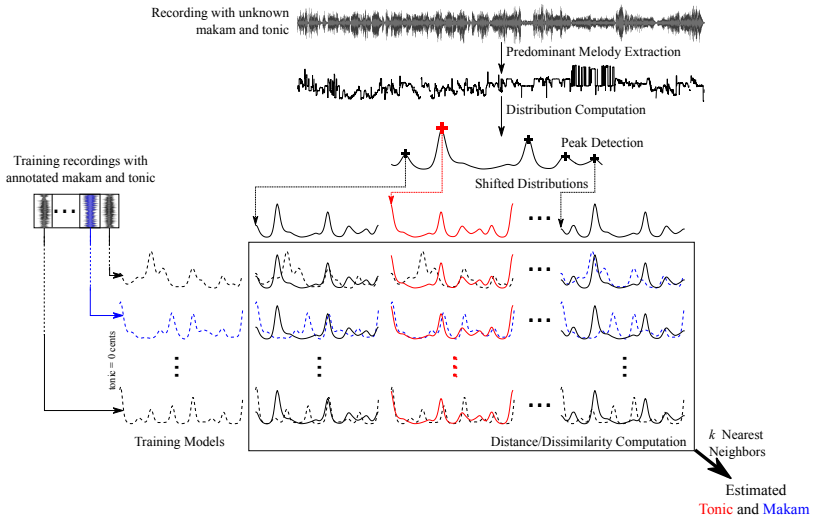


Figure 5.9: Block diagram of the joint estimation methodology. The shifted distribution in red and the training distribution in blue shows a close match.

select the k pairs with the lowest distance or dissimilarity and estimate the most occurring ⟨mode, karar candidate⟩ pair, i.e. $\langle \mu_i, \phi_j^{(a)} \rangle$ ($\mu_i \in \boldsymbol{\mathcal{M}}$, $\phi_j^{(a)} \in \boldsymbol{\Phi}^{(a)}$) as the makam $\mu^{(a)}$ and the karar $\kappa^{(a)}$ of the audio fragment (a). A procedure is shown in Figure 5.9.

5.7.3 Experiments

To evaluate the generalized methodology, we conducted exhaustive experiments on the largest makam recognition dataset of Ottoman-Turkish makam music (OTMM) (Section 5.7.3). In most of the karar identification and mode recognition studies in the past, the features extracted from the data,[25] the source code and the experimental results have not been generally shared.

We consider the unavailability of public tools, datasets and reproducible experimentations as major obstacles for computational research on OTMM and MIR in general. For this reason, we have implemented the generalized distribution matching methodology and packaged it as a open source toolbox called **MO**de **R**ecognition and **T**onic **Y**dentification Toolbox (Karakurt et al., 2016) (`MORTY`).[26] `MORTY` is free software writ-

[25] Excluding the commercial audio recordings, which cannot be generally made public due to copyright laws.

[26] `https://github.com/altugkarakurt/morty`

ten in *Python* 2.7 and licensed under *Affero GPLv3*.[27] Our primary aim is to provide open and flexible tools for the mode recognition and karar identification tasks, which can be applied to different music cultures while allowing the users to optimize the parameters easily according to the characteristics of the studied music.

This toolbox also includes the implementations of pitch and pitch-class distributions (Section 5.5)[28] and also the stable pitch and pitch-class extraction step (Section 5.6)[29] and hence provide the essential tools for several relevant tasks such as tuning and intonation analysis (Section 5.9 and 6.11). Moreover, `MORTY` has been recently used as a benchmark against novel methodologies proposed for mode recognition in Hindustaion and Carnatic music (Gulati, Serrà, Ishwar, et al., 2016; Gulati, Serrà, Ganguli, et al., 2016) (Appendix E.2).

In addition to the toolbox and experimentation code, the dataset (Section 5.7.3), the experiments (Section 5.7.3) and the results (Section 5.7.4) are available publicly via CompMusic website[30] for the sake of reproducibility.

Experimental Setup

In the experiments we use stratified 10-fold cross validation. Table 5.2 gives a combination of the parameters used in the experimental setup. We use grid search, to find the optimal parameters for OTMM. We use $\texttt{ATL-MEL}_f$ for predominant melody extraction (Atlı et al., 2014). The parameter combinations where the bin size b ($\hat{\boldsymbol{H}}$) is greater than or equal to 3 times the kernel width $\sigma(\hat{\boldsymbol{H}})$ are omitted. We also conduct experiments using the raw distributions, without smoothing. When the training model consists of a "single" distribution per mode, the number of neighbors, is always taken as 1 as each label is represented by a single data point. The minimum peak ratio, $\delta(\boldsymbol{H})$, is only used in karar identification and joint estimation tasks. The optimal value of the minimum peak ratio is found separately (Section 5.7.4).

For mode recognition, we mark the classification as $True$, if the estimated mode $\mu^{(a)}$ and the annotated mode $\mathfrak{m}^{(a)}$ for a recording (a) are the same. The karar octave in the orchestral performances of OTMM is ambiguous as each instrument plays the melody in their own register.

[27] https://www.gnu.org/licenses/agpl-3.0.en.html

[28] https://github.com/altugkarakurt/morty/blob/v1.2.1/morty/pitchdistribution.py

[29] https://github.com/altugkarakurt/morty/blob/v1.2.1/morty/pitchdistribution.py#L231-L265

[30] http://compmusic.upf.edu/node/319

Table 5.2: The summary of the tasks, features, training models and parameters used in the experiments

Symbol	Name	Values / Methods	Comment
	task	mode, karar, joint	
ϱ	predominant melody	ATL-MEL$_f$	extraction method specialized for OTMM
$\hat{\boldsymbol{H}}$	distribution	PD, PCD	
$\mathcal{T}$	type of the training model	single, multi	number of distribution per mode used in (Chordia & Şentürk, 2013; Gedik & Bozkurt, 2010)
$b\,(\hat{\boldsymbol{H}})$	bin size	$7.5, 15, 25, 50, 100$ cents	
$\sigma(\hat{\boldsymbol{H}})$	kernel width	"no smoothing" & $7.5, 15, 25, 50, 100$ cents	Combinations with $b\ (\hat{\boldsymbol{H}}) \geq 3\sigma(\hat{\boldsymbol{H}})$ are omitted.
$\diamondsuit$	distance or dissimilarity	L_1, L_2, L_3, Bhattacharyya, $1-$intersection, $1-$cross_correlation	$1-$intersection and $1-$cross_correlation are dissimilarities computed from the namesake similarity measures
k	number of nearest neighbors	$\{1, 3, 5, 10, 15\}$	for the "single" distribution per mode training model, the value is fixed to 1
$\delta(\boldsymbol{H})$	minimum peak ratio	$[0, 1]$	optimal found by a separate grid-search with a step of 0.05, is not used in makam recognition

Therefore, we aim to evaluate the karar pitch class and calculate the shortest octave-wrapped cent distance $\blacktriangle(\kappa^{(a)}, \mathfrak{k}^{(a)})$ between the estimated $\kappa^{(a)}$ and the annotated karar $\mathfrak{k}^{(a)}$ (Equation 5.3). If the distance is less than 25 cents, we consider the karar as correctly identified. In the case of joint estimation, we require both the karar and makam estimates to be correct.

For each fold, we compute the accuracy, which is the number of correct estimations divided by the total number of testing data. In Section 5.7.4, we report the average accuracies of the folds for each parameter combination. We also compare the karar identification results obtained in the karar identification and joint estimation tasks with the results obtained from the the last note detection method (Atlı et al., 2015) (Section 5.7.2). For all results below, the term "significant" refers to statistical significance at the $p = 0.01$ level as determined by a multiple comparison test using the Tukey-Kramer statistic.

To find an optimal for the minimum peak ratio, $\delta(\boldsymbol{H})$, we compute all distributions of each recording in the dataset using all the combinations of the bin sizes and the kernel widths given in Table 5.2. Then, we detect the peaks in each pitch distribution using a minimum peak ratio, from 0 (no threshold) to 1 (only keeping the highest peak). For each value of the minimum peak ratio, we note the number of distributions which has the annotated karar among the peaks ("tonic hits") and the total number of peaks obtained from each distribution. For the sake of reproducibility

all of the scripts, computed features, experiments and results are shared online.[31]

Test Dataset

In (Gedik & Bozkurt, 2010), the makam recognition method was evaluated on 172 solo audio recordings in 9 makams. To the best of our knowledge, this dataset represents the biggest number of recordings that has been used to evaluate makam recognition task, so far. As explained by the authors, these recordings were selected from the performances of "indisputable masters," and therefore they are musically representative of the covered makams. Nevertheless, the results are not reproducible as the dataset is not public.

The karar identification method proposed in (Bozkurt, 2008) was evaluated using 150 synthesized MIDI files plus 118 solo recordings. Similar to (Gedik & Bozkurt, 2010) the data is not pbulicly available. To the best of our knowledge, there exists only two open karar identification datasets for OTMM, both of which are compiled under the CompMusic project. The first one is provided in (Şentürk et al., 2013) (which will be explained in detail in Section 6.4) and it consists of 257 audio recordings. The second and the bigger test dataset is provided in (Atlı et al., 2015), consisting of 1093 recordings.[32] The recordings in both of the datasets are identified using MBIDs. Nevertheless, the features extracted from the audio recordings are not provided in either dataset. Therefore, the results are hard to reproduce.

Considering the lack of open test datasets for makam recognition and the drawbacks of the available karar identification datasets, we have gathered a test dataset of audio recordings with annotated makam and karar, called the *Ottoman-Turkish makam recognition dataset*.[33] The dataset covers 20 commonly performed makams[34] and it is composed of 1000 audio recordings. Following our constraint in the problem definition (Section 5.7.1), a single makam is performed in each recording (i.e. there are 50 recordings per makam). This dataset is currently the largest and the most comprehensive dataset for the evaluation of automatic makam

[31] https://github.com/sertansenturk/makam_recognition_experiments/tree/dlfm2016

[32] The datasets are hosted in https://github.com/MTG/turkish_makam_tonic_dataset/releases/

[33] https://github.com/MTG/otmm_makam_recognition_dataset/tag/v1.0.0

[34] i.e. Acemaşiran, Acemkürdi, Bestenigar, Beyati, Hicaz, Hicazkar, Hüseyni, Hüzzam, Karcığar, Kürdilihicazkar, Mahur, Muhayyer, Neva, Nihavent, Rast, Saba, Segah, Sultanıyegah, Suzinak and Uşşak

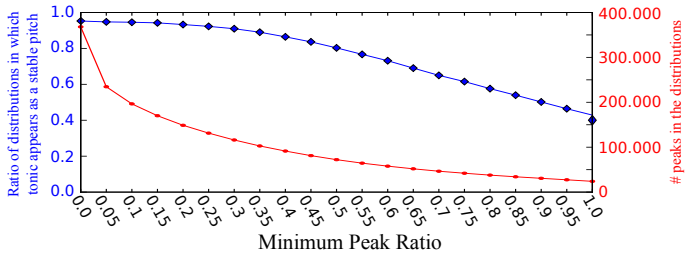


Figure 5.10: Total number of peaks and the ratio between the number of karar hits and number of all distributions.

recognition. Moreover, it is comparable to the aforementioned dataset provided in (Atlı et al., 2014) for the evaluation of karar identification methodologies.

Similar to (Atlı et al., 2015) and (Şentürk et al., 2013), the recordings in the dataset are labeled with MBIDs. The karar frequency of each recording is annotated manually by marking the karar frequency using the visual interface of Makam Toolbox. The karar frequency then is adjusted by synthesizing and listening the marked frequency synchronous to the audio playback using the same toolbox. The annotations are cross checked by at least two annotators. For the sake of reproducibility, we also provide the predominant melodies extracted from the audio recordings using $\texttt{ATL-MEL}_f$.

Similar to (Gedik & Bozkurt, 2010), the dataset is intended to be musically representative of OTMM. To achieve this, we selected the recordings of acknowledged musicians from the CompMusic makam corpus (Uyar et al., 2014), which is currently the most representative music corpus of OTMM aimed at computational research. The dataset covers contemporary and historical, monophonic and heterophonic recordings, as well as live and studio recordings. Some of the recordings have non-musical sections, such as clapping at the end of live recordings, announcements or scratch and hissing sounds (e.g. in historical recordings). This diversity gives us the opportunity to test the methods in a much more challenging environment, which hasn't been completely addressed in previous research (Gedik & Bozkurt, 2010).

5.7.4 Results

By inspecting Figure 5.10, we observe that the probability of finding the karar among the peaks is very high for minimum peak ratios less than 0.4 in the expense of an exponential increase in the karar candidates (peaks) and hence in the processing time. Since our scenario can toler-

ate a moderate increase in processing time, we select the minimum peak ratio, $\delta(\boldsymbol{H})$, as 0.15.

Table 5.3 shows the best results obtained after grid search. For mode recognition, multi-distribution per mode model yields an accuracy of 71.8% with the best parameter set while highest accuracy using single distribution per mode is 38.7%. For karar identification multi-distribution per mode performs with accuracy above 95% in 20 parameter sets and above 90% accuracy in 299 parameter sets out of 1440 experiments, where the highest accuracy obtained is 95.8%. Hence, the method is robust to a variety of parameter selections for karar identification. On the other hand, single distribution per mode model yields 89.8% accuracy with the best parameter set. For joint estimation the multi-distribution per mode model performs with 63.6% accuracy (86.1% karar identification and 65.2% makamrecognition accuracy) in the best configuration, while single distribution yields 27.6% (71.0% karar identification and 28.6% makamrecognition accuracy). For all three considered tasks, the optimal choices for distribution, distance/dissimilarity and training model are PCD, Bhattacharyya distance and multi-distribution per mode.

Table 5.3: The best parameter sets for each task. For all tasks PCDs using Bhattacharyya distance and traning multiple distributions per mode gives the best results.

Task	$\sigma(\hat{\boldsymbol{H}})$	$b\,(\hat{\boldsymbol{H}})$	**k**	**Accuracy**
Tonic	7.5	15	3	**95.8%**
Makam	25	25	10, 15	**71.8%**
Joint	15	7.5	15	**63.6%**

The method proposed in (Atlı et al., 2015) obtained 79.9% karar identification accuracy on our dataset. The best karar identification accuracy using PDs and single-distribution per mode as proposed in (Bozkurt, 2008) is 49.8%. Multi-distribution per mode method using PCDs outperforms both methods whether the makam is known (95.8% accuracy) or not (91.5% karar accuracy in joint estimation) with the majority of sub-optimal parameter sets. Figure 5.11 shows the distribution of the octave-wrapped cent distance (Equation 5.2) between the estimated and the annotated karar for each test and for all the parameter sets with 7.5 cent bin size.

These experiments revealed that certain parameter selections signifi-

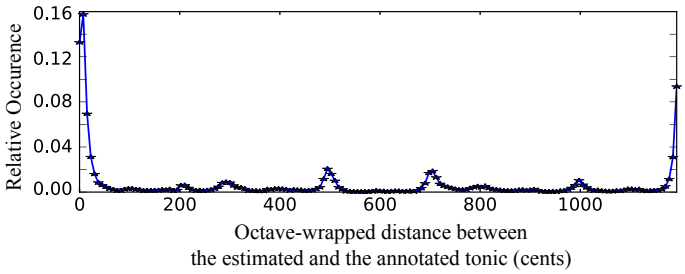


Figure 5.11: The distribution of octave-wrapped distances between the estimated and annotated karar for all parameter sets with 7.5 cent bin size.

cantly improve or diminish the methods' performances. These observations are listed below as a guidance:

- $\boldsymbol{\mathcal{T}}$**:** Multi-distribution training model (Chordia & Şentürk, 2013) performs significantly better than single-distribution training model (Gedik & Bozkurt, 2010).
- $\hat{\boldsymbol{H}}$**:** PCD significantly outperforms PD.
- b $(\hat{\boldsymbol{H}})$**:** Smaller bin size yields better results, however there is no significant distinction between 7.5, 15 and 25 cent bin sizes. Note that these bin sizes significantly outperform 50 and 100 cent bin sizes.
- $\sigma(\hat{\boldsymbol{H}})$**:** The 7.5, 15 and 25 cent kernel widths significantly improves the accuracy of the models compared to 50 and 100 cent kernel widths. No smoothing performs slightly worse than 7.5, 15 and 25 cent kernel widths. However, processing the distribution without smoothing is substantially slower due to the peak detection step.
- $\diamondsuit$**:** Using multi-distribution training model and PCDs, Bhattacharyya distance always yields the highest accuracy. It is significant except using $1-$intersection and $L1$ in karar identification.
- $\boldsymbol{k}$**:** Increasing the number of nearest neighbors increases the accuracy. Nevertheless, the increase does not make a significant impact except $k = 1$, which performs significantly worse than higher k values.
- $\delta(\boldsymbol{H})$**:** In the karar identification task, the true karar is typically among the detected peaks for minimum peak ratios below 0.4. Values smaller than 0.1 increases the processing time without any meaningful improvement in karar identification accuracy.

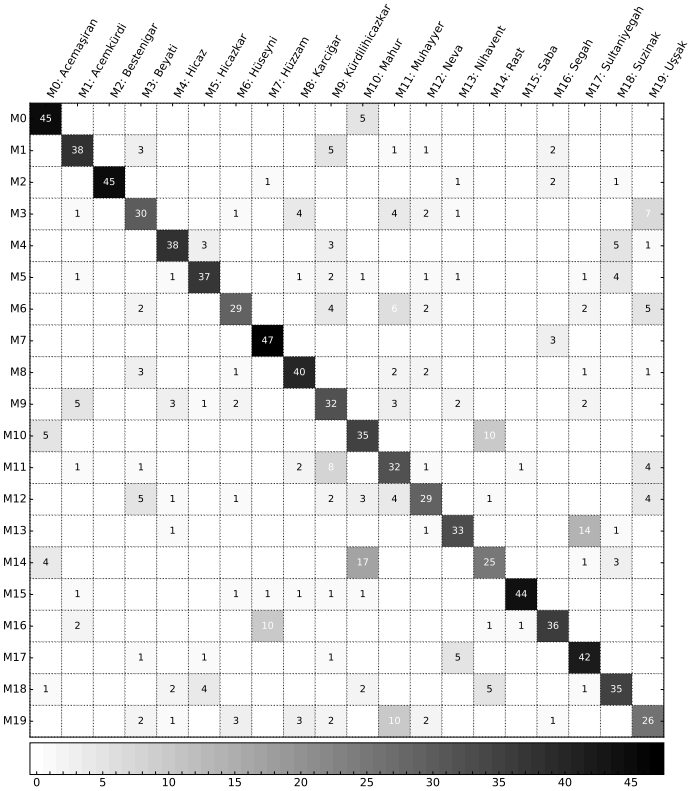


Figure 5.12: Confusion matrix of the best performing makam recognition experiment (Table 5.3).

5.7.5 Discussion

The drawback of the pitch distribution based methods is that they do not consider the temporal characteristics. When we inspected the results obtained from the experiments in 5.7.3, we observed that the confusions (Figure 5.12) are mainly between makams, which either have very similar intervals in their scale or contain similar sets of pitches. Similarly in (Gulati, Serrà, Ishwar, et al., 2016), the proposed method was better in classifying phrase-based ragas, while our method was better at classifying scale based ones (Appendix E.2).

In (Atlı et al., 2015), we showed that the last note detection method outperforms the karar identification method in (Bozkurt, 2008) (i.e. using PDs with single-model per mode) for OTMM. Our results validate these findings (the best is accuracy is 49.8 as stated in Section 5.7.4). Nevertheless, we show that using PCDs with multi-model per mode is superior to both methods even when the makam of the recording is not

known and even if the makam is found erroneously in the joint estimation process. While the estimated karar is typically around the annotation (Figure 5.11), the main confusion occurs around the fourth, fifth and seventh of the karar, which typically act as the melodic centers and/or anchor points in the melodic progression (Özkan, 2006).

For all the tasks defined in Section 5.7.1, we suggest using multi-distribution models approach with PCD and Bhattacharyya distance. If the estimation accuracy is a top priority, we suggest choosing a small b ($\hat{\boldsymbol{H}}$), $\sigma(\hat{\boldsymbol{H}})$ (7.5 or 15 cents) and $\delta(\boldsymbol{H})$ (0.15) as these parameters yield high accuracies. For applications requiring computational efficiency (e.g. mobile) or fast operation (e.g. real-time estimation), b ($\hat{\boldsymbol{H}}$), $\sigma(\hat{\boldsymbol{H}})$ (25 cents) and $\delta(\boldsymbol{H})$ (0.4) can be bigger, since reduced feature dimensions would substantially decrease the computational complexity. The number of neighbors may be chosen as any value higher than 1.

5.7.6 Summary

In this section, a generalized methodology based on (Bozkurt, 2008; Gedik & Bozkurt, 2010; Chordia & Şentürk, 2013) is presented. The methodology is implemented as an open toolbox for mode recognition and karar identification called `MORTY`. It is designed with flexibility in mind such that it can be easily modified and optimized to analyse large audio corpora. The implementation is evaluated on the largest makam recognition dataset of OTMM. The generalized method outperformed the state-of-the-art methodologies proposed for makam recognition (Gedik & Bozkurt, 2010) and karar identification (Bozkurt, 2008; Atlı et al., 2014). The toolbox has also been used to benchmark two novel mode recognition methodologies proposed for Indian art musics (Appendix E.2).

`MORTY` is also used as a part of our makam music analysis toolbox[35] in several tasks such as pitch and pitch-class distribution computation (Section 5.5), tuning analysis (Section 5.9) and melodic progression analysis (Section 5.10). The usage and implementation will be explained more in Section 5.11). In the future, we plan to apply dimension reduction and hashing techniques to summarize the features and speed up the classification for real-time mode and tonic estimation on short audio fragments. We would also like to incorporate the (Gulati, Serrà, Ganguli, et al., 2016), which has outperformed our methodology in mode recognition on Carnatic and Hindustani musics. We also hope that MORTY may be useful as a general tool for tonic identification, mode recognition and tuning analysis applied on different modal music traditions.

[35] `https://github.com/sertansenturk/tomato`

Table 5.4: The transpositions, the corresponding octave-wrapped cent distance intervals and the theoretical center of the pitch-class if the karar symbol is G4.

Ahenk	$\triangle(\kappa, \mathfrak{k}_{bolahenk})$	**Center when karar is G4**	
Bolahenk	$[-50, 50)$	293.67 Hz	
Davut-Bolahenk Mabeyni	$[50, 150)$	311.13 Hz	
Davut	$[150, 250)$	329.63 Hz	
Şah	$[250, 350)$	349.23 Hz	
Mansur-Şah Mabeyni	$[350, 450)$	370.00 Hz	
Mansur	$[450, 550)$	392.00 Hz	
Kız-Mansur Mabeyni	$[550, 650)$	415.31 Hz	$\times 2^n, \forall n \in \mathbb{Z}$
Kız	$[650, 750)$	440.01 Hz	
Yıldız	$[750, 850)$	466.17 Hz	
Müstahsen	$[850, 950)$	493.89 Hz	
Sipürde	$[950, 1050)$	523.26 Hz	
Bolahenk-Sipürde Mabeyni	$[1050, 1150)$	554.38 Hz	

5.8 Transposition

To obtain the transposition (ahenk) of an audio performance, the karar symbol of the makam of the recording is read by referring to a dictionary.[36] The theoretical frequency (according to the intervals defined by AEU theory) $\mathfrak{k}_{bolahenk}$ of the karar in Bolahenk (the default transposition of OTMM performances) is also noted.[37] The octave-wrapped cent distance $\triangle(\kappa, \mathfrak{k}_{bolahenk})$ from κ to $\mathfrak{k}_{bolahenk}$ is then computed (Equation 5.2). Finally, the transposition is matched by referring to the interval in Table 5.4, which contains the computed distance.[38] Note that the "Bolahenk Nısfiye" ahenk, which is an octave higher than Bolahenk, is omitted due to the ambiguity of karar octave in heterophonic recordings (Section 2.2).

5.9 Tuning

To analyze the tuning of each note performed in an audio fragment, we implemented the methodology explained in (Bozkurt et al., 2009).[39] The

[36] https://github.com/sertansenturk/ahenkidentifier/blob/v1.5.0/ahenkidentifier/data/tonic.json

[37] e.g. A4 $\approx$ 329.6 Hz, if the karar symbol of the makam is A4. Readers are reminded that the "typical" performance tuning of Western classical music (A4 = 440 Hz) is a fourth higher than Bolahenk (Section 2.2).

[38] The implementation is available at https://github.com/sertansenturk/ahenkidentifier.

[39] This work was done by Hasan Sercan Atlı and me between January and March 2016. We have equal contribution on implementing the methodology. I have also contributed to

method first obtains the set of stable pitches $\mathbf{\Phi}_P$ performed in an audio fragment (a) by applying peak detection on the pitch distribution as explained in (Section 5.6). The stable pitches are then normalized (Equation 5.1) with respect to the karar frequency κ (Section 5.7) and the performed scale degrees $\hat{\phi}^{\kappa}_{P,i} \in \hat{\mathbf{\Phi}}^{\kappa}_{P}$ (in cents) are obtained.

In parallel, the note symbols in the scale of the performed makam is inferred from the key signature of the makam and extended to $\pm$ two octaves. The note symbols are mapped to the theoretical scale degrees according to the AEU theory (Özkan, 2006) (e.g. if the karar symbol is G4, the scale degree of A4 is 9 Hc ≈ 203.8 cents).

Next, the performed scale degrees are matched with the theoretical scale degrees using a threshold of 50 cents (close to 2.5 Hc, which is reported as the optimal by Bozkurt et al. (2009)). If a performed scale degree is close to more than one theoretical scale degree (or vice versa), we only match the closest pair. As a trivial addition to (Bozkurt et al., 2009), we re-map the theoretical scale degrees to the note symbols and obtain the *stable pitch - note symbol* pairs, i.e. $\langle \phi_i, n_i \rangle$s. We also store the theoretical scale degree and the performed scale degree of each match.[40]

Figure 5.13 shows the extracted tuning over the pitch distribution of an audio recording in Hüseyni makam.[41] The frequency of each stable note is shown on the x-axis. The vertical dashed lines indicate the frequencies of the notes according to the theoretical intervals. The matched note symbol and the deviation from the theoretical scale degree of each stable pitch is displayed right next to the corresponding peak on the PD. It can be observed that the some of the notes (esp. çargah and hüseyni notes) substantially deviate from the AEU theory.

Note that the method explained in (Bozkurt et al., 2009) is limited to obtaining the tuning of the notes in the extended scale. Moreover, there is limited information about the intonation of the performed notes that can be retrieved from analysing the pitch distributions alone.[42] In Section 6.11, a score-informed method is proposed, which is able to capture the tuning and intonation of the majority of the performed notes.

---

the code with bug fixing, refactoring, documentation and deployment. Bilge Miraç Atıcı has also contributed to the initial stages of the development and also with bug fixing.

[40] Our implementation is available at `https://github.com/miracatici/notemodel`.

[41] `http://musicbrainz.org/recording/8b8d697b-cad9-446e-ad19-5e85a36aa253`

[42] e.g. the shape and spread of the peaks in the distributions

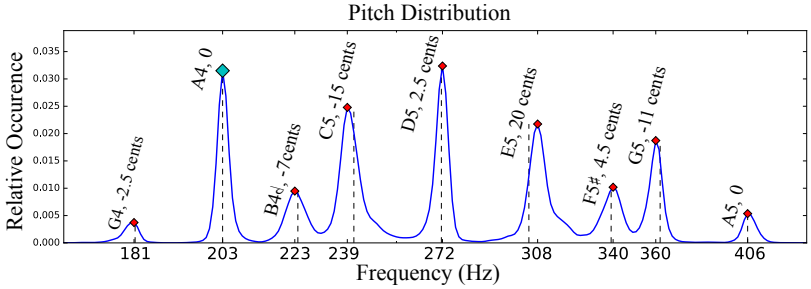


Figure 5.13: The tuning extracted from an audio recording in Hüseyni makam.

5.10 Melodic Progression

Bozkurt (2015) has proposed two similar models for analyzing the melodic progression (seyir) of music scores and audio recordings, respectively. For symbolic analysis, note sequence in the score is divided into small chunks. The note sequence in each chunk is synthesized (Section 4.2.2) by converting the note symbols to scale degrees according to the intervals defined by the AEU theory. Then the relative occurrence of the scale degrees and the mean of the scale degrees is computed for each chunk. For audio analysis, the predominant melody is divided into chunks and a pitch distribution is computed for each chunk. Finally the mean of the predominant melody of each chunk is computed. Bozkurt (2015) computes these representations from several sets of music scores or audio recordings grouped with respect to their makam to observe a "generalizable" melodic progression for each of the studied makams.

To analyse the melodic progressions in an audio fragment, these two approaches are combined by first extracting a predominant melody $\boldsymbol{\varrho}$ and dividing it to chunks with a certain frame size and overlap ratio. Then a PD $\boldsymbol{H}_P$ (Section 5.5) is computed for each chunk and the set of stable pitches $\boldsymbol{\Phi}_P$ are extracted from the PD of each chunk (Section 5.6).[43]

So far the melodic progression is used for visualization in the recording pages of Dunya-makam (Section 7.1.1). For consistency in visualization, we divide an input audio recording into 40 chunks with a 50% overlap. Figure 5.14, show the predominant melody (in green) extracted from an audio recording and along with the melodic progression. The black line indicates the mean of the predominant melody in each chunk. The dots show the location of the stable pitches. The radius of the dots are proportional to the height of the corresponding peak in the PD computed

[43] The implementation is available at `https://github.com/sertansenturk/seyiranalyzer`.

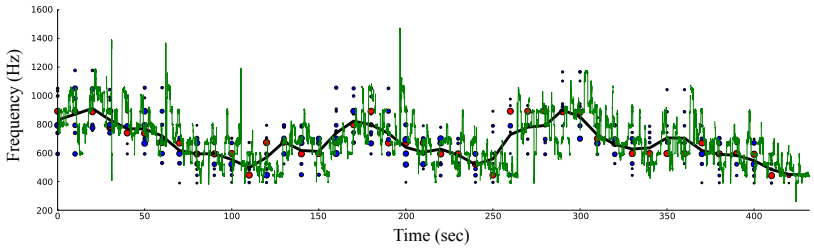


Figure 5.14: The predominant melody and melodic progression feature of an audio recording.

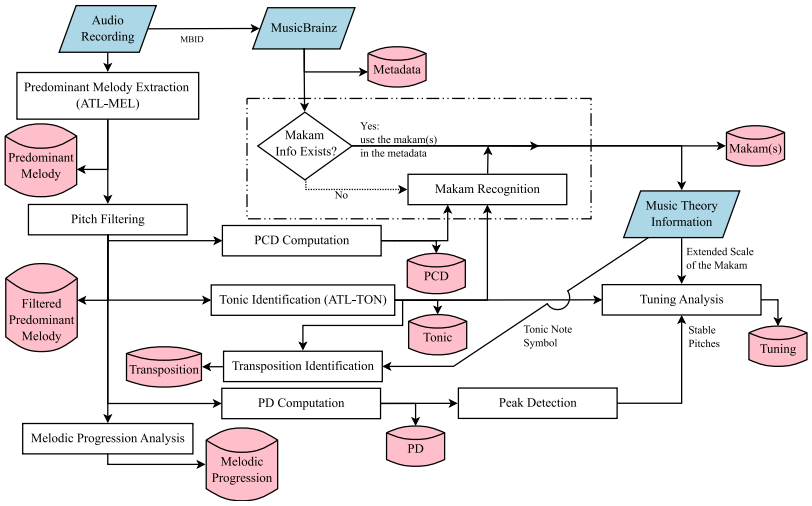


Figure 5.15: The

for the chunk and the red dot corresponds to the stable pitch extracted from the highest peak of the PD in its chunk. Note that the computed PDs are not displayed to avoid cluttering the Figure.

5.11 Combining Audio Analysis XX

Bunlar nasil bir araya gelecek, ne sirayla kullanilacak?

The algorithms can be called sequentially as shown in FigureXX using the audio analysis module in **T**urkish-**O**ttoman **M**akam (M)usic **A**nalysis **TO**olbox (`tomato`).[44] An overview of the implementations, please refer to Appendix D

Figure 5.15 shows the sequence of the analysis steps. The XX is implemented such that theory-agnostic steps such as the predominant melody

[44] `https://github.com/sertansenturk/tomato/tree/master/tomato/audio`

extraction (Section 5.2), and karar identification (Section 5.7) are computed for any audio input. On the other hand, some analysis step would be partially or completely unavailable, if the recording lacks the required metadata (e.g. transposition identification will fail for recordings with no makam metadata) or the music theory knowledge related to the metadata is incomplete (stable pitches will not be matched with the note symbols in tuning analysis, if the scale of the makam of the recording is not known).

5.12 Analysis of the CompMusic makam audio corpus

StatisticsXX

Add the Figure in the Turkey presentation where each feature is linked with statistics

Some of these will be improved with the joint analysis, explained in the next Chapter.

5.13 Conclusion

Summary XX

Other potential use cases of the melodic progression analysis are yet to be explored

Transcription

Adding Sankalp's method for makamrecognition. The mode recognition using the feature proposed in (Gulati, Serrà, Ganguli, et al., 2016) is able to capture both of these properties better with a slight increase in computational complexity.

Formal evaluation of Predominant melody

ADaptive tuning

Chapter 6

Joint Audio-Score Analysis

The most relevant representations of music are notations and audio recordings, each of which emphasizes a particular perspective and promotes different approximations in the analysis and understanding of music. Linking these two representations and analyzing them jointly should help to better study many musical facets by being able to combine complementary analysis methodologies. In order to develop accurate linking methods, we have to take into account the specificities of a given type of music. In this paper, we present a method for linking musically relevant sections in a score of a piece from *makam* music of Turkey (MMT) to the corresponding time intervals of an audio recording of the same piece. The method starts by extracting relevant features from the score and from the audio recording. The features of a given score section are compared with the features of the audio recording to find the candidate links in the audio for that score section. Next, using the sequential section information stored in the score, it selects the most likely links. The method is tested on a dataset consisting of instrumental and vocal compositions of OTMM, achieving 92.1% and 96.9% F_1-scores on the instrumental and vocal pieces, respectively. Our results show the importance of culture-specific and knowledge-based approaches in music information processing.

In analyzing a music piece, scores provide an easily accessible symbolic description of many relevant musical components. The audio recordings can provide information about the characteristics (e.g. in terms of dynamics or timing) of an interpretation of a particular piece. Parallel information extracted from score and audio recordings may facilitate computational tasks such as version detection (Arzt, Böck, & Widmer, 2012), source separation (Ewert & Müller, 2012), automatic accompaniment (Cont, 2010) and intonation analysis (Devaney, Mandel, & Fuji-

naga, 2012) add jazz intonation ismirXX.

When score is available, several tasks can be improved, facilitate. We override/recompute some of the features after the joint analysis such as tonic, predomiant melody, seyir model, pitch and pitch-class distributions and note models.XX

6.1 Nomenclature

Doldurma cumlesi XX

1. Let $f(x)$ be a fragment selected from an audio recording or a music score x. The fragment is basically all of the contents of the audio recording or the music score in the time-interval $t\big(f^{(x)}\big) = \big[t_{ini}\big(f^{(x)}\big)\ t_{fin}\big(f^{(x)}\big)\big]$.The fragment can refer to certain a event, a sequence of events or it can be selected arbitrarily, e.g. the first instance of the Teslim section in a score, a fragment between 50^{th} and 70^{th} seconds of an audio recording.

2. Let $\bar{\mathbf{N}}^{(b)}$ the note sequence in the music score (b). Each $\bar{n}_j^{(b)} \in \bar{\mathbf{N}}^{(b)}$ (where $j \in \big[1 : |\bar{\mathbf{N}}^{(b)}|\big]$) consists of a $\Big\langle n_j^{(b)}, d\Big(n_j^{(b)}\Big)\Big\rangle$ tuple, the elements of which represent the note symbol, note duration associated with the note, respectively.

3. We define the section sequence in the music score (b) as $\bar{\mathbf{S}}^{(b)} := \Big[\bar{s}_1^{(b)}, \ldots, \bar{s}_{|\bar{\mathbf{S}}^{(b)}|}\Big]$, with each $\bar{s}_i^{(b)}$ consisting of a *section symbol*, $s_i^{(b)}$, and a note sequence $\bar{\mathbf{N}}^{\left(\bar{s}_i^{(b)}\right)} = \left[\bar{n}_1^{\left(\bar{s}_i^{(b)}\right)},\ \bar{n}_2^{\left(\bar{s}_i^{(b)}\right)}, \ldots\right]$, where $\bar{\mathbf{N}}^{\left(\bar{s}_i^{(b)}\right)}$ is a subsequence of $\bar{\mathbf{N}}^{(b)}$. The note sequence of the sections with the same section symbol (e.g. repetitive sections) do not have to be identical due to different ending measures, volta brackets etc.

4. The sections in the score form the *score section symbol sequence*, $\mathbf{S}^{(b)} := \Big[s_1^{(b)}, \ldots, s_{|\mathbf{S}^{(b)}|}^{(b)}\Big]$, where $s_i^{(b)} \in \mathcal{S}_s$ and $i \in \big[1 : |\mathbf{S}^{(b)}|\big]$, with $|\mathbf{S}^{(b)}|$ being the number of sections in a score, repeated sections are counted individually.

5. Let $\mathcal{S}^{(b)} = \Big\{\mathcal{S}_s{}^{(b)}, unrelated\Big\}$ denote the *set of section symbols*. It consists of a set of symbols $\mathcal{S}_s{}^{(b)} := \big\{\mathbf{S}^{(b)}\big\}$, which represents all the $|\mathcal{S}_s{}^{(b)}|$ possible distinct section symbols in the composition; and an "$unrelated$" section, i.e. a segment with content not related to

any structural element of the musical form. The number of unique sections is $|\mathcal{S}| = |\mathcal{S}_s{}^{(b)}| + 1$.

6. Analogous, for the audio recording (a) we have the (true) *audio section sequence*, $\bar{\mathfrak{S}}^{(a)} = \left[\bar{\mathfrak{s}}_1^{(a)}, \ldots, \bar{\mathfrak{s}}_{|\bar{\mathfrak{S}}^{(a)}|}\right]$. Each element of the sequence, $\bar{\mathfrak{s}}_i^{(a)}$ ($i \in [1 : |\bar{\mathfrak{S}}^{(a)}|]$), has the section symbol, $\mathfrak{s}_i^{(a)}$, and covers a time interval in the audio, $t\left(\bar{\mathfrak{s}}_i^{(a)}\right)$, i.e. $\bar{\mathfrak{s}}_i^{(a)} = \left\langle \bar{\mathbf{N}}^{\left(\bar{\mathfrak{s}}_i^{(a)}\right)},\ \mathfrak{s}_i^{(a)},\ t\left(\bar{\mathfrak{s}}_i^{(a)}\right) \right\rangle$. The time interval is given as $t\left(\bar{\mathfrak{s}}_i^{(a)}\right) = \left[t_{ini}\left(\bar{\mathfrak{s}}_i^{(a)}\right)\ t_{fin}\left(\bar{\mathfrak{s}}_i^{(a)}\right)\right]$, where $t_{ini}\left(\bar{\mathfrak{s}}_1^{(a)}\right) = 0$ sec; $t_{fin}\left(\bar{\mathfrak{s}}_i^{(a)}\right) = t_{ini}\left(\bar{\mathfrak{s}}_{i+1}^{(a)}\right), \forall i \in [1 : |\bar{\mathfrak{S}}^{(a)}| - 1]$; and $t_{fin}\left(\bar{\mathfrak{s}}_{|\bar{\mathfrak{S}}^{(a)}|}^{(a)}\right)$ refers to the end of the audio recording.

7. For the audio recording we have the (true) *audio section symbol sequence*, $\mathfrak{S}^{(a)} = \left[\mathfrak{s}_1^{(a)}, \mathfrak{s}_2^{(a)}, \ldots\right]$, where $\mathfrak{s}_i^{(a)} \in \mathcal{S}$, $i \in \left[1 : |\mathfrak{S}^{(a)}|\right]$, with $|\mathfrak{S}^{(a)}|$ being the number of sections in the performance, including possibly multiple unrelated sections.

6.2 Melodic Feature Extraction

Score and audio recording are different ways to represent music. To compare these information sources, features that capture the melodic content given in each representation are extracted.

Thoughout the Chapter, predominant melody is typically extracted from the audio recordings (Section 5.2). As mentioned in Section 5.2, either of `SEN-YIN`$_f$, `SEN-MEL` or `ATL-MEL` is utilized in the experiments. In parallel, synthetic melody is extracted from the music scores (Section 4.2.2). The pitch intervals in the synthetic melody are computed either according to the performed tuning (Section 5.9) extracted from the audio recording to be aligned (Appendix B) or according to AEU theory (Section 6.7.2).

In section linking experiments (Section 6.7.2), the predominant melody (Section 5.2) and the synthetic melody (Section 4.2.2) is compared with the HPCPs (Section 5.3) and the synthetic HPCPs (Section 4.2.3), respectively.

The details of the relevant extraction methods will be given in the respective Sections. Figure 6.1 shows the a score except and an audio

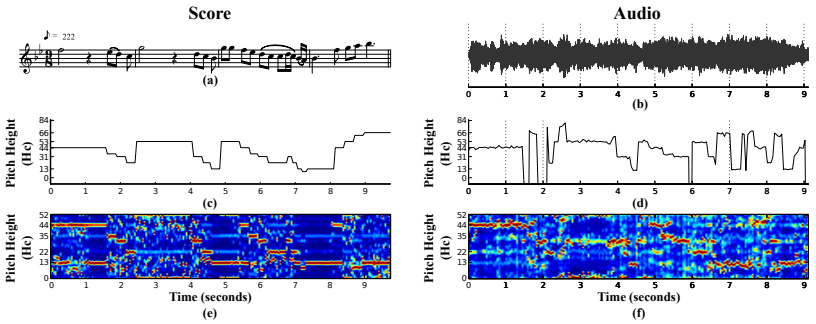


Figure 6.1: Score and audio representations of the first nakarat section of *Gel Güzelim* and the features computed from these representations. a) Score. b) Annotated section in the audio recording. c) Synthetic predominant melody computed from the note symbols and durations. d) predominant melody computed from the audio recording using SEN-MEL. The end of the predominant melody has a considerable number of octave errors. e) HPCPs computed from the synthesized MIDI. f) HPCPs computed from the audio recording.

waveform[1] of the first nakarat section of the composition, *Gel Güzelim*.[2] along with the features extracted from these sources.

6.3 Fragment Linking

In several tasks such as tonic identification (Figure 6.4) and composition identification (Figure 6.6), a complete alignment between the audio recording and music score is not necessary. On the other hand, structural differences between the music score and audio performance of OTMM (Section 2.2) have to be taken care of for complete alignment. This problem is approached from a bottom-up scheme by first aligning the sections in the music score separately and then estimating the global alignment from the individual paths (Section 6.7). Therefore, obtaining partial alignments between the audio recording and the score fragment(s) is the fundamental step in all audio-score alignment tasks described hereafter.

Fragment linking is defined as "marking the fragments, $[\bar{f}_1^{(a)}, \bar{f}_2^{(a)}, \dots]$, in the audio recording (a) at which a fragment $\bar{f}^{(b)}$ selected from the music score (b) are performed." This process forms a link, $\pi(\bar{f}_k^{(a)}, \bar{f}^{(b)})$,

[1] http://musicbrainz.org/recording/e7be8c2a-3309-4106-93b7-76cd6102a924

[2] http://musicbrainz.org/work/9aaf5c0b-4642-40fd-97ba-c861265872ce

between the score fragment $\bar{f}^{(b)}$ and each audio fragment $\bar{f}_k^{(a)}$ such that the start t_{ini} and the end t_{fin} of the linked fragments are mapped to each other (meinard gosterimi XX). Linking the audio and score fragments also imply relating the relevant attributes of each source, if available, e.g. the tonic symbol $\kappa^{(b)}$ of the score and the tonic frequency $\kappa^{(a)}$ of the audio recording. Similarly, additional useful information may be obtained from the linking process, e.g. the intra-alignment path, $\varpi^{(\bar{f}_k^{(a)},\bar{f}^{(b)})}$ between the fragments $\bar{f}_k^{(a)}$ and $\bar{f}^{(b)}$, which can be used later in note-level alignment (Section 6.8).

Two different methods are used for fragment linking: 1) The Hough transform (Section 6.3.1), and 2) Subsequence dynamic time warping (Section 6.3.2). Until the end of Section 6.7, an audio recording[3] of the composition *Şedaraban Sazsemaisi*[4] is used for illustration. Throughout this Section, it is assumed that the tonic pitch/pitch-class of the audio recording, $\kappa^{(a)}$, is already identified. Tonic identification step will be explained in Section 6.4.

6.3.1 Hough Transform

Given an audio recording (a) of a composition and a fragment $\bar{f}^{(b)}$ selected from the music score (b) of the same composition, we first extract either the predominant melody or the HPCPs from the audio recording. The audio feature is then normalized with respect to the performance tonic, $\kappa^{(a)}$, and a normalized predominant melody, $\hat{\boldsymbol{\varrho}}^{\kappa^{(a)},(a)}$, or HPCPs, $\hat{\boldsymbol{\Gamma}}^{\kappa^{(a)},(a)}$, is obtained. In parallel, a synthetic melody, $\hat{\boldsymbol{\Psi}}^{(\bar{f}^{(b)})}$, or synthetic HPCPs, $\hat{\boldsymbol{\Omega}}^{(\bar{f}^{(b)})}$, are computed. To compare the audio recording with each section in the score, we compute a distance matrix between the score feature, $\hat{\boldsymbol{\Psi}}^{(\bar{f}^{(b)})}$ or $\hat{\boldsymbol{\Omega}}^{(\bar{f}^{(b)})}$, of the fragment $(\bar{f}^{(b)})$ and the audio feature, $\hat{\boldsymbol{\varrho}}^{\kappa^{(a)},(a)}$ or $\hat{\boldsymbol{\Gamma}}^{\kappa^{(a)},(a)}$, of the whole recording, for either predominant melodies or HPCPs, respectively. distance matrix shows blobs.

If predominant melodies are chosen as the features, the distance matrix, $\boldsymbol{D}^{\kappa^{(a)},(a,\bar{f}^{(b)})}$, between the normalized audio predominant melody, $\hat{\boldsymbol{\varrho}}^{\kappa^{(a)},(a)}$, and the synthetic predominant melody, $\hat{\boldsymbol{\Psi}}^{(\bar{f}^{(b)})}$, of a score fragment $(\bar{f}^{(b)})$ is obtained by computing the pairwise smallest Hc distance between each point of the features using Equation 5.3. To recapitulate,

[3] http://musicbrainz.org/recording/efae832f-1b2c-4e3f-b7e6-62e08353b9b4

[4] http://musicbrainz.org/work/1eb2ca1e-249b-424c-9ff5-0e1561590257

each element $\boldsymbol{D}_{ij}^{\kappa^{(a)},(a,\bar{f}^{(b)})}$ in the distance matrix $\boldsymbol{D}^{\kappa^{(a)},(a,\bar{f}^{(b)})}$ is computed as:

$$\boldsymbol{D}_{ij}^{\kappa^{(a)},(a,\bar{f}^{(b)})} = \blacktriangle\left(\hat{\psi}_i^{(\bar{f}^{(b)})}, \hat{\rho}_j^{\kappa^{(a)},(a)}\right) \tag{6.1}$$

where $\hat{\psi}_i^{(\bar{f}^{(b)})}$ is the i^{th} sample of the synthetic melody $\hat{\boldsymbol{\Psi}}^{(\bar{f}^{(b)})}$ computed from the score fragment $\bar{f}^{(b)}$ (Section 4.2.2), $\hat{\rho}_j^{\kappa^{(a)},(a)}$ is the j^{th} sample of the predominant melody $\hat{\boldsymbol{\varrho}}^{\kappa^{(a)},(a)}$ normalized with respect to the tonic $\kappa^{(a)}$ of the audio recording (a) (Section 5.2), and $\blacktriangle(x, y)$ denotes the shortest octave-wrapped distance between the pitch values x and y (Section 5.4). The first element in the superscript, $\kappa^{(a)}$, indicates that the predominant melody, $\hat{\boldsymbol{\varrho}}^{\kappa^{(a)},(a)}$, extracted from (a) is normalized to cent-scale by taking the tonic $\kappa^{(a)}$ as the reference. $\boldsymbol{D}^{\kappa^{(a)},(a,\bar{f}^{(b)})}$ is a $\left|\hat{\boldsymbol{\varrho}}^{\kappa^{(a)},(a)}\right|$ x $\left|\hat{\boldsymbol{\Psi}}^{(\bar{f}^{(b)})}\right|$ matrix. Figure 6.2e shows a distance matrix computed for the Teslim *Şedaraban Sazsemaisi*.

If HPCPs are chosen as the features, the distance matrix, $\boldsymbol{D}^{\kappa^{(a)},(a,\bar{f}^{(b)})}$, between the normalized audio HPCPs, $\hat{\boldsymbol{\Gamma}}^{\kappa^{(a)},(a)}$, and the synthetic HPCPs, $\hat{\boldsymbol{\Omega}}^{(\bar{f}^{(b)})}$, of a score fragment, $(\bar{f}^{(b)})$, is obtained by taking the *cosine* distance between each HPCP frame. Cosine distance is a common feature used for comparing chroma features (Paulus et al., 2010) computed as:

$$\boldsymbol{D}_{ij}^{\kappa^{(a)},(a,\bar{f}^{(b)})} = 1 - \frac{\sum_{k=1}^{n_{\text{HPCP}}} \hat{\omega}_{ik}^{(\bar{f}^{(b)})}\ \hat{\gamma}_{jk}^{\kappa^{(a)},(a)}}{\sqrt{\left(\sum_{k=1}^{n_{\text{HPCP}}} \left(\hat{\omega}_{ik}^{(\bar{f}^{(b)})}\right)^2\right).\left(\sum_{k=1}^{n_{\text{HPCP}}} \left(\hat{\gamma}_{jk}^{\kappa^{(a)},(a)}\right)^2\right)}},$$
$$1 \leq i \leq |\hat{\boldsymbol{\Omega}}^{(\bar{f}^{(b)})}| \text{ and } 1 \leq j \leq |\hat{\boldsymbol{\Gamma}}^{\kappa^{(a)},(a)}| \tag{6.2}$$

where $\hat{\omega}_{ik}^{(\bar{f}^{(b)})}$ is the k^{th} bin of the i^{th} frame of the HPCPs (of $|\hat{\boldsymbol{\Omega}}^{(\bar{f}^{(b)})}|$ frames) of a fragment $(\bar{f}^{(b)})$, $\hat{\gamma}_{jk}^{\kappa^{(a)},(a)}$ is the k^{th} bin of the j^{th} frame of the HPCPs (of $|\hat{\boldsymbol{\Gamma}}^{\kappa^{(a)},(a)}|$ frames) extracted from the audio recording (a) and n_{HPCP} denotes the number of bins chosen for the HPCP computation. Cosine distance between two HPCP frames of length n_{HPCP} can be interpreted as 1 minus the dot product of the two frames, normalized to unit length on an n_{HPCP}-dimensional Euclidean space. The outcome is bounded to the interval $[0\ 1]$ for non-negative inputs, 0 denoting the "closest," which makes it possible to compare the relative distance between the frames of HPCPs that have unitless, non-negative values.

If the score fragment is performed in the audio, the distance matrix shows blob(s) in a diagonal trajectory formed by low distance values, which hint the location(s) of the score fragment in the audio (Figure 6.2e). The values of the points forming the blobs may be substantially greater than zero in practice, making it harder to distinguish the blobs from the background. Therefore, binary thresholding is applied to the distance matrix before applying the Hough transform to emphasize the diagonal blobs. A binary similarity matrix $\boldsymbol{B}^{\kappa^{(a)},(a,\bar{f}^{(b)})}$ is obtained as:

$$\boldsymbol{B}_{ij}^{\kappa^{(a)},(a,\bar{f}^{(b)})} = \begin{cases} 1, & \boldsymbol{D}_{ij}^{\kappa^{(a)},(a,\bar{f}^{(b)})} \leq \beta \\ 0, & \boldsymbol{D}_{ij}^{\kappa^{(a)},(a,\bar{f}^{(b)})} > \beta \end{cases} \tag{6.3}$$

where β is the binarization threshold. The binary similarity matrix $\boldsymbol{B}_{ij}^{\kappa^{(a)},(a,\bar{f}^{(b)})}$ of a fragment $\bar{f}^{(b)}$ shows which points between the score feature and the audio feature are similar enough to each other to be deemed as the same note (Figure 6.2f). For melody, the binarization threshold correspond to the minimum value of shortest octave wrapped distance in cents. For HPCPs, the binarization threshold is unitless in the interval $[0 : 1]$. Effect of the binarization threshold value is investigated in the section linking experiments (Section 6.7.2). In the preliminary section linking experiments, we additionally used a number of structural morphological operations (Serra, 1982; Ballard, 1981) to emphasize the blobs. They are explained in Appendix B separately for the sake of simplicity.

As can be seen in Figure 6.2f, these blobs can be approximated as line segments. To detect these segments, a common line detection method called the Hough transform is used (Ballard, 1981). Hough explanationXX. The Hough transform has also been previously used in musical tasks such as locating the formant trajectories of drum beats (Townsend & Sandler, 1993) and detecting repetitive structures in an audio recording for thumbnailing (Aucouturier & Sandler, 2002).

To detect the line segments, the Hough transform is applied to the binary similarity matrix (Figure 6.2g). The searched angles are restricted between $-26.57°$ and $-63.43°$, which allows the alignment to have a tempo deviation between 0.5 and 2 times the nominal tempo indicated in the **SymbTr**-score. Then, the peaks are selected from the obtained transformation matrix. These peaks indicate the angle and the distance to the origin of the most prominent line segments (Duda & Hart, 1972). Next, the line segments (i.e. the linear alignment paths, $\varpi^{\left(\bar{f}_k^{(a)},\bar{f}^{(b)}\right)}$ between the score fragment $\bar{f}^{(b)}$ and the estimated audio fragments $\bar{f}_k^{(a)}$ are computed from this set of peaks such that the line segment covers the entire duration of the section given in the score (Figure 6.2g).

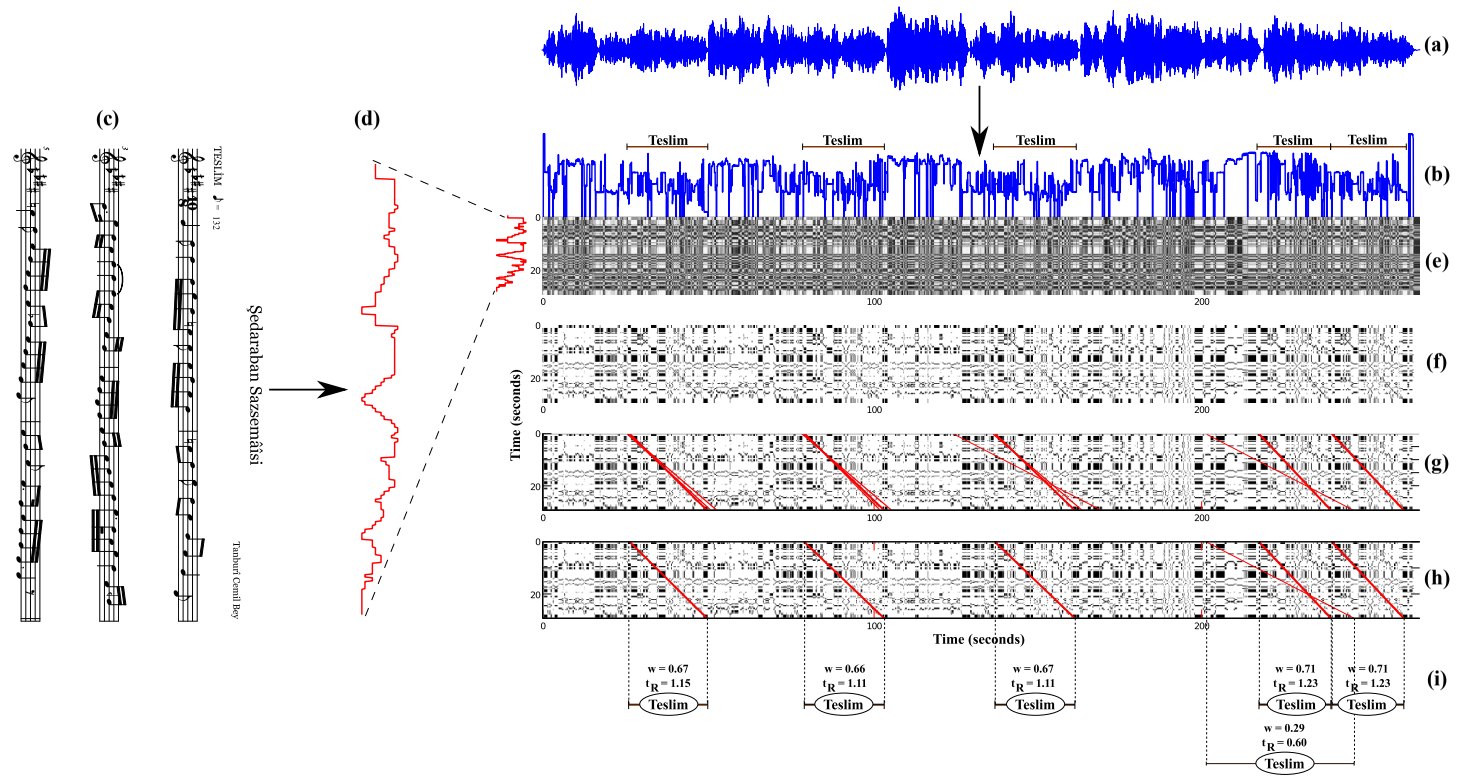


Figure 6.2: Steps in linking the Teslim section of the *Şedaraban Sazsemaisi* and an audio recording of the composition using the Hough transform for alignment, and predominant melody and synthetic melody as the input features. **a)** Audio waveform. **b)** Predominant melody extracted from the audio recording. Annotated Teslims are shown over the predominant melody. **c)** Teslim section of the *Şedaraban Sazsemaisi* **d)** Synthetic pitch computed from the Teslim section. **e)** Distance matrix between the predominant melody and the synthetic melody. White indicates the closest distance (0 Hc). **f)** Image binarization on distance matrix. White and black represent zero (dissimilar) and one (similar) respectively. **g)** Line detection using the Hough transform. **h)** Elimination of duplicates. **i)** Teslim candidates. The numerical values "w" and "τ_R" indicate the similarity and the relative tempo of the candidate respectively.

6.3.2 Subsequence Dynamic Time Warping

6.3.3 Similarity Computation

Using either the Hough transform or subsequence dynamic time warping (SDTW), we obtain a set of paths $\left\{ \varpi^{\left(\bar{f}_1^{(a)}, \bar{f}_2^{(b)}, \ldots\right)}, \right\}$. Each path $\varpi^{\left(\bar{f}_k^{(a)}, \bar{f}^{(b)}\right)}$ is denoted as:

$$\varpi^{\left(\bar{f}_k^{(a)}, \bar{f}^{(b)}\right)} = \left[\varpi_1^{\left(\bar{f}_k^{(a)}, \bar{f}^{(b)}\right)}, \ldots, \varpi^{\left(\bar{f}_k^{(a)}, \bar{f}^{(b)}\right)}_{\left|\varpi^{\left(\bar{f}_k^{(a)}, \bar{f}^{(b)}\right)}\right|} \right] \tag{6.4}$$

between the audio fragment $\bar{f}_k^{(a)}$ and the score fragment $\bar{f}^{(b)}$ with $\varpi_l^{\left(\bar{f}_k^{(a)}, \bar{f}^{(b)}\right)} = \left(r_l^{\left(\bar{f}_k^{(a)}\right)}, q_l^{\left(\bar{f}^{(b)}\right)} \right)$. $r_l^{\left(\bar{f}_k^{(a)}\right)} \in [1, I^m]$, $q_l^{\left(\bar{f}^{(b)}\right)} \in [1, J^m]$ and $l \in \left[1 : \left|\varpi^{\left(\bar{f}_k^{(a)}, \bar{f}^{(b)}\right)}\right|\right]$, where $\left|\varpi^{\left(\bar{f}_k^{(a)}, \bar{f}^{(b)}\right)}\right|$ is the length of the path $\varpi^{\left(\bar{f}_k^{(a)}, \bar{f}^{(b)}\right)}$. We compute a similarity, $s^{(m,n,k)} \in [0:1]$, between the score fragment and the audio recording for the tonic candidate $c_k^{(m)}$ by:

$$s^{(m,n,k)} = \frac{\sum_l B(r_l^{\left(\bar{f}_k^{(a)}\right)}, q_l^{\left(\bar{f}^{(b)}\right)})}{\left|\varpi^{\left(\bar{f}_k^{(a)}, \bar{f}^{(b)}\right)}\right|} \tag{6.5}$$

The number of non-zero pixels forming the line segment is normalized by the length of the line segment, giving the similarity $w(\bar{\pi}_k)$ of the segment.

6.3.4 Duplicate Link Removal

If two or more links have their borders in the same vicinity (± 6 seconds), they are treated as duplicates. This occurs frequently using the Hough transform since the line segments in the binary matrix are actually blobs. Hence, there might be line segments with slightly different parameters, effectively estimating the same candidate. Among the duplicates, only the one with the highest similairty is kept (Figure **??**d).

The regions covered by the remaining lines are chosen as the candidate time intervals, $t(\bar{\pi}_k) = [t_{ini}(\bar{\pi}_k)\ t_{end}(\bar{\pi}_k)]$ in seconds, for the particular section (Figure **??**e).

6.4 Score-Informed Tonic Identification

A novel method to identify the tonic using partial audio-score alignment is introduced in (Şentürk et al., 2013). First, the stable pitch classes are extracted from pitch class distribution of the recording (Section 5.9). Next, the predominant melody of the audio recording is normalized assuming each stable pitch class as a tonic candidate and the partial alignment method explained in (Şentürk, Holzapfel, & Serra, 2014) (Section 6.3) is applied between the score fragment and the audio recording. In the original method, the repetitive section is selected as the score fragment. For each tonic candidate the confidence of the alignments are accumulated. The candidate which accumulates the highest confidence value is estimated as the tonic. This method is shown to obtain near perfect results with a single error in a dataset composed of 257 recordings (Şentürk et al., 2013).

Some of the forms performed our corpus either don't have any repetitive sections or have multiple repetitive sections. For this reason, we compute the synthetic pitch from the start of the instrumental scores, and the first vocal section in the vocal scores. The first vocal section is preferred over the start in vocal compositions, because the instrumental prelude (tr: aranağme) sections given in the scores of the vocal compositions might be omitted or replaced with an improvisation in performances. Apart from changing the location of the selected fragment, we also experiment with the score fragment duration to improve the processing speed (Section **??**).

6.5 Score-Informed Tempo Estimation

The partial score alignment method in (Şentürk, Holzapfel, & Serra, 2014) (Section 6.3) is used as a preliminary step in (Holzapfel, Şimşekli, Şentürk, & Cemgil, 2015), to align the start of the recording with the start of the score. The relative tempo is simply inferred as the ratio between the duration of the the aligned audio fragment and the score fragment. The absolute tempo is computed by multiplying the relative tempo estimation with the nominal tempo given in the score. This method is reported to input reliable tempo estimations. We have incorporated this method into the score-informed tonic identification method (Section 6.4), by computing the tempo from the most confident alignment obtained from the estimated tonic[5].

[5]The compiled binary for joint tonic identification and tempo estimation is available at (link suppressed for anonymity).

6.6 Score-Informed Composition Identification

6.7 Section Linking

In this Section, we focus on marking the time intervals in the audio recording of a piece with the musically relevant structural elements (sections) marked in the score of the same piece (or briefly "section linking"). The proposed method extracts features from the audio recording and the sections in the score. From these features, similarity matrices are computed for each section. The method applies Hough transform (Duda & Hart, 1972) to the similarity matrices in order to detect section candidates. Then, it selects between these candidates by searching through the paths, which reflect the sequence of sections implied by the musical form, in a directed acyclic graph (DAG) directed acyclic graph. We optimize the method for the cultural-specific aspects of OTMM. By *linking* score sections with the corresponding fragments in the audio recordings, computational operations that are specific to this type of music, such as *makam* recognition (Gedik & Bozkurt, 2010), tuning analysis (Bozkurt et al., 2009) and rhythm analysis can be done at the section level, providing a deeper insight into the structural, melodic or metrical properties of the music.

6.7.1 Problem Definition and Methodology

We define *section linking* as "marking the time intervals in the audio recording at which musically relevant structural elements (sections) given in the score are performed." In this task, we start with a score and an audio recording of a music piece. The score and audio recording are known to be related with the same work (composition) via available metadata, i.e. they are already linked with each other in the document-level.

The score includes the notes, and it is divided into sections, some of which are repeated. These sections are known, and the start and end of each section are provided in the score, including the compositional repetitions. Therefore, we do not need any structural analysis to find the structural elements. From the start and end of each section, the sequence of the sections are known. The tempo and the makam of the piece are also available in the score. The audio recording follows the section sequence given in the score with possible section insertions, omissions, repetitions and substitutions. Moreover the performance might include various expressive decisions such as musical material that are not related to the piece, phrase repetitions/omissions, pitch deviations.

Following the definitions introduced in Section 4.3.1, we will apply our method to obtain the (estimated) audio section sequence $\bar{\mathbf{S}}^{(a)}$ in the au-

dio recording, where each *section link*, $\bar{\mathfrak{s}}_k^{(a)} = \left\langle \bar{\mathbf{N}}^{\left(\bar{s}_i^{(a)}\right)},\ s_k^{(a)},\ t(\bar{s}_k^{(a)}) \right\rangle$, in the sequence is paired with a section symbol in the composition $s_n \in \mathcal{S}_s$ or the "$unrelated$" section. Ideally, the *true audio section sequence*, $\bar{\mathfrak{S}}^{(a)}$, and *section link sequence*, $\bar{\mathbf{S}}^{(a)}$ should be identical. Note that the note-level alignment will be covered in Section 6.8.

Given the score representation of a composition and the audio recording of the performance of the same composition, the procedure to link the sections of a score with the corresponding sections in the audio recording is as follows:

To the best of our knowledge there only exists two methods (Şentürk, Holzapfel, & Serra, 2014; Holzapfel et al., 2015) aimed at identifying the musically relevant structures in the audio recordings of OTMM. Both methods utilize audio-score alignment between the target audio to be analyzed and the music score with annotated section information to link the sections in the score with the corresponding time-intervals in the audio. Note that both of these methods use SymbTr-txt scores, which have their sections manually annotated in advance in the reported experiments (Şentürk, Holzapfel, & Serra, 2014; Holzapfel et al., 2015).

6.7.2 Actual ExperimentsXX

Experimental Setup

Results

In (Şentürk, Holzapfel, & Serra, 2014), the partial audio-score alignment method explained in (Section 6.3) is applied using the sections in the score and candidate time-intervals in the audio recording for each of these sections are obtained. Then a directed acyclic graph is created by considering each candidate as a node and connecting each "node" with an edge if the time-distance between the start and end of two candidates are less than 3 seconds. The edge weights are computed as the transition probability from one node to another using a variable-length Markov model trained on recordings with annotated sections. The scores of the selected recordings follow the same section sequence with the score of the target recording. Then, the best path is found using a rule based scheme. Being a bottom-up approach, this method is not susceptible to global alignment errors, which is a common problem faced in methods trying to align the entire score with the audio recording. Moreover the method is can handle recordings with extra materials such as long silences, improvisations and performances of other compositions.

In (Holzapfel et al., 2015), Holzapfel et al. proposes a simpler audio-score alignment method using a hierarchical hidden Markov model (HHMM). The method aligns the entire music score with the audio recording by allowing jumps in the score in the section boundaries. In (Holzapfel et al., 2015), the HHMM based method is reported to outperform (Şentürk, Holzapfel, & Serra, 2014) under certain conditions. However, (Holzapfel et al., 2015) cannot handle recordings with extra materials, which are common in the performances of OTMM.

We modify the methodology proposed in (Şentürk, Holzapfel, & Serra, 2014) XX.

1. In (Şentürk, Holzapfel, & Serra, 2014), the sections are labeled manually according to their melody. Instead, we use the sections automatically extracted (Section 4.3.2) and labeled (Section 6.7.1) from the score.

2. In (Şentürk, Holzapfel, & Serra, 2014), only first occurrence of each section is used in the partial alignment step, ignoring other variants (e.g. the instances of a repetitive section, which have different measures in the end). We select the sections from the groups with "unique" melodies (i.e. "unique" cliques in the graph computed from the melodic dissimilarities between the synthetic pitch of each section, as explained in Section 6.7.1), so that the variants of the same melody are also included in the alignment. This allows us to

3. Most of the candidate estimations are erroneous. They are clearly separated from the correct estimations with respect to their weightsXX. Before forming the directed acyclic graph from the candidates, we use kmeans clustering citeMATLABkmeans++ to cluster the erroneous candidates and the correct candidates. We use Euclidean distance. We romve the cluster with lower weights.

4. The method requires training a VLMM for the different section progressions encountered in each form. We remove the VLMM used in the path computation and allow transition from one section to a connected section without any penalty. We only remove overlapping sections (first algorithm) and omit remove inconsistentXX and guess missing algorithms.

with some modifications.

6.8 Note-level Audio-Score Alignment

To our best knowledge the only alignment method evaluated on note-level alignment is (Şentürk, Gulati, & Serra, 2014). The method uses subsequence DTW (Müller, 2007) to refine the paths found earlier in the section linking step (Şentürk, Holzapfel, & Serra, 2014). This method is tested on a small dataset with 6 recordings consisting of 3896 note annotations. The method achieves an F_1-score of 0.66 with a tolerance of ± 200ms and 89% of the notes correctly aligned with a tolerance of ± 1sec.[6]

6.9 Audio-Lyrics alignment

Since the syllables in the lyrics are linked with the notes in the SymbTr-scores (Section 3.1.2), the onset of each syllable can be directly inferred from the relevant aligned note onset in the audio recording.

6.10 Score-Informed Predominant Melody Filtering

We can correct the remaining octave errors in the predominant melody simply by moving octave of the extracted the pitch contours to the octave of the synthetic pitch with respect to the tonic at each time instant[7].

In this step we also recompute the pitch distribution, the pitch class distribution (Section 5.5) and analyze the melodic progression (Section 5.10) using the octave-corrected predominant melody.[8]

6.11 Score-Informed Note Modeling

Since we have the aligned note onset and offsets and hence the pitch trajectory of each note, we can compute the stable pitch of each aligned note in the performance individually. Moreover we compute a histogram for each note symbol (Koduri et al., 2014) and describe how the note symbols are performed.[9] These features give us a means to specify both the overall and individual intonation and tuning characteristics of the performed notes.

[6] The compiled binary for joint structure analysis and note-level alignment is available at (link suppressed for anonymity).

[7] Our implementation is available at (link suppressed for anonymity).

[8] Our implementation is available at (link suppressed for anonymity).

[9] Our implementation is available at (link suppressed for anonymity).

6.12 Complete joint analysis

Chapter 7

Applications, Conclusions and Summary

7.1 Applications

7.1.1 Dunya

We have created a web application named Dunya-makam[1] to showcase our technologies developed for OTMM.[2] The application stores the data, executes the algorithms in our computational analysis methodology (Chapters 4-6) and display the analysis results synchronous to the audio playback (Section 7.1.1).

Dunya-makam stores the all the audio recordings and music scores in the CompMusic-makam corpora. The metadata is stored in MusicBrainz.[3] Currently, we display the analysis results obtained from joint analysisXX (Chapter 6).

To render each score element synchronously in the interface (Section 7.1.1), we first convert the score in text format to MusicXML and then to SVG (Section 4.4). We use LilyPond for MusicXML to SVG conversion, which allows us to record a mapping of each element between these different formats. This way, each object in the SVG file can be referenced by the note that it represents in the score (Section 7.1.1).

[1] http://dunya.compmusic.upf.edu/makam

[2] Main work of Alastair Porter, Andrés Ferraro and Mohamed Sordo XX. I was responsible for developing and implementing the analysis methodologies described in the Chapters 4-6 and its integration to the application.

[3] Please visit http://compmusic.upf.edu/node/280 to access the data, metadata, code of the web application and the audio-score alignment methodology, extracted features and analysis results.

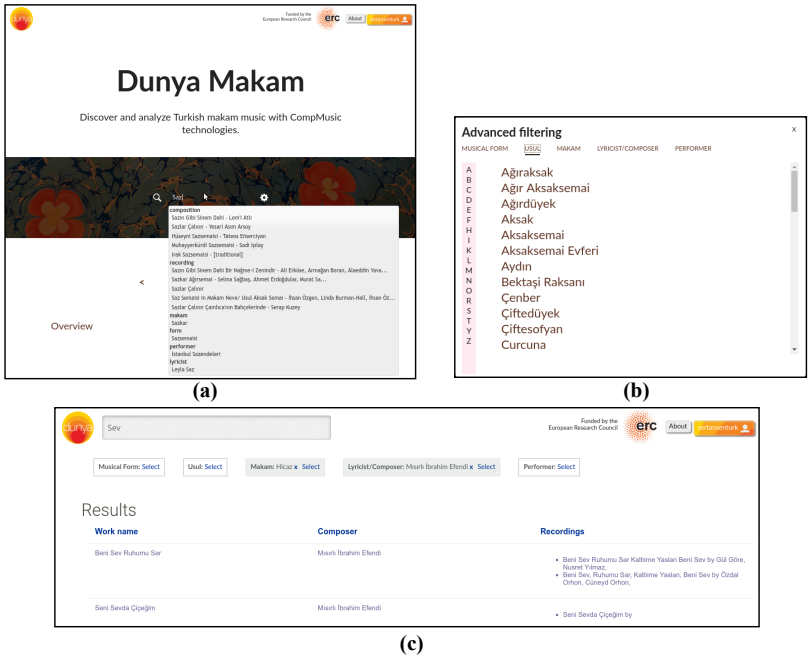


Figure 7.1: XX **a)** The main page with search results of query *Saz* showed instantly, **b)** The advanced filtering pop-up, accessed by clicking the cog-shaped button next to the search bar, and **c)** The search page, showing the composition results for the query *Sev* filtered by the Hicaz makam and *Mısırlı İbrahim Efendi* (`http://musicbrainz.org/artist/d8ab1cd7-262c-444f-8e6b-ee226022a316`) as the composer/lyricist.

Data Storage and Algorithms

Dunya (Porter, Sordo, & Serra, 2013a; Porter & Serra, 2014) is a web application that is developed with Django framework.[4] The audio recordings, music scores and relevant metadata are stored in a PostgreSQL database.[5] It's possible to manage information about the stored data and submit analysis tasks on the data from the administration panel. The output of each analysis can be used as an input of another analysis module and/or be displayed on the interface (Section 7.1.1).

Interface

The interface interaction is developed with Javascript. In the front page, thge user the user can search the works, makams, forms, usuls, composers,

[4] https://www.djangoproject.com/
[5] `https://www.postgresql.org/`

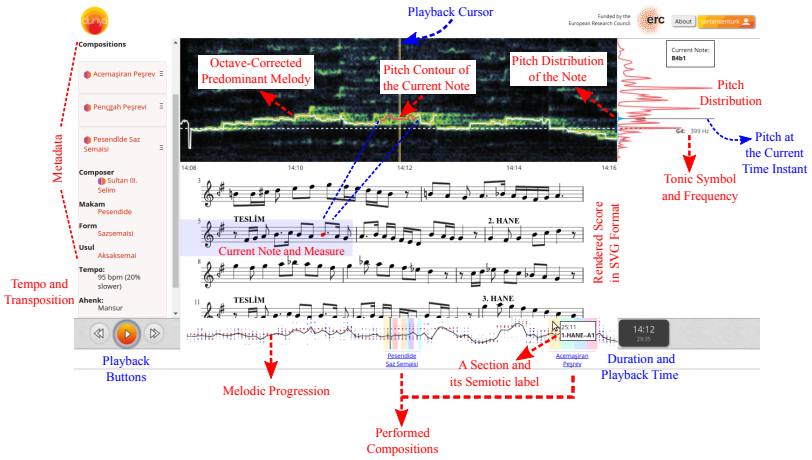


Figure 7.2: The recording page

lyricists and performers by typing in the search bar, navigate to the showcased recording pages and also access to the static *Project Overview, Statistics* and *Results* pages. The results of the query are displayed simultaneously using Ajax[6] (Figure 7.1a). The user can also start by filtering the works according to these attributes without supplying a query from by selecting the attribute value from "Advanced filtering," which can be accessed by clicking the cog-shaped button next to the search bar (Figure 7.1b). Afterwards, the user is directed to the search results (Figure 7.1c). In this page, the user can modify the query and also add, remove or modify the filters. The search is organized with respect to the works. The user can select one of the audio recordings relevant to the work. Once the user selects a recording, the recording page is opened (Figure 7.2). The page consists of four different parts:

1. **Left Panel:** The metadata about the composition and the recording. Score informed tempo (Section 6.5) and transposition (Section 5.8) for each composition is also included inside the composition tabs in this panel.

2. **Top Panel:** The audio features. These include a spectrogram, the octave-corrected predominant melody (Section 6.10). The tonic frequency (Section 6.4) is drawn with a dashed line. The chunk of the predominant melody on the time-interval of the current aligned note is highlighted in red.

[6] http://api.jquery.com/jquery.ajax/

3. **Centre Panel:** The music score. The SVG elements corresponding to the current note (if enabled) and the measure are highlighted. We obtain the SVG element related to the aligned note from the mappings mentioned in Section 4.4. We also find the current measure by searching the two closest measure line elements, which encloses the position of the note element in the SVG file.

4. **Bottom Panel:** The playback buttons, audio timeline, playback time instance and audio duration. The melodic progression (Section 5.10) extracted from the octave-corrected predominant melody is drawn in the background of the timeline. The timeline can be clicked to jump the playback to the desired time instance. If there are multiple compositions performed in the audio recording, the time intervals are indicated in the bottom of the timeline. The coloured regions mark the time-intervals of the performed sections. When the user hovers over the panel, the semiotic label of the section (Section 4.3.2) at this time is displayed.

Summary

Display only audio analysis and score analysis
Audio-lyrics alignment is displayed separately XX
Other future extensions? XX

Appendix A

List of Publications by the author

Appendix B

Preliminary Section Linking Experiments

To properly apply morphological operations, the similarity matrix is first subtracted from 1 such that the "valleys" become "hills." Then, the image is dilated. The structural element is picked as a binary diagonal beam lying in the 2^{nd} and 4^{th} quadrants with the focus at the origin (Table B.1). Next, the similarity matrix is eroded twice. The structural element is a similar beam to the beam defined for dilation, but smaller (Table B.2). Later, to remove noises, the similarity matrix is opened with the same structuring element used in the erosions. Next, the similarity matrices are converted into binary images by applying thresholding, such that all values higher than 0.96 are given the value one and all other values are assigned to zero. Structural component analysis is done on the binary image to find the blobs. All blobs that are not in the desired diagonal orientation (i.e. lying between 0 and -90 degrees) are removed. From the remaining blobs only the biggest 20% are picked. As a last step in pre-processing the similarity matrix, the image is dilated by a 3x3 square structuring element to slightly widen the diagonals.

B.1 Results

Table B.1: Structural element defined for the dilation operation

1	1	1	.	.	.	.	.	.
1	1	1	.	.	.	.	.	.
1	1	1	1	.	.	.	.	.
.	.	1	1	.	.	.	.	.
.	.	.	.	1	.	.	.	.
.	.	.	.	.	1	1	.	.
.	.	.	.	.	1	1	1	1
.	.	.	.	.	.	1	1	1
.	.	.	.	.	.	1	1	1

Table B.2: Structural element defined for the erosion and opening operations

1	1	.	.	.
1	1	.	.	.
.	.	1	.	.
.	.	.	1	1
.	.	.	1	1

Appendix C

Resources

Additional Material «Whatever could not make it to the main text»

Resources List all specific contributions: code, papers, datasets, examples. Discuss specific datasets and code, with links, explain the dataset organization, access.

one idea: identify recorded cycles which are very close to the depicted patterns, and provide them as acoustic documentation on a website and multimedia appendix to the thesis

Appendix D

Towards Open and Reproducible Research XX

In the start of the CompMusic project, there had been a lack of open and comprehensive datasets and also open-source software, which is related to the studied music cultures. Moreover, the existing tools were not addressing a few of the research problems (such as TARSOSXX), were hard to deploy (such as Makam Toolbox) and impractical to deploy (e.g. to Dunya-makamXX). Same with data and experiments (both the scripts and results).

While I had not identified the lack of public tools, datasets and experiments initially as a problem itself, I experienced that this absence as a major obstacle for reproducing, building on top of and hence advancing Music Information Research (MIR) afterwards.XX As a result, I have dedicated a fair amount of effort in the last year of my Ph.D. to reimplement the existing state-of-the-art applied on OTMM (e.g. most of the methodologies described in Chapter 5) and package the code of the novel methodologies proposed in this thesis (e.g. the audio-score alignment methodologies described in Section 6.3-6.8) with modularity, readability and extensibility in mind. In the meantime I have also been taking a

This Chapter gives an overview of implementation of the tools released, data, experimentsXX within the dissertation. I dont claim the thesis is reproducible, in fact many of the earlier research such as (Şentürk et al., 2012) is the contrary. The aim of this Chapter is document and also provide a "better" start for future researchers. XX

D.1 Software

The analysis tasks described throughout the thesis are packaged into separate repositories for the sake of modularity. They are publicly hosted in *GitHub*[1] with complete version history. To promote open science and reproduciblity, the repositories are published freely under *Affero GPLv3 License*.[2] Within the development cycle, the code is periodically released according to the convensions defined by the *Semantic Versioning* v2.0.0.[3] Moreover, each release is automatically assigned a Digital Object Identifier (DOI) using *Zenodo*'s[4] Github integration to be able to discover and cite the code with proper versioning.[5] In order to detect erroneous behaviour (e.g. bugs) immediately and ensure reliable operation, the majority of the repositories consist of unit tests. These unit tests, along with code style validation,[6] code coverage[7] and code quality[8] checkers, are invoked automatically using *Travis CI*,[9] when a "commit" is pushed to GitHub. This automated setup, helps to identify and solve software bugs earlier; reduces the time, effort and risks introduced by code repetition; improves readability and enhances sustainable development, in general.

Except the audio-score alignment code, all tools are implemented in *Python* 2.7. They generally follow the conventions of object-oriented programming. For coding style consistency, *PEP* 8 style guide[10] is strictly followed. The code in each package is organized into modules.[11] Moreover the packages include the setup scripts,[12] which would allow the user to install these packages to different machines with ease. The developed code depends on other open source software such as *NumPy* (Walt, Colbert, & Varoquaux, 2011) and *SciPy* (Jones, Oliphant, Peterson, et al., 2001–) for numeric computations, *scikit-learn* (Pedregosa et al., 2011) for machine learning related tasks, *matplotlib* (Hunter, 2007) for visualizations, *pandas* (McKinney, 2010) for processing tabular data, *NetworkX* (Hagberg, Schult, & Swart, 2008) for graph analysis, *eyeD3* [13] for reading embedded

[1] https://github.com/
[2] https://www.gnu.org/licenses/agpl-3.0.en.html
[3] http://semver.org/spec/v2.0.0.html
[4] https://zenodo.org/
[5] https://guides.github.com/activities/citable-code/
[6] using *flake8* (http://flake8.pycqa.org/en/latest/) for the code in *Python*
[7] using *codecov* (https://codecov.io/) for the code in *Python*
[8] using *QuantifiedCode* http://docs.quantifiedcode.com/ for the code in *Python*
[9] https://travis-ci.org/
[10] https://www.python.org/dev/peps/pep-0008/
[11] https://docs.python.org/2/tutorial/modules.html
[12] https://docs.python.org/2/distutils/setupscript.html
[13] http://eyed3.nicfit.net/

metadata in audio files, *MusicBrainz NGS bindings*[14] for crawling MusicBrainz, *LilyPond* (Nienhuys & Nieuwenhuizen, 2003) for score engraving and Essentia (Bogdanov et al., 2013) for audio processing. *Jupyter notebooks*[15] are provided as "user manuals" of each package to demonstrate example usage. Since one motivations of the thesis is handling large digital audio collections, we also provide examples of parallelization through *ipyparallel*,[16] which is a part of the *IPython project* (Pérez & Granger, 2007).

There exists are two repositories, which are not written in *Python*. As explained in Section 4.5, the symbolic phrase segmentation package is written in *MATLAB* scripting language.[17] The repo is a fork of Bozkurt, Karaosmanoğlu, et al. (2014)'s original source code and it introduces performance optimizations and wrapper functions for feature extraction, training, testing etc. The second is the audio-score alignment code, which is mainly written in *MATLAB* scripting[18] too, except the dynamic time warping (DTW) implementation in *C*. The DTW implementation is mainly written by Sankap Gulati with modifications for OTMM (such as the implementation of Equation 5.3) introduced by myself. The implementation is compiled as a *mex* function[19] to interface with the *MATLAB* scripts. Currently, the phrase segmentation[20] and audio-score alignment[21] code is compiled into binaries for Linux and MacOSX using *MATLAB compiler*[22] in *MATLAB R2015a (8.5)*, hence the algorithms can be called using *MATLAB Runtime* [23] without the need of a *MATLAB* proprietary license. In the future, we would like to port the audio-score alignment code to *Cython*[24] (i.e. the DTW implementation will stay in *C*). We would also like to update all the code in *Python* 2.7 to latest version of *Python 3* for sustainability in the future.[25]

To automate the automatic description of the music scores, audio recordings both indivudually and jointly, I have implemented toolbox

[14] https://github.com/alastair/python-musicbrainzngs
[15] http://jupyter.org/
[16] https://github.com/ipython/ipyparallel/releases
[17] http://es.mathworks.com/products/matlab/
[18] http://es.mathworks.com/products/matlab/
[19] http://es.mathworks.com/help/matlab/ref/mex.html
[20] Hosted in its package releases: https://github.com/MTG/makam-symbolic-phrase-segmentation/releases
[21] Hosted in https://github.com/sertansenturk/tomato_binaries
[22] http://www.mathworks.com/products/compiler/
[23] http://www.mathworks.com/products/compiler/mcr/
[24] Cython
[25] We will start working on *Python 3*+ support, as soon as the Essentia bindings are available: https://github.com/MTG/essentia/issues/138.

called **T**urkish-**O**ttoman **M**akam (M)usic **A**nalysis **TO**olbox (`tomato`). `tomato` is a comprehensive and easy-to-use toolbox for the analysis of audio recordings and music scores of OTMM. The aim of the toolbox is to easily analyze large-scale audio recording and music score collections of Turkish-Ottoman makam music, using the state-of-the-art methodologies explained throughout the thesis, which are designed specifically for the culture-specific characteristics of OTMM. `tomato` also implements the sequence of the steps in the complete score analysis (Section 4.5), complete audio analysis (Section 5.11) and complete joint analysis (Section 6.12). The toolbox is designed such that complete analysis methods is able to output partial results in case some steps fail during the analysis. As described in Section 7.1, `tomato` is already integrated to Dunya-makam, the prototype web application of CompMusic for the discovery of OTMM.

The implementations of all the methods described in Chapters 4-6 are summarized in the Tables D.1-D.3.

In the future, we would like to provide Future, add proper API and documentation. Add docker images of tomato to easily deploying to web services and also for the sake of reproducibility.XX

Table D.1: An overview of the implementations of the score analysis methodologies and score format converters explained in Chapter 4.

Task	Methodology	Inputs	Package	Version	Comments
Metadata extraction	Section 4.1	Work MBID	`https://github.com/sertansenturk/makammusicbrainz`	v1.3.0	Requires internet connection to access MusicBrainz
Phrase segmentation	(Bozkurt, Karaosmanoğlu, et al., 2014), Section 4.3.2	**SymbTr**-txt score	`https://github.com/MTG/makam-symbolic-phrase-segmentation`	v1.0-alpha.1	Forked from `http://akademik.bahcesehir.edu.tr/~bbozkurt/112E162.html`. The binary is hosted in the releases of the package, called *phraseSeg*
Section extraction	(Şentürk & Serra, 2016b), Section 4.3.2	**SymbTr**-txt score	`https://github.com/sertansenturk/symbtrdataextractor`	v2.1.0	Implemented in *symbtrdataextractor/section.py*
Semiotic labeling	(Şentürk & Serra, 2016b), Section 4.3.2	**SymbTr**-txt score	`https://github.com/sertansenturk/symbtrdataextractor`	v2.1.0	Implemented in *symbtrdataextractor/structurelabeler.py*
Synthetic melody extraction	(Şentürk, Holzapfel, & Serra, 2014), Section 4.2.2	**SymbTr**-txt score	`https://github.com/sertansenturk/symbtrdataextractor`	v2.1.0	Implemented in *symbtrdataextractor/scoreprocessor.py*. *fragmentLinker* package (Table D.3) uses an internal implementation.)
Complete score analysis	Sections 4.1- 4.3	**SymbTr**-txt score, **SymbTr**-mu2 score (optional), work MBID (optional)	`https://github.com/sertansenturk/symbtrdataextractor`	v2.1.0	Calls all the tasks above and also cross-validates the metadata obtained from MusicBrainz, **SymbTr**-txt and **SymbTr**-mu2 scores
SymbTr-txt to MusicXML conversion	Section 4.4	**SymbTr**-txt score, **SymbTr**-mu2 score (optional), work MBID (optional)	`https://github.com/burakuyar/MusicXMLConverter`	v1.2.1	**SymbTr**-mu2 and work glsMBID can be input to enhance the work metadata embedded to the MusicXML score
SymbTr-MusicXML to LilyPond conversion	Section 4.4	**SymbTr**-MusicXML score	`https://github.com/hsercanatli/makam-musicxml2lilypond`	v1.2.1	
LilyPond to SVG conversion	Section 4.4	**SymbTr**-MusicXML score	`https://github.com/sertansenturk/tomato`	v0.9.1	Implemented in *tomato/symbolic/scoreconverter.py*
Complete score analysis and format conversion	Chapter 4	**SymbTr**-txt score, **SymbTr**-mu2 score (optional), work MBID (optional)	`https://github.com/sertansenturk/tomato`	v0.9.1	Implemented in *tomato/symbolic*. Depends on all packages above.
Additional **SymbTr** tools	Section 4.5.2	**SymbTr**-txt or **SymbTr**-mu2 score	`https://github.com/MTG/SymbTr-extras`	v0.3.dev	Tools to manipulate the music scores to maintain consistency in formatting (e.g. encoding, line breaks), content (e.g. rest names, usul annotations) etc. Depends on *makammusicbrainz*, *symbtrdataextractor* and *MusicXMLConverter*.

Table D.2: An overview of the implementations of the audio analysis methodologies explained in Chapter 5.

Task	Methodology	Inputs	Package	Version	Comments
Metadata extraction	Section 5.1	Audio file or recording MBID	https://github.com/sertansenturk/makammusicbrainz	v1.3.0	Requires internet connection to access MusicBrainz
Predominant melody extraction	(Atlı et al., 2014), Section 5.2.1	Audio file	https://github.com/sertansenturk/predominantmelodymakam	v1.2.0	See ATL-MEL
Predominant melody filtering	(Bozkurt, 2008), Section 5.2.1	Predominant Melody	https://github.com/hsercanatli/pitchfilter	v1.2.1	See ATL-MEL$_f$
Pitch distribution extraction	(Bozkurt, 2008), Section 5.5	Filtered predominant melody (using ATL-MEL$_f$)	https://github.com/altugkarakurt/morty	v1.2.1	Implemented in *morty/pitchdistribution.py*
Pitch-class distribution extraction	(Chordia & Şentürk, 2013), Section 5.5	Filtered predominant melody (using ATL-MEL$_f$)	https://github.com/altugkarakurt/morty	v1.2.1	Implemented in *morty/pitchdistribution.py*
Stable pitch extraction	(Smith III & Serra, 1987), Section 5.6	Pitch distribution	https://github.com/altugkarakurt/morty	v1.2.1	Computed by *detect_peaks* function in *morty/pitchdistribution.py*.
Stable pitch-class extraction	(Smith III & Serra, 1987), Section 5.6	Pitch-class distribution	https://github.com/altugkarakurt/morty	v1.2.1	Computed by *detect_peaks* function in *morty/pitchdistribution.py*.
Tonic identification I	(Atlı et al., 2015), Section 5.7.2	Filtered predominant melody (using ATL-MEL$_f$)	https://github.com/hsercanatli/tonicidentifier_makam	v1.3.1	See ATL-TON. Uses the PD and PCD implementations in MORTY
Tonic identification II	(Karakurt et al., 2016), Section 5.7.2	Pitch-class distribution, makam (optional)	https://github.com/altugkarakurt/morty	v1.2.1	If makam input is not given, the task correcponds to joint estimation of makam and tonic.
Transposition identification	Section 5.8	Tonic, makam or tonic Symbol	https://github.com/sertansenturk/ahenkidentifier	v1.5.0	
Makam recognition	(Karakurt et al., 2016), Section 5.7.2	Pitch-class distribution,tonic (optional)	https://github.com/altugkarakurt/morty	v1.2.1	If tonic input is not given, the task correcponds to joint estimation of makam and tonic.
Tuning analysis	(Bozkurt et al., 2009), Section 5.9	Stable pitches, makam	https://github.com/miracatici/notemodel	v1.2.1	Depends on MORTY for PD implementation and peak detection. The implementation accepts a PD, instead of the stable pitches for the sake of usability and computes the stable pitches internally. The method fails, if the scale of the makam is not available.
Melodic progression analysis	(Bozkurt, 2015), Section 5.10	Filtered predominant melody (using ATL-MEL$_f$)	https://github.com/sertansenturk/seyiranalyzer	v1.1.1	Depends on MORTY for PD computation
Complete audio analysis	Section 5.11	Audio file	https://github.com/sertansenturk/tomato	v0.9.1	Implemented in *tomato/audio*. Depends on all packages above.

Table D.3: An overview of the implementations of the joint analysis methodologies explained in Chapter 6.

Task	Methodology	Inputs	Package	Version	Comments
Fragment linking	(Şentürk, Holzapfel, & Serra, 2014), Section 6.3	Predominant melody of the audio recording (using ATL-MEL) using ATL-MEL, **SymbTr**-score fragment	https://github.com/sertansenturk/fragmentLinker	v0.1.0	Fundamental step in all audio-score alignment tasks/implementations. Implemented in *+fragmentLinker/@CandidateLinkEstimator/*. Synthetic melody is computed internally from the score.
Tonic identification	(Şentürk et al., 2013), Section 6.4	Predominant melody of the audio recording (using ATL-MEL), synthetic melody of the music score fragment	https://github.com/sertansenturk/fragmentLinker	v0.1.0	Implemented in *+makamLinker/TonicIdentifier*. Obtained jointly with the tempo. The binary is hosted in *tomato_binaries*, called *extractTonicTempoTuning*
Tempo estimation	(Holzapfel et al., 2015), Section 6.5	Predominant melody of the audio recording (using ATL-MEL), synthetic melody of the music score fragment	https://github.com/sertansenturk/fragmentLinker	v0.1.0	Implemented in *+makamLinker/TempoEstimator*. Obtained jointly with the tonic. The binary is hosted in *tomato_binaries*, called *extractTonicTempoTuning*
Composition identification	(Şentürk & Serra, 2016a), Section 6.6	Predominant melody of the audio recording (using ATL-MEL), synthetic melody of the music score fragment	https://github.com/sertansenturk/fragmentLinker	v0.1.0	Implemented in *+makamLinker/CompositionIdentifier*. Obtained jointly with the tonic.
Section linking	(Şentürk, Holzapfel, & Serra, 2014), Section 6.7	Predominant melody and tonic of the audio recording, synthetic melody and semiotic section labels of the music score fragment, tempo of the audio recording (optional)	https://github.com/sertansenturk/fragmentLinker	v0.1.0	Implemented in *+makamLinker/SectionLinker*. Obtained together with the aligned notes. The binary is hosted in *tomato_binaries*, called *alignAudioScore*. Fails if the section annotations are missing in the score.
Note-level alignment	(Şentürk, Gulati, & Serra, 2014), Section 6.8	Inputs of section linking task, section links	https://github.com/sertansenturk/fragmentLinker	v0.1.0	Implemented in *+makamLinker/NoteAligner*. Obtained together with the section links. The binary is hosted in *tomato_binaries*, called *alignAudioScore*
Predominant Melody Filtering	Section 6.10	Predominant melody of the audio recording (using ATL-MEL), aligned notes	https://github.com/sertansenturk/alignedpitchfilter	v1.1.0	
Note model computation	(Şentürk et al., 2016), Section 6.11	Predominant melody of the audio recording filtered with respect to the note alignments, aligned notes, Makam or Tonic Symbol	https://github.com/sertansenturk/alignednotemodel	v1.1.1	Depends on MORTY for PD implementation and peak detection.
Complete joint analysis	Section 6.12	Audio file, predominant melody of the audio recording (using ATL-MEL), **SymbTr** score, score features computed by *symbtrdataextractor*	https://github.com/sertansenturk/tomato	v1.1.1	Implemented in *tomato/joint*. Depends on all packages above.

Appendix E

Applications in Other Music Cultures

Appendix F

Glossary

F.1 Technical Terms and Datasets

clique Clique XX
CompMusic CompMusicXX
directed acyclic graph Directed acyclic graph XX
Django Django XX
Dunya DunyaXX
Dunya-makam Dunya-makamXX
edge Edge XX
Essentia an open-source library for audio analysis and audio-based music information retrieval (Bogdanov et al., 2013)
graph Graph XX
the Hough transform XX
LilyPond score engraver and the format with the same name XX
link XX
Makam Toolbox a makam audio analysis framework developed by Gedik and Bozkurt (2010)
MATLAB MATLABXX
maximal clique XX
MELODIA Essentia implementation of the predominant melody extraction method proposed by Salamon and Gómez (2012)
MusicBrainz an open music encyclopedia of metadata
MusicXML XX
node Node XX
PostgreSQL PostgreSQL XX
similar clique XX
Sonic Visualizer an audio analysis and visualization application developed at the Centre for Digital Music, Queen Mary, University of

London
SoundFont soundfont XX
subgraph Subgraph XX
TiMidity++ TiMidity++ XX
unique clique XX

F.2 Generic Music ConceptsXX

mode the melodic framework XX
tonic TonicXX

F.3 Ottoman-Turkish Makam Music

ahenk TranspositionXX
bolahenk default ahenk of OTMM. G4≈XX
curcuna an usul with 10 beats in a cycle
çargah the note indicated by the C5 note in the staff notation
form XX
Hicaz a makam XX
Hüseyni a makam XX
hüseyni the note indicated by the E5 note in the staff notation
ilahi the most common religious form of classical Ottoman-Turkish makam music repertoire.
kapalı curcuna a variant of the curcuna usul with "closed" strokes
karar TonicXX
makam the melodic framework of Ottoman-Turkish art and folk music
Nihavent a makam XX
peşrev XX
şarkı the most common vocal form of classical Ottoman-Turkish makam music repertoire. Its literal translation is "song."
sazsemaisi XX
seyir melodic progression XX
taksim an unmetered melodic improvisation of a makam
terennüm an unmetered melodic improvisation of a makam
teslim Repetitive section in Peşrev and Sazsemaisi forms
türkü the most common vocal form of folk Ottoman-Turkish makam music repertoire.
usul the rhythmic framework of Ottoman-Turkish art and folk music

F.4 Acronyms

SymbTr the collection of machine-readable Ottoman-Turkish makam music scores by K. Karaosmanoğlu (2012)
TMKH Türk Müzik Kültürünün Hafızası (English: "Memory of Turkish Music Culture" Collection)
TRT-TTMA TRT Tarihi Türk Müziği Arşivi) (English: TRT Historical Turkish Music Archive
TSMD Türk Sanat Müziği Derlemi (English: Turkish Art Music Corpus)
UHHD Uzun Hava Humdrum Database
k`NN` k nearest neighbors
`ATL-MEL` the predominant melody extraction procedure described in (?, ?)
`ATL-MEL`$_f$ the variant of the predominant melody extraction procedure described in (?, ?) using the post filtering method described in (Bozkurt, 2008)
`ATL-TON` the tonic identification method described in (Atlı et al., 2015)
`DTW` dynamic time warping
`LCM` least common multiplier
`MORTY` **MO**de **R**ecognition and **T**onic **Y**dentification Toolbox (Karakurt et al., 2016)
`OMR` optical music recognition
`SDTW` subsequence dynamic time warping
`SEN-YIN`$_f$ the predominant melody extraction procedure described in (Şentürk et al., 2012)
`SEN-MEL` the predominant melody extraction procedure described in (Şentürk, Holzapfel, & Serra, 2014)
`VMD-SIM` the predominant melody extraction method proposed by (Şimşek et al., 2016)
`VMD` variational mode decomposition
`YIN` the fundamental pitch extraction method proposed by (De Cheveigné & Kawahara, 2002)
`tomato` **T**urkish-**O**ttoman **M**akam (M)usic **A**nalysis **TO**olbox
AEU theory the music theory XX
DAG directed acyclic graph
DOI Digital Object Identifier
Hc Holderian comma
HPCP harmonic pitch class profile (Gómez, 2006)
ICT Information and Communication Technologies
MBID MusicBrainz Identifier
MIDI Musical Instrument Digital Interface
MIR Music Information Research

MIREX Music Information Retrieval EXchange
OTMM Ottoman-Turkish makam music
PCD pitch-class distribution
PD pitch distribution
PDF Portable Document Format
SVG Scalable Vector Graphics Format
TRT Türkiye Radyo ve Televizyon Kurumu (English: Turkish Radio and Television Corporation)
TSV tab separated values

Bibliography

Abdoli, S. (2011, October 24-28). Iranian traditional music dastgah classification. In *Proceedings of the 12th international society for music information retrieval conference* (pp. 275–280). Miami (Florida), USA. [58]

Arel, H. S. (1968). *Türk musikisi nazariyatı (the theory of turkish music)*. ITMKD yayınları. [5]

Arzt, A., Böck, S., & Widmer, G. (2012). Fast identification of piece and score position via symbolic fingerprinting. In *Proceedings of 13th international society for music information retrieval conference (ismir)* (pp. 433–438). [79]

Atalay, N. B., & Yöre, S. (2011). *Türk sanat müziği derlemi.* www.tsmderlemi.com. [13]

Atlı, H. S. (2016). *Türk makam müziği'nin ezgisel boyutuna yönelik İnteraktif eğitim programı* (Unpublished master's thesis). Bahçeşehir Üniversitesi. [xxx, 22, 51, 52]

Atlı, H. S., Bozkurt, B., & Şentürk, S. (2015). A method for tonic frequency identification of Turkish makam music recordings. In *5th international workshop on folk music analysis (fma)* (pp. 119–122). Paris, France: University Pierre et Marie Curie. Retrieved from `http://sertansenturk.com/uploads/publications/atli2015tonic_fma.pdf` [xxxi, 54, 58, 59, 67, 68, 69, 70, 72, 110, 117]

Atlı, H. S., Uyar, B., Şentürk, S., Bozkurt, B., & Serra, X. (2014). Audio feature extraction for exploring Turkish makam music. In *3rd international conference on audio technologies for music and media.* Ankara, Turkey: Bilkent University. Retrieved from `http://sertansenturk.com/uploads/publications/atli2014feature_atmm.pdf` [22, 50, 51, 66, 69, 73, 110]

Aucouturier, J.-J., & Sandler, M. (2002, 6). Finding repeating patterns in acoustic musical signals: Applications for audio thumbnailing. In *Proceedings of 22nd international audio engineering society conference: Virtual, synthetic, and entertainment audio.* Retrieved from http://www.aes.org/e-lib/browse.cfm?elib=11163 [85]

Ballard, D. H. (1981). Generalizing the hough transform to detect arbitrary shapes. *Pattern recognition*, *13*(2), 111–122. [85]

Benetos, E., & Holzapfel, A. (2015). Automatic transcription of Turkish microtonal music. *Journal of the Acoustical Society of America*, *138*(4), 2118-2130. [57, 58]

Bimbot, F., Deruty, E., Sargent, G., & Vincent, E. (2012, October 8-12). Semiotic structure labeling of music pieces: Concepts, methods and annotation conventions. In *Proceedings of the 13th international society for music information retrieval conference (ismir)* (pp. 235–240). Porto, Portugal. [35]

Bod, R. (2002). Memory-based models of melodic analysis: Challenging the gestalt principles. *Journal of New Music Research*, *31*(1), 27–36. [29]

Bogdanov, D., Wack, N., Gómez, E., Gulati, S., Herrera, P., Mayor, O., Roma, G., Salamon, J., Zapata, J., & Serra, X. (2013). Essentia: An audio analysis library for music information retrieval. In *Proceedings of 14th international society for music information retrieval conference (ismir)* (pp. 493–498). [27, 49, 107, 115]

Bozkurt, B. (2008). An automatic pitch analysis method for Turkish maqam music. *Journal of New Music Research*, *37*(1), 1–13. [22, 47, 48, 51, 54, 56, 58, 60, 61, 62, 68, 70, 72, 73, 110, 117]

Bozkurt, B. (2012). A system for tuning instruments using recorded music instead of theory-based frequency presets. *Computer Music Journal*, *36*, 43–56. [57]

Bozkurt, B. (2015). Computational analysis of overall melodic progression for Turkish makam music. In M. Ayari (Ed.), *Penser l' improvisation* (p. 266-290). Sampzon, France: Delatour France. Retrieved from http://compmusic.upf.edu/system/files/static_files/Bozkurt_SeyirAnalysis_BookChapter_2015.pdf [76, 110]

Bozkurt, B., Ayangil, R., & Holzapfel, A. (2014). Computational analysis of Turkish makam music: Review of state-of-the-art and challenges. *Journal of New Music Research*, *43*(1), 3-23. [20, 47]

Bozkurt, B., Karaosmanoğlu, M. K., Karaçalı, B., & Ünal, E. (2014). Usul and makam driven automatic melodic segmentation for Turkish

music. *Journal of New Music Research*, *43*(4), 375-389. [21, 22, 29, 32, 38, 41, 42, 107, 109]

Bozkurt, B., Yarman, O., Karaosmanoğlu, M. K., & Akkoç, C. (2009, March). Weighing diverse theoretical models on Turkish maqam music against pitch measurements: A comparison of peaks automatically derived from frequency histograms with proposed scale tones. *Journal of New Music Research*, *38*(1), 45–70. [54, 58, 74, 75, 89, 110]

Cambouropoulos, E. (2001). The local boundary detection model (lbdm) and its application in the study of expressive timing. In *Proceedings of the international computer music conference* (pp. 17–22). [29]

Casey, M. A., Veltkamp, R., Goto, M., Leman, M., Rhodes, C., & Slaney, M. (2008). Content-based music information retrieval: Current directions and future challenges. *Proceedings of the IEEE*, *96*(4), 668-696. [3]

Cha, S.-H., & Srihari, S. N. (2002). On measuring the distance between histograms. *Pattern Recognition*, *35*(6), 1355–1370. [62]

Chordia, P., & Şentürk, S. (2013). Joint recognition of raag and tonic in North Indian music. *Computer Music Journal*, *37*(3). [xxvi, 47, 55, 58, 59, 60, 61, 62, 63, 67, 71, 73, 110]

Chordia, P., & Rae, A. (2007). Raag recognition using pitch-class and pitch-class dyad distributions. In *Proceedings of 8th international conference on music information retrieval (ismir)* (pp. 431–436). Vienna, Austria. [58]

Cont, A. (2010). A coupled duration-focused architecture for real-time music-to-score alignment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *32*(6), 974–987. [79]

De Cheveigné, A., & Kawahara, H. (2002). YIN, a fundamental frequency estimator for speech and music. *Journal of Acoustical Society of America*, *111*(4), 1917–1930. [47, 117]

Devaney, J., Mandel, M., & Fujinaga, I. (2012). A study of intonation in three-part singing using the automatic music performance analysis and comparison toolkit (AMPACT). In *Proceedings of 13th international society for music information retrieval conference (ismir)* (pp. 511–516). [79, 80]

Dighe, P., Karnick, H., & Raj, B. (2013, November 4-8). Swara histogram based structural analysis and identification of indian classical ragas. In *Proceedings of the 14th international society for music information retrieval conference.* [58]

Duda, R. O., & Hart, P. E. (1972). Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM*, *15*(1), 11–15. [85, 89]

Ederer, E. B. (2011). *The theory and praxis of makam in classical Turkish music 1910-2010* (Unpublished doctoral dissertation). University of California, Santa Barbara. [5]

Ellis, D. P., & Poliner, G. E. (2007). Identifying 'cover songs' with chroma features and dynamic programming beat tracking. In *Proceedings of ieee international conference on acoustics, speech and signal processing (icassp)* (Vol. 4, pp. 1429–1432). [3]

Ewert, S., & Müller, M. (2012). Score-informed source separation for music signals. In M. Müller, M. Goto, & M. Schedl (Eds.), *Multimodal music processing* (Vol. 3, pp. 73–94). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik. [79]

Fujihara, H., & Goto, M. (2012). Lyrics-to-audio alignment and its application. In M. Müller, M. Goto, & M. Schedl (Eds.), *Multimodal music processing* (Vol. 3, pp. 23–36). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik. [3]

Gedik, A. C., & Bozkurt, B. (2010). Pitch-frequency histogram-based music information retrieval for Turkish music. *Signal Processing*, *90*(4), 1049–1063. [xxvi, 54, 55, 58, 59, 60, 61, 62, 63, 64, 67, 68, 69, 71, 73, 89, 115]

Gómez, E. (2006). *Tonal description of music audio signals* (Unpublished doctoral dissertation). Universitat Pompeu Fabra. [xxiii, 27, 47, 52, 54, 117, 131]

Gulati, S. (2011). *A Tonic Identification Approach for Indian Art Music* (Master's Thesis). Universitat Pompeu Fabra, Barcelona, Spain. [58]

Gulati, S., Serrà, J., Ganguli, K. K., Şentürk, S., & Serra, X. (2016). Time-delayed melody surfaces for rāga recognition. In *Proceedings of 17th international society for music information retrieval conference (ismir 2016)* (pp. 751–757). New York, NY, USA. Retrieved from http://sertansenturk.com/uploads/publications/gulati2016ragarecognition_ismir.pdf [58, 66, 73, 78]

Gulati, S., Serrà, J., Ishwar, V., Şentürk, S., & Serra, X. (2016). Phrase-based rāga recognition using vector space modeling. In *Proceedings of 41st ieee international conference on acoustics, speech and signal processing (icassp 2016)* (p. 66-70). Shanghai, China: IEEE. Retrieved from http://sertansenturk.com/uploads/publications/gulati2016phraseRaga_icassp.pdf [57, 58, 66, 72]

Hagberg, A. A., Schult, D. A., & Swart, P. J. (2008, August). Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in science conference (SciPy2008)* (pp. 11–15). Pasadena, CA USA. [106]

Holzapfel, A. (2010). *Similarity methods for computational ethnomusicology* (Unpublished doctoral dissertation). University of Crete. [3]

Holzapfel, A., Şimşekli, U., Şentürk, S., & Cemgil, A. T. (2015). Section-level modeling of musical audio for linking performances to scores in Turkish makam music. In *Ieee international conference on acoustics, speech and signal processing (icassp)* (pp. 141–145). Brisbane, Australia: IEEE. Retrieved from http://sertansenturk.com/uploads/publications/holzapfel2015linking_icassp.pdf [88, 90, 91, 111]

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, *9*(3), 90-95. Retrieved from http://scitation.aip.org/content/aip/journal/cise/9/3/10.1109/MCSE.2007.55 [106]

Jackendoff, R. (1985). *A generative theory of tonal music*. MIT Press. [29]

Jones, E., Oliphant, T., Peterson, P., et al. (2001–). *SciPy: Open source scientific tools for Python*. Retrieved from http://www.scipy.org/ ([Online; accessed 2016-08-24]) [106]

Karakurt, A., Şentürk, S., & Serra, X. (2016). MORTY: A toolbox for mode recognition and tonic identification. In *Proceedings of the 3rd international digital libraries for musicology workshop (dlfm 2016)* (pp. 9–16). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/2970044.2970054 [21, 59, 60, 65, 110, 117]

Karaosmanoğlu, K. (2012). A Turkish makam music symbolic database for music information retrieval: SymbTr. In *Proceedings of 13th international society for music information retrieval conference (ismir)* (pp. 223–228). Porto, Portugal. [xxxiii, 11, 12, 27, 117]

Karaosmanoğlu, M. K., Bozkurt, B., Holzapfel, A., & Doğrusöz Dişiaçık, N. (2014). A symbolic dataset of Turkish makam music phrases. In *Proceedings of 4th international workshop on folk music analysis* (pp. 10–14). Istanbul, Turkey. [32, 33]

Koduri, G. K., Gulati, S., Rao, P., & Serra, X. (2012). Rāga recognition based on pitch distribution methods. *Journal of New Music Research*, *41*(4), 337–350. [57]

Koduri, G. K., Ishwar, V., Serrà, J., & Serra, X. (2014, January). Intonation analysis of rāgas in Carnatic music. *Journal of New Music Research*, *43*(01), 72–93. Retrieved from http://www.tandfonline.com/doi/abs/10.1080/09298215.2013.866145 [47, 58, 92]

Krumhansl, C. L., & Shepard, R. N. (1979). Quantification of the hier-

archy of tonal functions within a diatonic context. *Journal of experimental psychology: Human Perception and Performance*, *5*(4), 579–594. [54]

Lartillot, O., & Ayari, M. (2009). Segmentation of Tunisian modal improvisation: Comparing listeners' responses with computational predictions. *Journal of New Music Research*, *38*(2), 117–127. [29]

Lartillot, O., Yazıcı, Z. F., & Mungan, E. (2013). A pattern-expectation, non-flattening accentuation model, empirically compared with segmentation models on traditional Turkish music. In *Proceedings of the 3rd international workshop on folk music analysis* (pp. 63–70). Amsterdam, Netherlands. [29]

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady* (Vol. 10, pp. 707–710). [33]

Martin, B., Robine, M., & Hanna, P. (2009). Musical structure retrieval by aligning self-similarity matrices. In *Proceedings of the 10th international society for music information retrieval conference (ismir)* (pp. 483–488). [3]

McKinney, W. (2010). Data structures for statistical computing in Python. In S. van der Walt & J. Millman (Eds.), *Proceedings of the 9th Python in science conference (SciPy2010)* (pp. 51–56). [106]

Müller, M. (2007). *Information retrieval for music and motion* (Vol. 6). Springer Heidelberg. [92]

Müller, M., & Ewert, S. (2008, September). Joint structure analysis with applications to music annotation and synchronization. In *Proceedings of the 9th international conference on music information retrieval (ISMIR)* (pp. 389–394). Philadelphia, Pennsylvania, USA. [3]

Müller, M., Ewert, S., & Kreuzer, S. (2009). Making chroma features more robust to timbre changes. In *Ieee international conference on acoustics, speech and signal processing (icassp)* (pp. 1877–1880). [47]

Niedermayer, B. (2012). *Accurate audio-to-score alignment - data acquisition in the context of computational musicology* (Unpublished doctoral dissertation). Johannes Kepler Universität, Linz. [3]

Nienhuys, H.-W., & Nieuwenhuizen, J. (2003). LilyPond, a system for automated music engraving. In *Proceedings of the XIV colloquium on musical informatics (XIV CIM 2003)* (Vol. 1, pp. 167–171). [107]

Özkan, I. H. (2006). *Türk mûsikısı nazariyatı ve usûlleri: Kudüm velveleleri*. Ötüken Neşriyat (in Turkish). [73, 75]

Paulus, J., Müller, M., & Klapuri, A. (2010). State of the art report:

Audio-based music structure analysis. In *Proceedings of 11th international society for music information retrieval conference (ismir)* (pp. 625–636). [47, 84]

Pearce, M. T., Müllensiefen, D., & Wiggins, G. A. (2010). Melodic grouping in music information retrieval: New methods and applications. In *Advances in music information retrieval* (pp. 364–388). Springer. [29]

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, *12*(Oct), 2825–2830. [106]

Pikrakis, A., Theodoridis, S., & Kamarotos, D. (2003). Recognition of isolated musical patterns using context dependent dynamic time warping. *IEEE Transactions on Speech and Audio Processing*, *11*(3), 175–183. [3]

Popescu-Judetz, E. (1996). *Meanings in Turkish musical culture*. Istanbul: Pan Yayıncılık. [5, 6]

Porter, A., & Serra, X. (2014). An analysis and storage system for music research datasets. In *1st international digital libraries for musicology workshop* (p. 1-3). London, United Kingdom. [96]

Porter, A., Sordo, M., & Serra, X. (2013a). Dunya: A system for browsing audio music collections exploiting cultural context. In *14th international society for music information retrieval conference* (p. 101-106). Curitiba, Brazil. Retrieved from `files/publications/porter-ISMIR-2013.pdf` [96]

Porter, A., Sordo, M., & Serra, X. (2013b, 11). Dunya: A system for browsing audio music collections exploiting cultural context. In *Proceedings of 14th international society for music information retrieval conference (ismir 2013)*. Curitiba, Brazil. [11]

Powers, et al., H. S. (n.d.). *Mode*. Grove Music Online, Oxford Music. Online: `http://www.oxfordmusiconline.com/subscriber/article/grove/music/43718pg5S`. [57]

Pérez, F., & Granger, B. E. (2007). IPython: A system for interactive scientific computing. *Computing in Science & Engineering*, *9*(3), 21-29. Retrieved from `http://scitation.aip.org/content/aip/journal/cise/9/3/10.1109/MCSE.2007.53` [107]

Salamon, J., & Gómez, E. (2012). Melody extraction from polyphonic music signals using pitch contour characteristics. *IEEE Transactions on Audio, Speech, and Language Processing*, *20*(6), 1759-1770. [48, 49, 50, 115]

Şentürk, S. (2011). *Computational modeling of improvisation in Turkish folk music using variable-length Markov models* (Unpublished master's thesis). Georgia Institute of Technology. [3, 13, 54]

Şentürk, S., Ferraro, A., Porter, A., & Serra, X. (2015). A tool for the analysis and discovery of Ottoman-Turkish makam music. In *Extended abstracts for the late breaking demo session of the 16th international society for music information retrieval conference (ismir)*. Málaga, Spain. [11, 43]

Şentürk, S., Gulati, S., & Serra, X. (2013). Score informed tonic identification for makam music of Turkey. In A. d. S. Britto Jr., F. Gouyon, & S. Dixon (Eds.), *Proceedings of 14th international society for music information retrieval conference (ismir 2013)* (pp. 175–180). Curitiba, Brazil: Pontifícia Universidade Católica do Paraná. Retrieved from http://sertansenturk.com/uploads/publications/senturk2013karar_ismir.pdf [21, 22, 56, 58, 68, 69, 88, 111]

Şentürk, S., Gulati, S., & Serra, X. (2014). Towards alignment of score and audio recordings of Ottoman-Turkish makam music. In A. Holzapfel (Ed.), *Proceedings of 4th international workshop on folk music analysis* (pp. 57–60). Istanbul, Turkey: Computer Engineering Department, Boğaziçi University. Retrieved from http://sertansenturk.com/uploads/publications/senturk2014alignment_fma.pdf [22, 92, 111]

Şentürk, S., Holzapfel, A., & Serra, X. (2012, 12/07/2012). An approach for linking score and audio recordings in makam music of Turkey. In *2nd compmusic workshop* (pp. 95–106). Istanbul, Turkey. Retrieved from http://mtg.upf.edu/system/files/publications/20-Sertan-Senturk-2nd-CompMusic-Workshop-2012_0.pdf [27, 48, 105, 117]

Şentürk, S., Holzapfel, A., & Serra, X. (2014, 3). Linking scores and audio recordings in makam music of Turkey. *Journal of New Music Research*, *43*(1), 34–52. Retrieved from http://sertansenturk.com/uploads/publications/senturk2014linking_jnmr.pdf [xxxi, 21, 22, 40, 43, 47, 49, 54, 58, 88, 90, 91, 92, 109, 111, 117]

Şentürk, S., Koduri, G. K., & Serra, X. (2016). A score-informed computational description of svaras using a statistical model. In *Proceedings of 13th sound and music computing conference (smc 2016)* (p. 427-433). Hamburg, Germany: Zentrum für Mikrotonale Musik und Multimediale Komposition (ZM4) Hochschule für Musik und Theater. Retrieved

from http://sertansenturk.com/uploads/publications/senturk2016notemodel_smc.pdf [58, 111]

Şentürk, S., & Serra, X. (2016a). Composition identification in Ottoman-Turkish makam music using transposition-invariant partial audio-score alignment. In *Proceedings of 13th sound and music computing conference (smc 2016)* (p. 434-441). Hamburg, Germany: Zentrum für Mikrotonale Musik und Multimediale Komposition (ZM4) Hochschule für Musik und Theater. Retrieved from http://sertansenturk.com/uploads/publications/senturk2016compositionIdentification_smc.pdf [111]

Şentürk, S., & Serra, X. (2016b). A method for structural analysis of Ottoman-Turkish makam music scores. In *Proceedings of 6th international workshop on folk music analysis (fma 2016)* (pp. 39–46). Dublin, Ireland: Dublin Institute of Technology. Retrieved from http://sertansenturk.com/uploads/publications/senturk2016symbolicanalysis_fma.pdf [21, 30, 109]

Serra, J. (1982). *Image analysis and mathematical morphology*. London.: Academic Press.[Review by Fensen, EB in: J. Microsc. 131 (1983) 258.] Review article General article, Technique Microscopy Staining, Mathematics, Cell size (PMBD, 185707888). [85]

Serrà, J., Serra, X., & Andrzejak, R. G. (2009). Cross recurrence quantification for cover song identification. *New Journal of Physics*, *11*(9). Retrieved from http://stacks.iop.org/1367-2630/11/i=9/a=093017 [3, 40, 47]

Serra, X. (2011). A multicultural approach in music information research. In *Proceedings of 12th international society for music information retrieval conference (ismir)* (p. 151-156). [3]

Serra, X. (2014, 27/01/2014). Creating research corpora for the computational study of music: the case of the CompMusic project. In *AES 53rd international conference on semantic audio.* London, UK: AES. [7, 9]

Shetty, S., & Achary, K. (2009). Raga mining of Indian music by extracting arohana-avarohana pattern. *International Journal of Recent Trends in Engineering*, *1*, 362–366. [58]

Signell, K. L. (1986). *Makam: Modal practice in Turkish art music*. Da Capo Press. [6]

Şimşek, B. Ö., Bozkurt, B., & Akan, A. (2016). Fundamental frequency estimation for heterophonical Turkish music by using VMD. In *2016 24th signal processing and communication application conference (siu)* (pp. 1625–1628). [47, 52, 117]

Smith III, J. O., & Serra, X. (1987). *Parshl: an analysis/synthesis program for non-harmonic sounds based on a sinusoidal representa-*

tion. CCRMA, Department of Music, Stanford University. [57, 110]

Srinivasamurthy, A., Holzapfel, A., & Serra, X. (2014). In Search of Automatic Rhythm Analysis Methods for Turkish and Indian Art Music. *Journal of New Music Research*, *43*(1), 97–117. [2]

Suma, S. M., & Koolagudi, S. G. (2015). Information systems design and intelligent applications: Proceedings of second international conference india 2015, volume 1. In K. J. Mandal, C. S. Satapathy, M. Kumar Sanyal, P. P. Sarkar, & A. Mukhopadhyay (Eds.), (pp. 865–875). New Delhi: Springer India. [58]

Temperley, D. (2004). *The cognition of basic musical structures*. MIT press. [29]

Temperley, D., & Marvin, E. W. (2008). Pitch-class distribution and the identification of key. *Music Perception: An Interdisciplinary Journal*, *25*(3), 193–212. [54]

Tenney, J., & Polansky, L. (1980). Temporal gestalt perception in music. *Journal of Music Theory*, *24*(2), 205–241. [29]

Thomas, V., Fremerey, C., Müller, M., & Clausen, M. (2012). Linking sheet music and audio - Challenges and new approaches. In M. Müller, M. Goto, & M. Schedl (Eds.), *Multimodal music processing* (Vol. 3, pp. 1–22). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik. [2, 47]

Tomita, E., Tanaka, A., & Takahashi, H. (2006). The worst-case time complexity for generating all maximal cliques and computational experiments. *Theoretical Computer Science*, *363*(1), 28 - 42. Retrieved from http://www.sciencedirect.com/science/article/pii/S0304397506003586 [34]

Townsend, M., & Sandler, M. B. (1993). Pattern recognition for formant trajectories using the Hough transform. In *14. colloque sur le traitement du signal et des images, fra, 1993* (pp. 1355–1358). [85]

Tura, Y. (1988). *Türk musıkisinin meseleleri*. Pan Yayıncılık, Istanbul (in Turkish). [5]

Tzanetakis, G., Kapur, A., Schloss, W. A., & Wright, M. (2007). Computational ethnomusicology. *Journal of interdisciplinary music studies*, *1*(2), 1–24. [3]

Uyar, B., Atlı, H. S., Şentürk, S., Bozkurt, B., & Serra, X. (2014). A corpus for computational research of Turkish makam music. In *1st international digital libraries for musicology workshop* (pp. 57–63). London, United Kingdom. Retrieved from http://sertansenturk.com/uploads/publications/uyar2014corpus_dlfm.pdf [12, 43, 69]

Walt, S. v. d., Colbert, S. C., & Varoquaux, G. (2011). The NumPy array: A structure for efficient numerical computation. *Computing in Science & Engineering*, *13*(2), 22-30. Retrieved from http://scitation.aip.org/content/aip/journal/cise/13/2/10.1109/MCSE.2011.37 [106]

Wang, A. L.-C. (2003). An industrial strength audio search algorithm. In *Proceedings of 4th international society for music information retrieval conference (ismir)* (pp. 713–718). [3]

Index