OPTICAL MUSIC RECOGNITION OF HAMPARSUM NOTE MANUSCRIPTS

by

Kürşat Çınar

Submitted to Graduate School of Natural and Applied Sciences

in Partial Fulfillment of the Requirements

for the Degree of Master of Science in

Computer Engineering

Yeditepe University

2019

OPTICAL MUSIC RECOGNITION OF HAMPARSUM NOTE MANUSCRIPTS

APPROVED BY:

Assist. Prof. Dr. Dionysis Goularas .... [illegible]
(Thesis Supervisor)
(Yeditepe University)

Prof. Dr. Cem Ünsalan .... [illegible]
(Marmara University)

Assist. Prof. Dr. Onur Demir .... [illegible]
(Yeditepe University)

DATE OF APPROVAL: ..../..../2019

ACKNOWLEDGEMENTS

I am grateful for the support and help of my supervisor, Assist. Prof. Dr. Dionysis Goularas. Pursuing my thesis under his supervision has been an experience which broadens the mind and presents an unlimited source of learning.

I would like to thank my beloved wife Erinç, for her encouraging attitude when I lost my motivation, also my pretty little son Doruk for letting me to work instead of playing with him, and finally all my family for their reliance on me.

ABSTRACT

OPTICAL MUSIC RECOGNITION OF HAMPARSUM NOTE MANUSCRIPTS

This study presents an optical music recognition method for manuscript music compositions written with the Hamparsum musical note system. This musical note system, which was developed by the famous musician Hamparsum Limonciyan, was widely used by musicians and composers during the last two centuries of the Ottoman Empire, when it was mentioned that the golden age of music was experienced. Although the use of various notation systems continued in previous periods, there is no other notation system as widely used as Hamparsum note. Many popular musical compositions produced during and after that period, coinciding with the reign of Sultan Selim III, who is also a musician, were recorded with Hamparsum's note. The symbols that make up the Hamparsum note are the fact that the Ottoman music circles are not very familiar with them, and that they are mostly used in the ancient Armenian Khaz notation system, but the reason why it was widely used is due to the fact that Turkish music has been successfully adapted to the maqam and pitch with accidentals structure. Because of the fact that today European notation is in a worldwide position, Hamparsum has no old popularity. However, it is still known to continue to be used in Armenian Gregorian churches. In terms of the symbols and general structure of the Hamparsum notation, it has significant differences with the European notation. Before anything else, a note in the Hamparsum system can consist of more than one symbol. The method presented in this study includes an optical music recognition process that includes the methods of classifying the features extracted by applying a 2D Gabor filter bank with the Support Vector Machines method and then matching the identified symbols to the most appropriate Hamparsum note pattern. The output of the application developed for testing the proposed method consists of pre-defined codes that indicate the European note equivalents. This study aims at passing the musical works written in Hamparsum with a machine-readable translation and contributing to the dissemination of this unique cultural heritage. This suggested method can be applied to similar notations by using the same machine learning procedure and by modifying the template matching parameters.

ÖZET

HAMPARSUM NOTASI EL YAZMALARININ OPTİK MÜZİK TANIMASI

Bu çalışma, Hamparsum nota sistemi ile yazılmış el yazması müzik eserleri için bir optik müzik tanıma yöntemi sunmaktadır. Ünlü müzisyen Hamparsum Limonciyan tarafından geliştirilen bu sistem, Osmanlı İmparatorluğu'nun son iki yüzyılı boyunca, musikinin altın çağının yaşandığı belirliten bir dönemde, müzisyenler tarafından yaygın olarak kulanılmıştır. Önceki dönemlerde çeşitli nota sistemlerinin kullanımı devam ettiyse de, Hamparsum kadar kabul gören başka bir nota sistemi bulunmamaktadır. Kendisi de bir müzisyen olan Sultan III. Selim'in hükümdarlığına rastlayan bu dönem ve sonrasında üretilen bir çok popüler eser, Hamparsum notası ile kayıt altına alınmış ve günümüze kadar ulaştırılabilmesi mümkün olmuştur. Hamparsum notasını oluşturan semboller, Osmanlı müzik çevrelerinin çok da aşina olmadığı, çoğunlukla antik Ermeni Khaz nota karakterlerinden oluşmasına rağmen, yaygın kabul görmesinin nedeni Türk musikisinin makam ve arızalı perde yapısına başarılı bir şekilde uyarlanabilmiş olmasıdır. Günümüzde Avrupa notasyonunun dünya çapında geçerli bir konumda olması nedeniyle, Hamparsum notası eski popüleritesine sahip değildir. Ancak halen başta Ermeni Gregoryen kiliseleri olmak üzere, kısmen de olsa kullanımının devam ettiği bilinmektedir. Hamparsum notası barındırdığı semboller ve genel yapısı itibarıyla, Avrupa notasyonuna göre önemli farklılıklar barındırmaktadır. Her şeyden önce, Hamparsum sisteminde bir nota birden fazla sembolden oluşabilmektedir. Bu çalışmada sunulan yöntem, bir 2D Gabor filtre bankası uygulanarak çıkarılan özelliklerin, Destek Vektör Makineleri yöntemi ile sınıflandırılması, ardından belirlenmiş sembollerin en uygun Hamparsum nota şablonu ile eşleştirilmesi yöntemlerini barındıran bir optik müzik tanıma işlemini kapsamaktadır. Önerilen yöntemin test edilmesi için geliştirilen uygulamanın çıktısı, Avrupa nota karşılıklarını belirten ön tanımlı kodlardan oluşmaktadır. Bu çalışma Hamparsum notası ile yazılmış müzik eserlerinin makine tarafından okunabilecek bir çeviriden geçirilmesi ve bu eşsiz kültürel mirasın yaygınlaştırılmasına katkı sunma amacı da taşımaktadır. Önerilen bu yöntemin, aynı makine öğrenme prosedürünü kullanarak ve şablon eşleştirme parametrelerini değiştirerek benzer notasyonlara da uygulanabileceği düşünülmektedir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

θ	Orientation of 2D Gabor filter
φ	Phase offset of 2D Gabor filter
γ	Spatial aspect ratio of the Gaussian curve
σ	Standard deviation of the Gaussian envelope
λ	Wavelength of the sinusoidal factor

API	Application programming interface
ARFF	Attribute relation file format
CV	Cross validation
DBN	Dynamic Bayesian network
GCN	Gong-Che notation
GUI	Graphical user interface
HBM	Hellenic Byzantine music
HMM	Hidden Markov models
LTO	Leave ten out
OCR	Optical character recognition
OMR	Optical music recognition
ROI	Region of interest
SVM	Support vector machine
TSM	Thai sheet music

1. INTRODUCTION

Music is the ordering of tones or sounds to produce compositions with unity and continuity [1]. Transcribing music compositions is rather critical so that the existing harmony could be passed on to next generations. Otherwise, it is inevitable that the pieces will change over time and perhaps lose the structure they contain.

Since the beginning of mankind, musical sounds and rhythmic thoughts have been wanted to be recorded through some written codes and signs, but from the First Age until the Middle Ages there has not been any significant work of music remained. It is known that various cultures throughout history have developed music writing systems, even though they are primitive. In the Middle Ages (between the 9th and the 13th centuries), the first music script has come out with the so-called "neuma" signs (dots, curves, lines, bonds, etc.) written on the lyrics, in order to express ups and downs of the songs played in the churches and monasteries [2]. The neumas, which have been used without precisely given sound heights and durations, could be used to indicate the remembrance of the melodies. While the first notation marks in music consisted of the interpretation of a known melody, the indication of the tones and the temporal values of the sounds with the markings used in their writing has been a milestone in the history of the music writing.

The invention of the Hamparsum notation has a revolutionary effect in the Ottoman-Turkish music tradition. The tradition of this music is based on the transference by practising. Within the framework of the master and apprentice relationship, the existing knowledge was transferred to the next generation and the same fiction was followed in the training of future generations. Although various notation systems were invented and used by small groups in the period, there was not a generally accepted system until the invention of the Hamparsum notation. Moreover, Hamparsum's notation system was sufficient in terms of satisfying the musical needs, and due to its similarity to the Western notation, it was accepted by different circles and became widespread. It has been widely used by Ottoman - Turkish music circles for more than two centuries. On this occasion, a large number of precious music pieces have reached the present day. Hamparsum notation today no longer has the old popularity, but still continues to be used common spelling of divines in Armenian churches located in Turkey. Ottoman - Turkish music has an extremely rich heritage. However, due to the lack of a standardized notation system adopted for many years, this heritage has been transmitted to a very limited extent. At the moment, as a result of the widespread acceptance of internationally validated western notation, the number of people who are able to use the Hamparsum notation has decreased considerably. The ability to read and translate manuscripts of Hamparsum notation has the risk of becoming more and more difficult.

The possibility of automating the translation of the Hamparsum notation scores was considered as a method that could contribute to the solution of the problem. In the literature, there is no other study suggesting a method for automated translation of Hamparsum note manuscripts. For this reason, OCR and OMR methods applied on some other domains were examined at the beginning.

In this study, a method for the recognition of manuscripts in Hamparsum notation and to translate them into the Western notation is proposed. Leon Hanciyan's collection of musical works written in the Hamparsum notation which is currently hosted by the Ottoman Archives Department in Istanbul, was accepted as the main data source of this study. The main challenge was to translate the manuscripts in Hamparsum music notation into Western music notation. However, the lack of a clean set of data for both training and testing, as an impediment made the pre-processing section more difficult than expected.

In pre-processing, a special noise reduction technique and some manual operations have been applied to isolate the Hamparsum symbols. After obtaining images that contain only Hamparsum components, a list of well known methods has been applied for symbol segmentation and feature extraction. Classification of training and testing data sets was achieved by using Support Vector Machine (SVM) classifier. After recognition, a cross validation technique was employed to scale the success rates of the system. Furthermore, utilizing the knowledge of symbols identities, matching procedure of main symbols and auxiliary symbols has been achieved. To be able to detect the corresponding musical note, a template matching method was applied on the candidate notes.

This thesis has sections as follows: Informations gathered by a literature research from the OMR and OCR point of view are presented in the next chapter, Chapter 2. Prior information about basic musical concepts and Hamparsum notation structure were shared in Chapter 3. Detailed explanations of methods that have been utilized would be found in

Chapter 4. Analysis and design theory of the proposed system were explained in Chapter 5. The graphical user interface and the system description in terms of implemented program were presented in Chapter 6. The accuracy of the system and results in detail were shared in Chapter 7. As a result, in the last chapter, Chapter 8, the study as the first effort on OMR of the Hamparsum notation was concluded with a few sentences.

2. RELATED WORKS

Optical music recognition (OMR) literature consists of dozens of researches that mostly related with European notation system domain. Besides, any OMR study which models Hamparsum notation system has been published yet. Somehow, not only European notation, but also some local traditional notation systems has been covered under OMR major. Even if OMR has many distinctions from optical character recognition (OCR), they have significant intersective stages. To be able to kick off an OMR study on any notation systems or an OCR study on any languages, a comprehensive research is required. The studies summarized below were chosen among those thought to be important as a result of this research process.

A study on an OMR method for typeset music scores in European notation system has been proposed by Florence Rossant in 2002 [3]. The input range of the study in terms of the data set has been limited by digitally scanned and already binarized typesets. Even if the data set covers only typeset scores, during optical scanning process it may remain significant spotted components on the image. Therefore, pre-processing and analysis stages in Rossant's publication is rather related with this study in terms of symbol isolation, generating hypothesis on each symbol, analyzing symbols via using produced hypotheses and making decisions on matching symbols with music notes by using all the generated information. Having staff lines and being able to utilize them as a reference for scaling, correcting and evaluating is an advantageous property for the domain of European notation point of view. The staff spacing which remains unchanged for the whole score, provides an important information for normalizing the size of image. Furthermore, to be able to detect the pitch of a specific note by utilizing the position of the head of note symbol on staff lines establishes a comfort area for the segmentation. And moreover, in case of a skewness on the image, it would be possible to correct by utilizing staff lines. Staff lines produces many benefits until here, however comprising intersection and connection with the symbols makes the isolation and segmentation of notes a bit more complicated. At the beginning of the segmentation stage, staff lines and irrelevant spots, lines etc. to be removed. The cleaning procedure sometimes may behave aggressive and remove some necessary symbols like accidentals and rests; which would cause of information loss for global decision stage. In OMR literature, there are some widespread methods called for symbol recognition like neural networks, moments, morphological techniques. However, for this study, template matching method has been preferred. Rossant explains that, basically template matching is the method of checking the correlation with the training data set and even if somewhat loss of information on pre-processing and segmentation stages, at least correlation value would provide a significant information on remaining components. The disadvantageous attitude of template matching is the font style sensitivity, however it is possible to produce a solution for that by providing a new set of training data. In analysis of symbols section, correlation values of each isolated symbols would be calculated according to various standard geometric criterias. In case of ensuring thresholds values, the detection of the model with the highest correlation would be achieved and denoted. Before the global decision section, all the hypotheses generated so far needs to be merged and evaluated together. In the end, the proposed OMR system for typeset music scores in European notation has an average success rate above 97 percent.

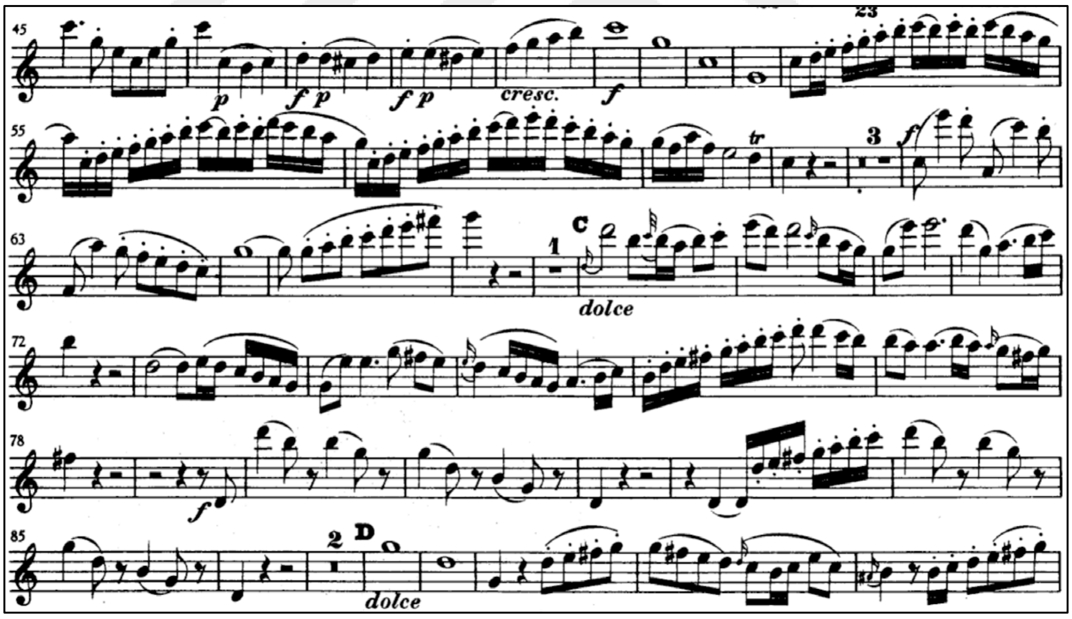
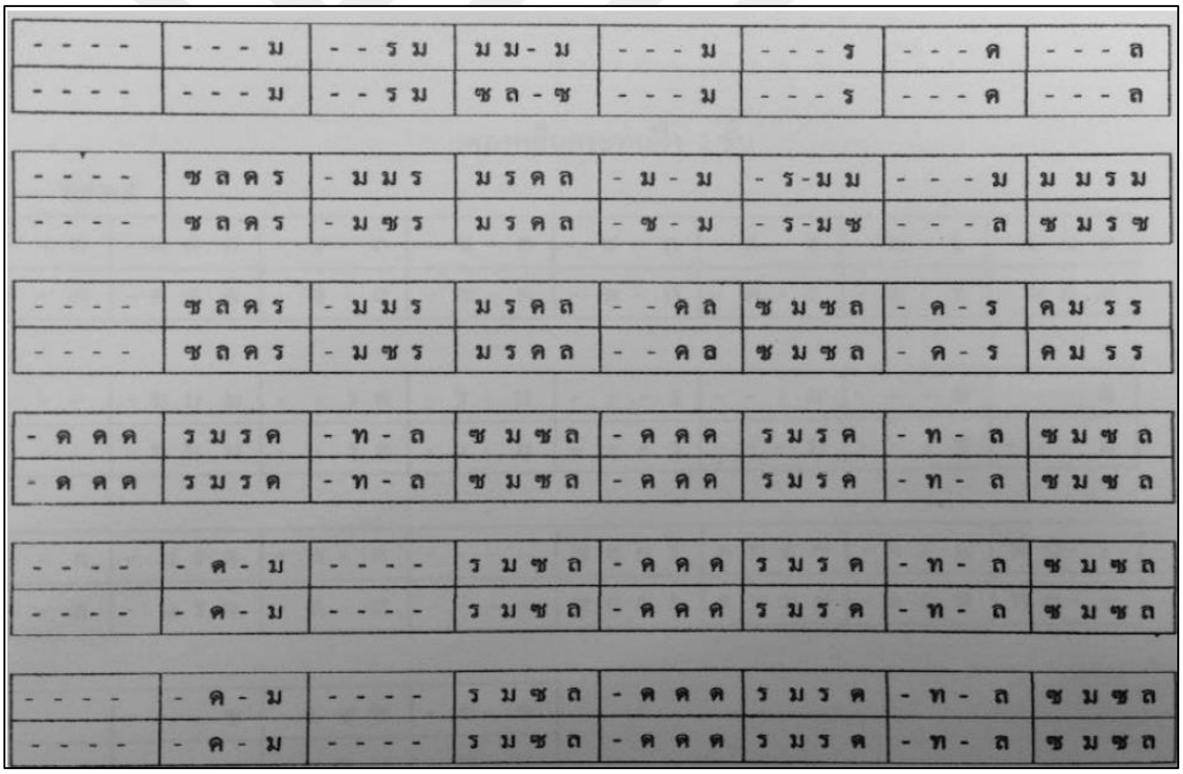


Figure 2.1. European music notation

In 2012, Rebelo et al. published an article to evaluate the recent status of the popular methods in the OMR in the field of European music notation [4]. A typical framework has been described in four main stages: preprocessing, recognition of musical symbols, reconstruction of the musical information and construction of a musical notation model. The image preprocessing would produce the input of recognition stage as usual. Usage of various techniques such as enhancement, binarization, elimination of noises, blurring, de-skewing etc. for preprocessing would help to make recognition process more robust. Binarization is usually a mandatory process in order to fetch the data that are required for recognition and to eliminate unnecessary details which may cause kind of ambiguity for the next stages. In this publication, various thresholding techniques are referred to among the binarization techniques. Furthermore, taking staff lines as reference in terms of length, thickness and vertical line distance within same staff would be critical to be able to normalize the scale of each component. The recognition stage contains three subparts: detection and removal of staff lines, segmentation of symbols and recognition of symbols. For the reconstruction of musical notation the individual symbols recognized in previous stage would be combined to be able to generate musical symbols. However, the likelihood of losing necessary data in preprocessing stage or producing erroneous information in recognition stage is an obstacle to boosted results. Therefore, some graphical and syntactical rules have been applied on to validate and solve irrelevancies. The final representation stage of the OMR framework collects all the information gained in prior stages and merge them to generate a graphical output of music publishing file such as MIDI or MusicXML. This framework is not represented as the immutable law of OMR in European music notation; one would narrow down or extend up the stages or add new methods or remove some of the methods, depending on domain requirements. Nevertheless, the described framework for the OMR of European music notation covers various popular techniques and presents a generic flow chart, which would be helpful for the researchers who are interested in the same domain.

The most of OMR related works are inherently based on European notation which is world wide accepted music writing method contemporarily. On the other hand, before European notation system became widespread, humanity has developed manifold music notation systems. A certain number of OMR related studies based on various traditional music notation systems has taken part in literature. Kusakunniran et al. [5] which have been proposed an OMR method for traditional Thai sheet music (TSM) which is based on Thai symbols to represent music notes in 2014. TSM system has a row and column based structure, which consists of cells with a note symbol inside of each cell. For that reason, a differentiation on the domains occurs among OMR studies. The flow of the study starts with the edge detection process of a binary image by using canny edge detection method in order to remove irrelevant scars and unnecessary layers. After that, a clean music score which includes necessary information for isolation and segmentation has been handled. In segmentation stage, removing of lines, detection of music lines and isolation of note symbols are the processes applied. Some statistical methods have been utilized through the lines detected in order to detect candidate music rows and candidate notes. To be able to apply a classification and recognition mechanism on detected candidate note symbols, support vector machine (SVM) method has been utilized. TSM consists of 8 symbols at all; SVM creates a separate classification models for each shape. Training these models via assigning 80 positive and 80 negative samples to every single note symbol has been preferred. The proposed method on OMR for traditional TSM has an accuracy higher than 80 percent.



- - - -	- - - ม	- - ร ม	ม ม - ม	- - - ม	- - - ร	- - - ค	- - - ล
- - - -	- - - ม	- - ร ม	ซ ล - ซ	- - - ม	- - - ร	- - - ค	- - - ล
- - - -	ซ ล ค ร	- ม ม ร	ม ร ค ล	- ม - ม	- ร - ม ม	- - - ม	ม ม ร ม
- - - -	ซ ล ค ร	- ม ซ ร	ม ร ค ล	- ซ - ม	- ร - ม ซ	- - - ล	ซ ม ร ซ
- - - -	ซ ล ค ร	- ม ม ร	ม ร ค ล	- - ค ล	ซ ม ซ ด	- ค - ร	ค ม ร ร
- - - -	ซ ล ค ร	- ม ซ ร	ม ร ค ล	- - ค ล	ซ ม ซ ล	- ค - ร	ค ม ร ร
- ค ค ค	ร ม ร ค	- ท - ด	ซ ม ซ ด	- ค ค ค	ร ม ร ค	- ท - ด	ซ ม ซ ด
- ค ค ค	ร ม ร ค	- ท - ล	ซ ม ซ ล	- ค ค ค	ร ม ร ค	- ท - ล	ซ ม ซ ด
- - - -	- ค - ม	- - - -	ร ม ซ ด	- ค ค ค	ร ม ร ค	- ท - ด	ซ ม ซ ด
- - - -	- ค - ม	- - - -	ร ม ซ ด	- ค ค ค	ร ม ร ค	- ท - ด	ซ ม ซ ด
- - - -	- ค - ม	- - - -	ร ม ซ ด	- ค ค ค	ร ม ร ค	- ท - ด	ซ ม ซ ด
- - - -	- ค - ม	- - - -	ร ม ซ ด	- ค ค ค	ร ม ร ค	- ท - ด	ซ ม ซ ด

Figure 2.2. Thai sheet music notation

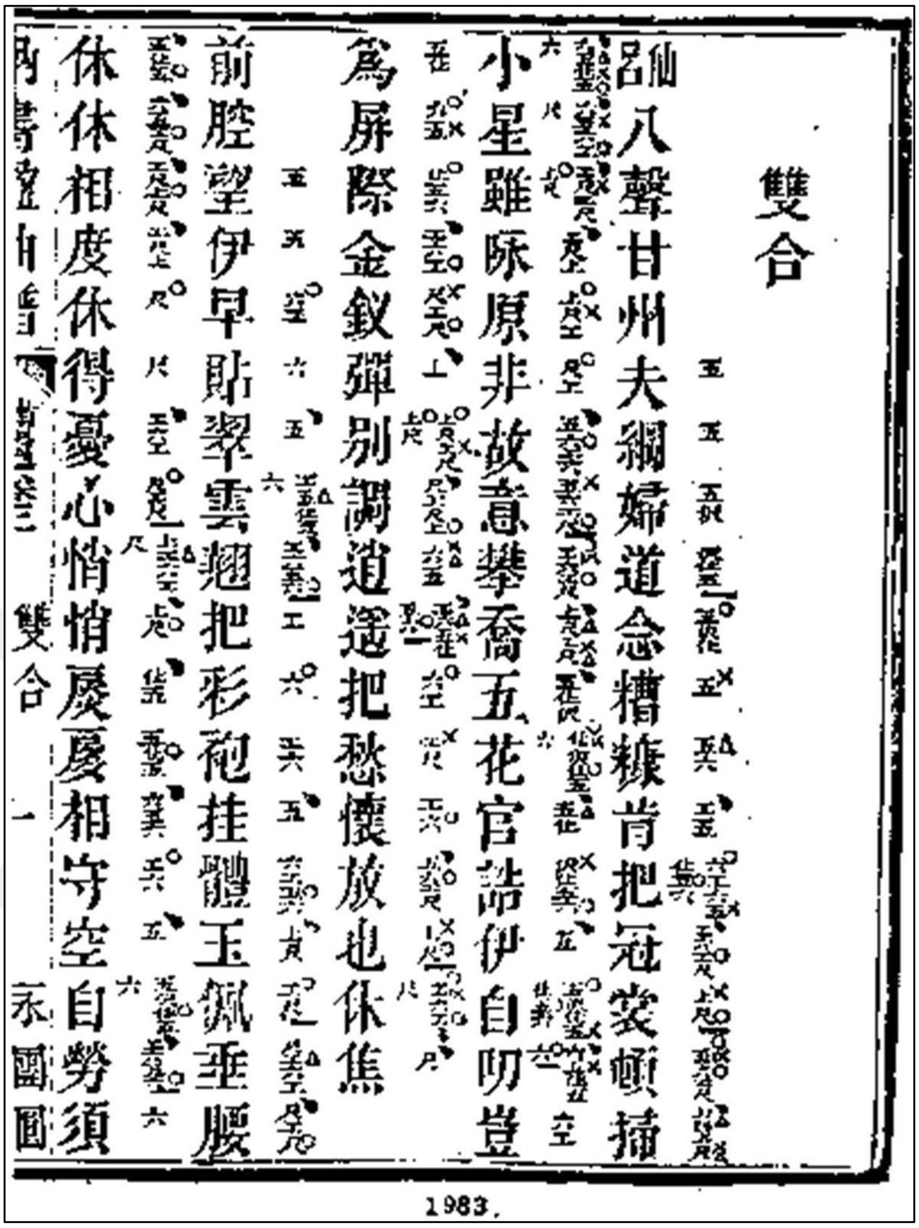


Figure 2.3. Gong-Che notation

An OMR system based on another traditional music notation has also been published by Chen and Sheu in 2014 [6]. The mentioned study proposes a method for recognition of Chinese Kunqu Opera scores written in Gong-Che notation (GCN) and transformation into a machine readable format. Pre-processing, segmentation, feature extraction, symbol recognition, musical semantics and audio representation stages has been covered for the proposed method. As similar with most OMR related work, Chen and Sheu have given significant importance on pre-processing operations, in order to boost overall performance of the system. GCN contains a static structure in terms of component layouts in the score sheet which appears to be an advantageous feature. Nevertheless, due to GCN's typological difficulties and inability to have satisfactory clean scanned data set, some sort of information around 2.7 percent has been lost in segmentation stage. In the feature extraction stage, four features have been specified according to a cellular structure. GCN contains Chinese letters as note symbols, and also some specialized symbols for other musical markers. Therefore, the domain of the study is relevant with both OMR and OCR in terms of symbol recognition process. The K-nearest neighbor (KNN) classifier, Bayesian decision theory and a genetic algorithm based on a heuristic search have been tested and utilized to achieve symbol recognition. Since the recognition rates varies between 42.62 percent and 67.26 percent, the need for more powerful approaches has been concluded.

In another study by Gezerlis and Theodoridis [7], a system for the off-line optical character recognition (OCR) of the orthodox Hellenic Byzantine Music (HBM) notation symbols has been proposed in 2002. A grayscale bitmap image database of approximately 18000 symbols has been created for the 71 distinct characters each has 250 different patterns. In the first step of pre-processing stage, grayscale images have been binarized and then, the removal of noises such as dot and hole elimination and image size normalization processes have been applied on each pattern. After binarization stage, feature generation part has been elaborated into structural features and statistical features. The utilized structural feature techniques are Euler numbers, principal axis and ratio of horizontal bounding rectangle. For statistical features, contour tracing, adaptive projection method (4-projections) and discrete wavelet transform methods have been employed. In contour tracing step, adaptive starting point method and approximation for the contours of each character by using Bezier splines of n^{th} order method have been utilized. Adaptive projection method consists of 4-projections such as horizontal, vertical, left diagonal and right diagonal. Discrete wavelet transform has been applied on the contour function and the projection vector in order to obtain final feature vector which gave the best success rates in classification. In classification stage, nearest neighbor and neural networks methods have been tested and compared. 50 samples of 250 for each symbol have been used for training and the remaining 200 have been used for testing. Neural networks achieved a 96.4 percent success rate while narest neighbor has 98.1 percent. Nearest neighbor gave better results and rationally it has been selected for the next steps. The main reason of the loss of success rates in both classification methods seems like the similarity of some characters shapes. Therefore a post classification method has been developed to be able to increase the accuracy of the low accurate recognized symbols and the success rate has been increased to 99.1 percent. For the overall evaluation of the proposed system, leave-ten-out cross validation method gave 99.4 percent of success rate. At the end of the paper, some other methods like template matching and Support Vector Machine (SVM) have been adviced for the future consideration of the domain.

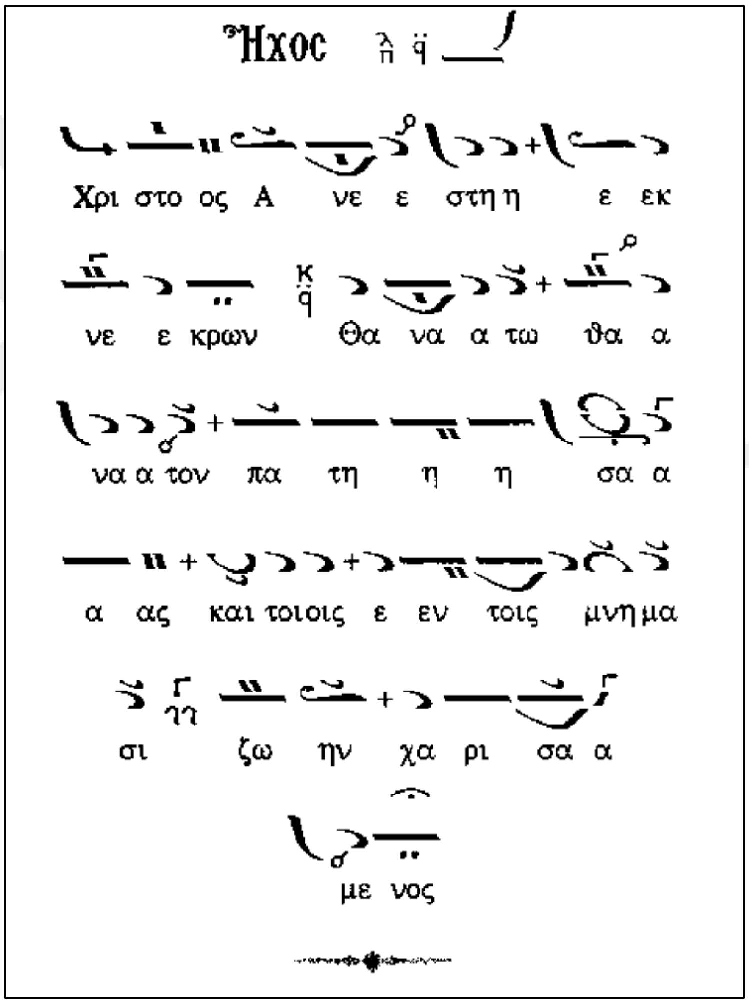

Ἦχος
Χρι στο ος Α νε ε στη η ε εκ
νε ε κρων Θα να α τω ϑα α
να α τον πα τη η η σα α
α ας και τοιοις ε εν τοις μνη μα
σι ζω ην χα ρι σα α
με νος

Figure 2.4. Hellenic Byzantine music notation

OMR is a customized sub-branch of OCR for music. Although not semantically, it may be helpful to look at pre-processing and recognition stages of other OCR studies. At first glance, although some of the Hamparsum symbols appear to be similar to Arabic letters typographically, there are actually significant distinctions. Nevertheless, it would be useful to take a look at the pros and cons of OCR researches in this area. Asebriy et al. [8] published an overview on various studies on offline handwriting Arabic character recognition systems and presented a general model of OCR system in 2014. The presented general model follows 4 main stages respectively, pre-procesing, character segmentation, features extraction and recognition and post-processing. A digital copy of an Arabic manuscript that was retrieved by using scanner, probably needs to be cleaned by removal of noises or de-skewing operations or some corrections in pre-processing stage. As long as retrieving a cleaned up sample of handwritten text, this will make consecutive segmentation and recognition stages more easier and more efficient. Arabic language has a writing direction of right to left and top to bottom. Therefore, segmentation of lines is the first step that needs to be overcome and then follows the segmentation of words, sub-words, and characters. As far as individual symbols or separate group of characters retrieved, feature extraction results would have a distinctive mission for their recognition accuracy. For the creation of the training model and the classification of testing data set, the extracted features would be the basic building blocks. In order to be able to tune the recognition testing results, post-processing stage would play the finishing role. The offered model here for the recognition of handwritten Arabic texts is quite similar to the presented models for various OMR systems above. Handwriting Arabic has a cursive typology, and almost every character has different forms, depending on its position in the word in which it exists. Thereof, many studies on handwritten Arabic OCR has been proposed by using Hidden Markov Models (HMM) which is priorly popular statistical model in the area of speech recognition. One of the investigated studies [9] achieved a success of 86,73 percent, however in case of an additional re-ranking process has been applied after HMM, the accuracy increased to 89,24 percent. The system also combined two different features: intensity features to train HMM and topological features in re-ranking stage to be able to achieve improved accuracy. In another presented study [10], H. Alkhateeb et al. has been compared two different recognition approaches: HMM and Dynamic Bayesian Network (DBN) classifiers. As a result of aforementioned study, HMM has produced better accuracy than DBN. A novel study for offline Arabic handwritten recognition based on structural techniques has been presented in 2013 [11]. A polygonal approximation for segmentation stage has been used and the recognition stage has been established by using a fuzzy polygon matching algorithm. Moreover, some prototype selection and lexicon reduction techniques have been employed to make the system generate better results. The accuracy of the recognition was 79,58 percent. Kessentini et al. [12] have been tested a multi-stream HMM for offline handwritten Arabic recognition. However the best accuracy achieved was 79,80 percent. As a result of [8], in the domain of offline cursive Arabic texts, the best results has been obtained by using HMM and re-ranking.

As a summary, almost every OMR model covers these four main stages: pre-processing, segmentation, feature extraction, recognition. In pre-processing phase, binarization and elimination of noises keeps the working area clean. De-skewing and scaling techniques enable segmentation of more standard and comparable symbol samples. Symbols that can be successfully segmented with minimal data loss directly affect the feature extraction stage. Likewise, well extracted features has a significant impact on recognition accuracy. It would not be wrong to pronounce it aloud: the success rate of each step has a decisive importance for the accuracy of the next step. Regardless of the system or alphabet being studied, the typology of symbols and characters is also the most critical indicator in determining the methods to be used for segmentation, feature extraction and recognition stages. Since there has not been any prior OMR study on Hamparsum notation, a model to compare directly could not be found. European music notation has a system that displays the notes and symbols on the stave. Arabic language also has a distinctive cursive typology and this is the main reason why Hidden Markov Model (HMM) is very popular for Arabic handwriting OCR studies, but not very popular among OMR systems. However, it would not produced satisfactory results yet to obtain a tangible success on Arabic manuscripts. Thai sheet music (TSM) notation recognition system promotes SVM classifier for the recognition. Gong-Che notation (GCN) has not achieved sufficient accuracy by KNN and Bayesian decision theory methods and a genetic heuristic search algorithm. Hellenic Byzantine music (HBM) notation recognition system has one of the most accurate results for handwritten OMR domain. The authors presented some techniques including contour tracing, adaptive 4-projections, discrete wavelet transform, nearest neighbor, neural networks, etc. Although adequate results were obtained, it was suggested to test template matching and Support Vector Machine (SVM) methods in future studies.

3. BACKGROUND

A musical notation might be defined as a system to reflect sounds on the paper by following a standard and predefined set of rules via using a specific character set. Basically, character combinations should represent sound frequencies, rest descriptors and duration indicators to be able to obtain musical compositions. Hamparsum Limonciyan, the inventor of the Hamparsum notation, developed his system to compose Ottoman classical music pieces and Armenian Church liturgies. Most of the symbols had been deployed from Armenian Khaz notation and adapted to Turkish music system. Hamparsum symbols consist of 13 common note symbols and additional auxiliary symbols which may differ in terms of shaping according to the author's preference and writing style. Nevertheless, commonly used auxiliary symbol representations have been taken into consideration by depending on the observations gathered during this study. Before going further of the recognition system, a clear understanding on basic musical knowledge and a description of the Hamparsum notation may help to comprehend methodology and design stages.

In the European notation, each note with the same duration is represented by the same symbols and separated from each other depending on their position on the stave. In Hamparsum notation, each note has a different body symbol and they are able to indicate their durations, octaves and accidental frequencies by using auxiliary symbols at the top and the bottom. In the following subsections, a brief information on sound frequencies and comparison between Turkish and European music structures at first and then a detailed explanation on Hamparsum music notation system and the investigation on the data set that has been used in this study will be covered.

3.1. MUSICAL BACKGROUND

The names of the notes have meanings in terms of tone frequencies. Each note indicates a digital frequency value behind the names. There are two different musical note representations that are globally accepted today (Table 3.1).

The first representation makes use of letters like C, D, E, F, G, A, B and the second one uses specialized names respectively Do, Re, Mi, Fa, Sol, La Si. These two types of representation have an exact equivalency in between. The frequency values of the notes are as in the following table (Table 3.2).

Table 3.1. European music note representations

	EUROPEAN MUSIC NOTES						
1st Representation	C	D	E	F	G	A	B
2nd Representation	Do	Re	Mi	Fa	Sol	La	Si

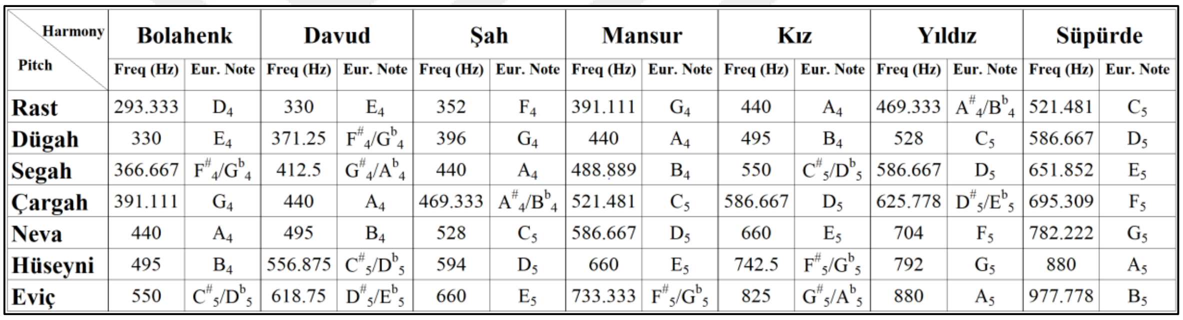
Table 3.2. The frequency values of the notes

Note	Frequency (Hz)	Wavelength (cm)	Note	Frequency (Hz)	Wavelength (cm)	Note	Frequency (Hz)	Wavelength (cm)
C_3	130.81	263.74	C_5	523.25	65.93	C_7	2093.00	16.48
$C^{\#}_3/D^b_3$	138.59	248.93	$C^{\#}_5/D^b_5$	554.37	62.23	$C^{\#}_7/D^b_7$	2217.46	15.56
D_3	146.83	234.96	D_5	587.33	58.74	D_7	2349.32	14.69
$D^{\#}_3/E^b_3$	155.56	221.77	$D^{\#}_5/E^b_5$	622.25	55.44	$D^{\#}_7/E^b_7$	2489.02	13.86
E_3	164.81	209.33	E_5	659.25	52.33	E_7	2637.02	13.08
F_3	174.61	197.58	F_5	698.46	49.39	F_7	2793.83	12.35
$F^{\#}_3/G^b_3$	185.00	186.49	$F^{\#}_5/G^b_5$	739.99	46.62	$F^{\#}_7/G^b_7$	2959.96	11.66
G_3	196.00	176.02	G_5	783.99	44.01	G_7	3135.96	11.00
$G^{\#}_3/A^b_3$	207.65	166.14	$G^{\#}_5/A^b_5$	830.61	41.54	$G^{\#}_7/A^b_7$	3322.44	10.38
A_3	220.00	156.82	A_5	880.00	39.20	A_7	3520.00	9.80
$A^{\#}_3/B^b_3$	233.08	148.02	$A^{\#}_5/B^b_5$	932.33	37.00	$A^{\#}_7/B^b_7$	3729.31	9.25
B_3	246.94	139.71	B_5	987.77	34.93	B_7	3951.07	8.73
C_4	261.63	131.87	C_6	1046.50	32.97	C_8	4186.01	8.24
$C^{\#}_4/D^b_4$	277.18	124.47	$C^{\#}_6/D^b_6$	1108.73	31.12	$C^{\#}_8/D^b_8$	4434.92	7.78
D_4	293.66	117.48	D_6	1174.66	29.37	D_8	4698.63	7.34
$D^{\#}_4/E^b_4$	311.13	110.89	$D^{\#}_6/E^b_6$	1244.51	27.72	$D^{\#}_8/E^b_8$	4978.03	6.93
E_4	329.63	104.66	E_6	1318.51	26.17	E_8	5274.04	6.54
F_4	349.23	98.79	F_6	1396.91	24.70	F_8	5587.65	6.17
$F^{\#}_4/G^b_4$	369.99	93.24	$F^{\#}_6/G^b_6$	1479.98	23.31	$F^{\#}_8/G^b_8$	5919.91	5.83
G_4	392.00	88.01	G_6	1567.98	22.00	G_8	6271.93	5.50
$G^{\#}_4/A^b_4$	415.30	83.07	$G^{\#}_6/A^b_6$	1661.22	20.77	$G^{\#}_8/A^b_8$	6644.88	5.19
A_4	440.00	78.41	A_6	1760.00	19.60	A_8	7040.00	4.90
$A^{\#}_4/B^b_4$	466.16	74.01	$A^{\#}_6/B^b_6$	1864.66	18.50	$A^{\#}_8/B^b_8$	7458.62	4.63
B_4	493.88	69.85	B_6	1975.53	17.46	B_8	7902.13	4.37

The subscript numbers on the right of the note letters indicates the level of octave in terms of musical terminology. In other words, these numbers give the positions on the piano keyboard. The reference note in this table is the note A and it is clearly visible that A has relatively smooth frequency values. Depending on the reference frequency value of the note A_4, the calculation of each frequency values is as shown below (3.1).

$$frequency = 440 \,.\, 2^{\frac{n}{12}} \\ n = \{\ldots, -21, -20, \ldots, 0, 1, \ldots 27, 28, \ldots\} \tag{3.1}$$

In the European music, the above information is as clear as it is explained. However, the Turkish music is rather more complex than the European music, because of having transpositional structure in terms of tonal chords. The term of harmony indicates a specific chord in the Turkish music. When the harmony changes, a sequence of tones transposes in terms of frequencies. The main harmony varieties are Bolahenk, Davud, Şah, Mansur, Kız, Yıldız, Süpürde [13]. Each type of harmony is a transposition of another. An example of transposition will be given below after defining the tone pitch concept of the Turkish music.



Harmony / Pitch	Bolahenk		Davud		Şah		Mansur		Kız		Yıldız		Süpürde	
	Freq (Hz)	Eur. Note	Freq (Hz)	Eur. Note	Freq (Hz)	Eur. Note	Freq (Hz)	Eur. Note	Freq (Hz)	Eur. Note	Freq (Hz)	Eur. Note	Freq (Hz)	Eur. Note
Rast	293.333	D_4	330	E_4	352	F_4	391.111	G_4	440	A_4	469.333	$A^{\#}_4/B^b_4$	521.481	C_5
Dügah	330	E_4	371.25	$F^{\#}_4/G^b_4$	396	G_4	440	A_4	495	B_4	528	C_5	586.667	D_5
Segah	366.667	$F^{\#}_4/G^b_4$	412.5	$G^{\#}_4/A^b_4$	440	A_4	488.889	B_4	550	$C^{\#}_5/D^b_5$	586.667	D_5	651.852	E_5
Çargah	391.111	G_4	440	A_4	469.333	$A^{\#}_4/B^b_4$	521.481	C_5	586.667	D_5	625.778	$D^{\#}_5/E^b_5$	695.309	F_5
Neva	440	A_4	495	B_4	528	C_5	586.667	D_5	660	E_5	704	F_5	782.222	G_5
Hüseyni	495	B_4	556.875	$C^{\#}_5/D^b_5$	594	D_5	660	E_5	742.5	$F^{\#}_5/G^b_5$	792	G_5	880	A_5
Eviç	550	$C^{\#}_5/D^b_5$	618.75	$D^{\#}_5/E^b_5$	660	E_5	733.333	$F^{\#}_5/G^b_5$	825	$G^{\#}_5/A^b_5$	880	A_5	977.778	B_5

Figure 3.1. Various frequency values and corresponding European notes of Turkish music pitches depending on harmony types

The transpositional structure makes it tough to define pitch names in terms of corresponding frequencies. Because it depends on the level of harmony type. Instead of representing each frequency by a specific letter or a specific note like Do, Re, Mi, etc. as in the European music, the Turkish music slices each harmony type into pitches and calls each pitch with a specific name i.e. Çargah, Neva, Hüseyni, Eviç, Rast, Dügah, Segah. In this manner, for example, whilst 440 Hz frequency -which reflects A4 tone in European music- is represented as pitch of Dügah in Mansur harmony, Rast pitch in Kız harmony has a frequency of 440 Hz too (Figure 3.1).

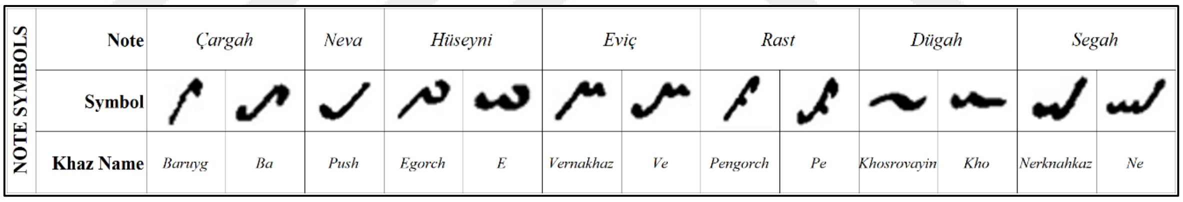
In the above summary (Figure 3.1), the European music notes and pitches of the Turkish music harmonies are mapped approximately. Giving exact matchings in terms of frequencies is not possible, however ignoring small and inconspicuous differences would be the solution to be able to translate these two systems in between.

Informations provided thus far would help to gather a clear understanding on the basics of the European and the Turkish sound systems, then getting familiar with the notes and their corresponding Hamparsum representations.

3.2. THE HAMPARSUM NOTATION

3.2.1. Main Symbols

The main body symbols of the Hamparsum notation has different shapes and specific names that were deployed from ancient Khaz notation symbol namings. In the following table (Figure 3.2), Khaz names of Hamparsum symbols and corresponding Turkish tone pitches are mentioned.



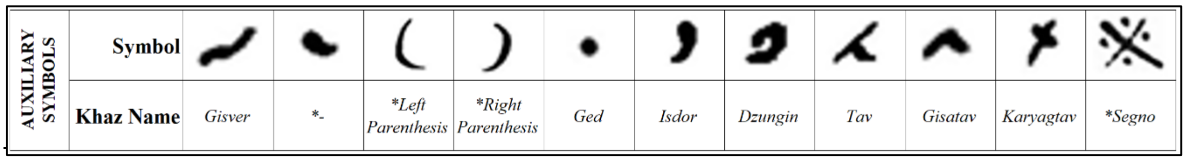
NOTE SYMBOLS														
	Note	*Çargah*		*Neva*	*Hüseyni*		*Eviç*		*Rast*		*Dügah*		*Segah*	
	Symbol													
	Khaz Name	*Baruyg*	*Ba*	*Push*	*Egorch*	*E*	*Vernakhaz*	*Ve*	*Pengorch*	*Pe*	*Khosrovayin*	*Kho*	*Nerknahkaz*	*Ne*

Figure 3.2. Khaz names of Hamparsum symbols and corresponding Turkish pitch names

The list of Khaz names were provided by an interview with the Rev. Fr. Dr. Krikor Damadian, Pastor of the Armenian Church of Christ The King in Kadiköy, Istanbul. Furthermore, cross-validation with the namings was carried out by a scientific journal publication by Karamahmutoğlu [14]. In case of ignoring the differences between the pronunciations, the only difference between these two sources is about the short namings. According to Damadian, the short names indicates the different forms of the symbols which generally represents a different octave value of the same note. However, Karamahmutoğlu inclines that the short names are only the abbreviations of the long names. The description of Damadian can differentiate between different symbol shapes of the same notes, so this study is based on this definition.

3.2.2. Auxiliary Symbols

Auxiliary symbols are used to indicate of a change in octave level, specify a time duration of a note or a rest, foreshadow an accidental tone like sharp or flat or give information about the change in the position to go in the flow of the composition, e.g. segno.



AUXILIARY SYMBOLS	Symbol											
	Khaz Name	*Gisver*	*-	**Left Parenthesis*	**Right Parenthesis*	*Ged*	*Isdor*	*Dzungin*	*Tav*	*Gisatav*	*Karyagtav*	**Segno*

Figure 3.3. Auxiliary Hamparsum symbols and their names in Khaz notation

Karamahmutoğlu did not emphasize the names of auxiliary symbols in her publication. On the other hand, Damadian shared their names verbally as he knew them during the interview. The above table shows the list of auxiliary symbols that have been taken into consideration in this study and their corresponding Khaz names beneath (Figure 3.3).

3.2.3. Notes

A Hamparsum note consists of at least a main note symbol. It might have a combination with one or more auxiliary symbols. One of the purposes to combine with auxiliaries is to differantiate the tone from default tone of the main symbol. Another reason to do that is to indicate the duration of current note or the duration together with neighbor notes. The last aim of using auxiliaries is to change the octave of the same note. A sample of a composition that was written in Hamparsum notation is as shown in Figure 3.4.

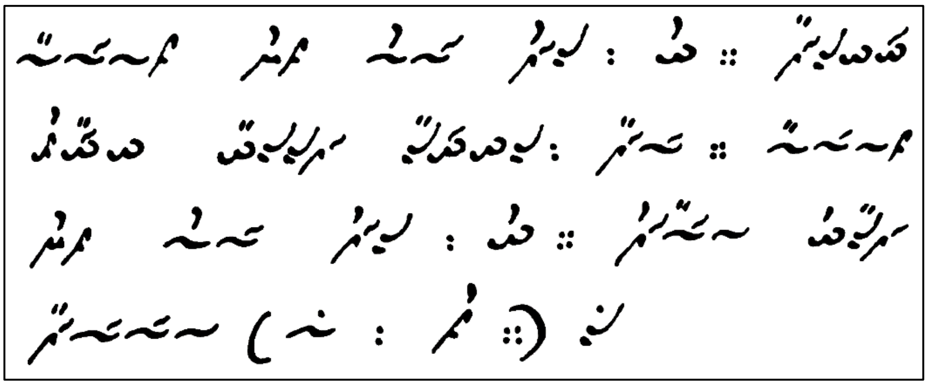


Figure 3.4. A sample Hamparsum score

As it is compared with European music in “3.1 Musical Background” section, Turkish music has a set of harmonies and sequences of the note frequencies that differ depending on the type of harmony. In order to overcome this complexity, Mansur harmony has been taken into consideration in figure (Figure 3.5) to be able to express the Hamparsum notes in both European stave presentation and pitch names of the Turkish music.

3.2.4. Durations, Rests and Bars

To symbolize the amount of time unit passed during a note is playing or during a rest symbol is acting, some of the auxiliary symbols must be utilized. In both the Turkish music and the European music, the durations have represented by using a numeric ratios, like 4/4, 16/4, 2/4, etc. These fractional numbers do not indicate time units in terms of seconds; they only give relative durations which might imply different seconds values based on the tempo frequency value of the song. During the flow of the song, an instrument player or a singer would know the approximate time will be past on a specific note. The time duration and rest indicators are shown by using the same symbols in Hamparsum notation. In European notation, durations and rests are represented with different symbols. A sum up list is shared in the table below (Figure 3.6).

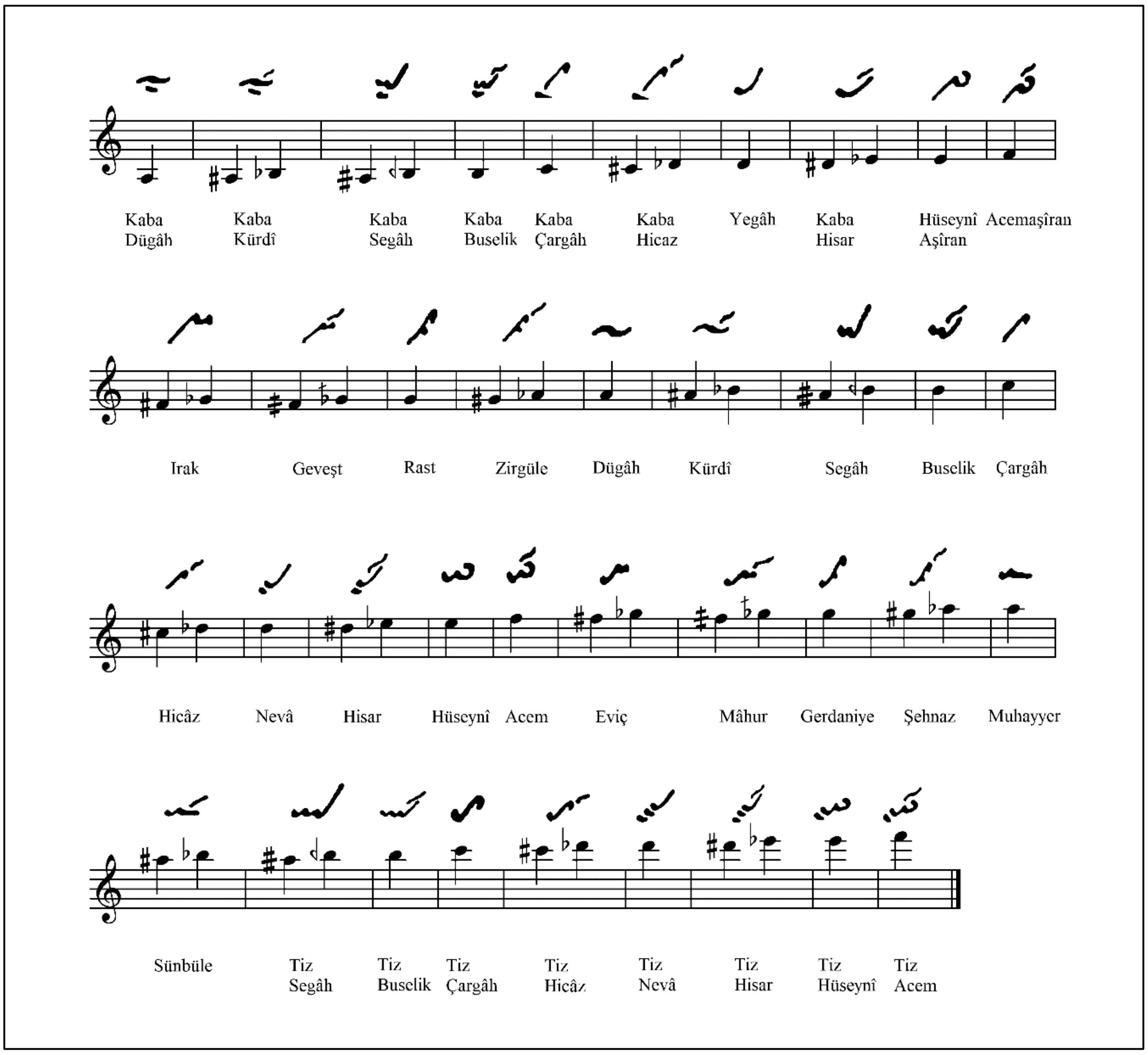
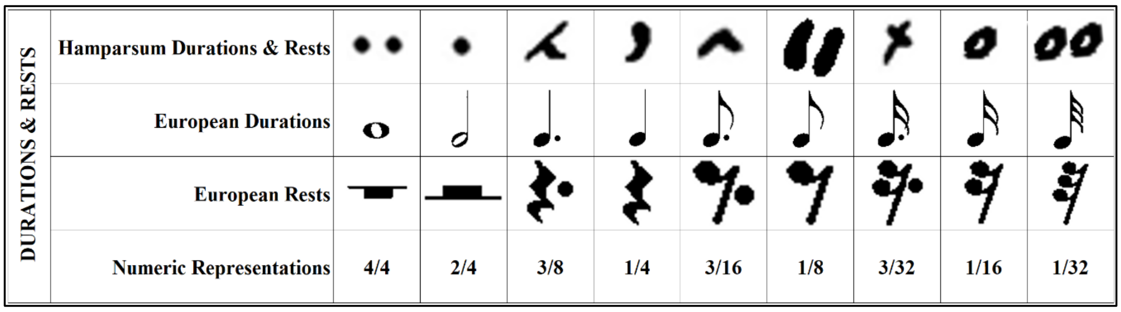


Figure 3.5. Hamparsum notes and corresponding European notes and Turkish pitches



DURATIONS & RESTS									
Hamparsum Durations & Rests	• •	•							
European Durations	𝅝	𝅗𝅥	♩.	♩	♪.	♪	𝅘𝅥𝅯.	𝅘𝅥𝅯	𝅘𝅥𝅰
European Rests	𝄻	𝄼	𝄽.	𝄽	𝄾.	𝄾	𝄿.	𝄿	𝅀
Numeric Representations	4/4	2/4	3/8	1/4	3/16	1/8	3/32	1/16	1/32

Figure 3.6. Hamparsum duration and rest symbols

The bar is the indicator that the end of a composition part, such as end of a measure, end of a line or end of the whole score (Figure 3.7). The parenthesis in Hamparsum notation and the corresponding European symbols -that is used to show two partitions with similarities but with small nuances with a numeric order to play- are also visible below figure.

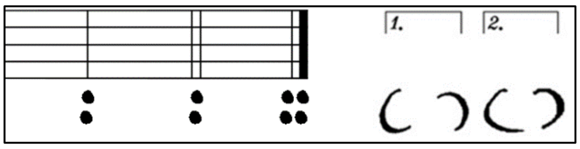


Figure 3.7. Hamparsum bar symbols and corresponding European notation bars

3.2.5. Exclusive Rules

Many of the rules has already been explained above subsections. Nevertheless, there are some more rules worth to emphasize. The Hamparsum notation has the direction of writing left to right and top to bottom, which is similar with the European notation. However the most important point of separation between these two notation systems is that the European notes are written on a stave which consists of 5 horizontal and smooth lines, and the Hamparsum notes are written without any lines or stave.

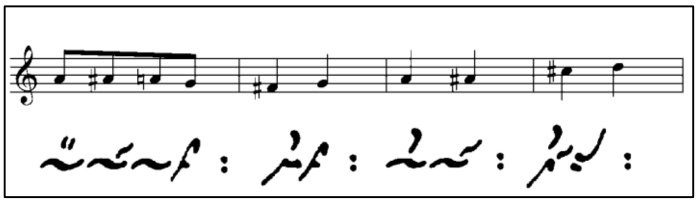


Figure 3.8. An example of translation between European and Hamparsum notations

Writing in Hamparsum notation is like writing sentences in an arbitrary human language. Hamparsum notation has not a cursive style, i.e. all the main symbols and auxiliaries must be written as separately, they should not touch each other. In case of a connectivity between different notes or symbols, that means a mistake has been made by the author.

Moreover, the most important nuance of this notation system is about the usage of the duration indicators. If a note symbols has a duration symbol on top of itself that means the valid duration is emphasized by itself. However, if a specific note symbol does not have any duration auxiliary on the top, it should inherit the duration of the left adjacent note. If the note on the left also does not have a time duration too, the first note that has a time duration symbol should be searched to the left. On the above figure (Figure 3.8), the leftmost note is Dügah with a duration of ⅛. However next respectively Kürdi, Dügah, Rast notes do not have a time duration indicator, so they are also accepted as ⅛. When the next group has started, a new Irak note with a ¼ time duration comes and makes the adjacent Rast ¼.

4. METHODOLOGY

The data set used for the training and the testing stages of this study is rather noisy; because the original pieces are antique and the delivered digital copies had not been scanned with high quality. Therefore, image pre-processing becomes more sensitive in terms of both removal of irrelevant elements and preserving relevant components. Furthermore, a utilized sort of thresholding method is explained for the purpose of segmentation. The border following method as a contour detector, and a seed fill algorithm are utilized to locate each symbols accurately and clear their bounding boxes by removing irrelevant components that are overflowed. Feature extraction process which generates the most important information is achieved by using 2D Gabor filter bank. As far as having accurate features for each component, Support Vector Machine classifier would be employed to be able to recognize symbols by the help of their structural features. After gathering the identities of each individual symbol, binary dilation operation would be useful to give the related symbols a meaning in terms of their musical representation. In this section, utilized methods and techniques for the recognition of handwritten Hamparsum music scores are explained in detail.

4.1. COLOR REPRESENTATION MODELS

For humans and all other animals with trichromatic vision, colors are one of the most descriptive components among objects around. A generic RGB representation of a color image requires the amounts of red, green and blue attributes for each pixels. Red, green and blue are the primary colors in terms of light absorption; so that each color would be produced by a mixture of these. RGB color model is suitable for the representation of colors in electronic systems such as televisions and computer monitors etc. Usually, RGB model is indicated by a cube (Figure 4.1).

Hue, Saturation and Intensity (HSI) is the transformation of RGB cube into a conical between black and white vertices (Figure 4.1). HSI represents the colors in the same way as the human eye. Hue component defines the exact color from 0 to 360 degrees. Saturation refers to the density of the color; can be defined as the amout of dilution with white. Intensity is the brightness of the color, in other words the average grey level and may represented by the grayscale diagonal between black and white vertices of RGB cube.

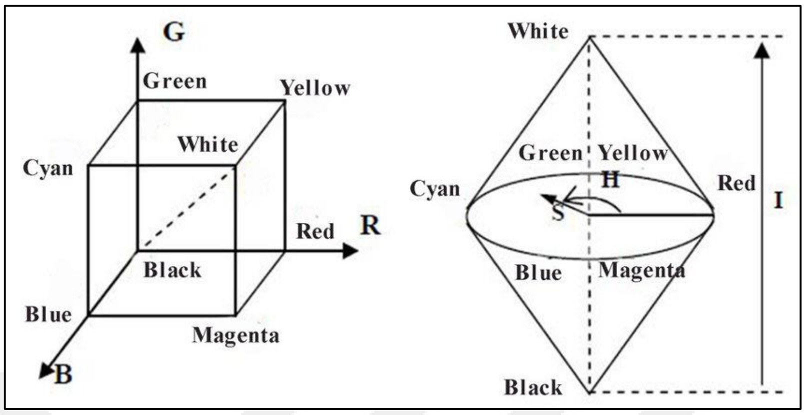


Figure 4.1. RGB and HSI color model representations

4.1.1. Intensity Layer Extraction

The formula of RGB to HSI transformation is as shown below (4.1). First of all RGB colors are converted to YC_1C_2. After that the intensity would be extracted by variable Y which is calculated by the mean of red, green and blue values [15].

$$\begin{bmatrix} Y \\ C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1 & -1/2 & -1/2 \\ 0 & -\sqrt{3}/2 & \sqrt{3}/2 \end{bmatrix} . \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

- $I = Y$ ▪ $S = \sqrt{C_1^2 + C_2^2}$ (4.1)
- $if\ C_2 \geq 0\ then\ H = cos^{-1}(\frac{C_2}{S})$

  $else \quad H = 2\pi - cos^{-1}(\frac{C_2}{S})$

4.2. THRESHOLDING

In this study, thresholding is applied for image segmentation process which is vital for the next feature extraction step. There are many techniques proposed for image segmentation, however thresholding is rather lean approach and it is much effective at the same time.

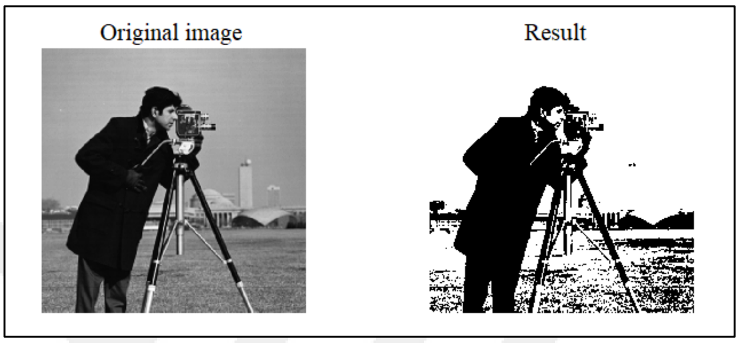


Figure 4.2. Input and output of threshold process

Different thresholding methods are widely used in the area of pattern recognition. However the basic (simple) thresholding approach is preferred in this study [16]. The method would be described as the replacement of each pixel with pure black color value if the intensity is less than a constant threshold value; else the intensity is greater than the constant, in that case the replacement would be done with pure white color value (Figure 4.2).

4.3. BORDER FOLLOWING METHOD

Due to the fact that there are no other pixel colors different than black and white, the most significant features in terms of detecting the contours of shapes included in the image are binarized pixel values and the values of adjacent pixels. In this study, a border following method is employed which was described by Suzuki and Abe [17] to achieve contour detection of each symbol precisely. Suzuki method offers two different approaches on border following purpose. The first approach detects outer borders of each components and the holes surrounded by an outer component. Unlike, the second approach describes an algorithm to detect only outer borders of each components. For the recognition of

Hamparsum symbols, the second approach developed by Suzuki and Abe provides an simply efficient and accordingly fast solution.

Describing the second approach of Suzuki and Abe requires to explain some definitions and assumptions. A 1-component is a shape that consists of only pixels of value 1 (1-pixels); and similarly a connected 0-pixels forms a 0-component. In two dimensional (2D) image representation, connections of a pixel with adjacents are defined by pixel connectivity, such as 4-connected (Von Neumann) neighborhood and 8-connected (Moore) neighborhood (Figure 4.3).

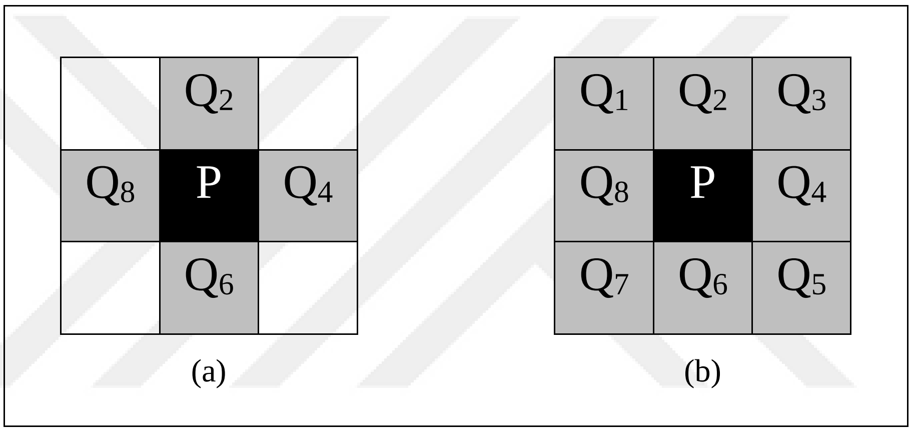


Figure 4.3. (a) Von Neumann neighbors and (b) Moore neighbors of pixel P

4-connected adjacent defines such a pixel Q which is a neighbor of pixel P in case of sharing at least an edge. On the other hand, 8-connected neighborhood covers not only 4-connected adjacency feature, but also the vertex connections (Figure 4.3). The main idea behind the scene is connected 1-pixels represents a 1-component and similarly adjacent 0-pixels represents a 0-component. In other words, the components are the connected pixels of the same value in a binarized image. Furthermore, if black pixels are represented by 4-connected neighborhood, that automatically means white pixels are in 8-connected form, or vice versa. To be able to explain the method, black pixels are considered as having density values of 1 and whites (background) are accepted as having density values of 0.

A border point (i, j) is described as a point between a 1-component S_1 and 0-component S_2; and S_2 represents the background. Assume that (i, j) is a member of S_1 and (p, q) is a member of S_2 which is connected to (i, j). Whole set of borders between S_1 and S_2

indicates the outer border for S_1 component. Besides, it can be said that S_2 surrounds S_1. Moreover, the frame of the image is the parent border of an outer border between S_1 and S_2.

The border following algorithm applies a raster scan on each pixel of the image to extract the borders and the surroundness among the components. The direction of raster scan would be described as left to right and top to bottom. When a pixel (i, j) is found which satisfies the conditions of a border point is assumed as the starting point of a border and assign a sequential number on that point to differentiate a newly found border. During the scan, each connected border points are assigned to an identifier numbers. Raster scan ends at the lowest rightmost pixel of the frame. In the end, member pixels of each border, border starting pixels and rightmost border line-ending pixels would be marked with separate numeric values, which means the contours of each symbol in the image are successfully identified.

4.4. SEED FILL ALGORITHM

In case of contour detection is achieved, that means bounding boxes of each symbol have already been detected. In terms of Hamparsum notation point of view, bounding boxes of symbols and note characters are transitive and the possibility for existance of intersections among near components is significantly high. To be able to overcome the intersection among bounding boxes issue defined above, employing a segmentation operation is a necessity. Heckbert had been developed an algorithm [18] to detect individual segments in an image and to fill each segment pixels with an identifier numeric value.

Algorithm 4.1. A seed fill algorithm [18]

```
   fill: set the pixel at (x, y) and all of its 4-connected neighbors
   with the same pixel value to the new pixel value nv.
   A 4-connected neighbor is a pixel above, below, left, or right of a pixel.
Pixel: type ← int;
Window: type ← record [xmin, ymin, xmax, ymax: int];      inclusive window

procedure fill(
   x, y: int;                                    seed point
   nv: int;                                      new pixel value
   win: Window;                                  screen window
   pixelread: function(x, y: int): Pixel;        procedure for reading pixels
   pixelwrite: procedure(x, y: int; pv: Pixel);  procedure for writing pixels
   );

start, xl, x2, dy: int;
ov: Pixel;                                       old pixel value

Segment: type ← record [y, xl, xr, dy: int];
   Filled horizontal segment of scanline y for xl ≤ x ≤ xr.
   Parent segment was on line y – dy. dy = 1 or –1

max: const int ← 10000;                          max depth of stack
stack: array[0..max – 1] of Segment;             stack of filled segments
sp: int ← 0;                                     stack pointer

procedure push(y, xl, xr, dy: int);              push new segment on stack
begin
   if sp < max and y + dy ≥ win.ymin and y + dy ≤ win.ymax then begin
      stack[sp].y ← y;
      stack[sp].xl ← xl;
      stack[sp].xr ← xr;
      stack[sp].dy ← dy;
      sp ← sp + l;
   end;
endproc push;

procedure pop(y, xl, xr, dy: ref int);           pop segment off stack
begin
   sp ← sp – 1;
   dy ← stack[sp].dy;
   y ← stack[sp].y + dy;
   xl ← stack[sp].xl;
   xr ← stack[sp].xr;
endproc pop;
```

```
**begin** $procedure\ fill$
  $ov \leftarrow pixelread(x, y);$                               *read pixel value at seed point*
  **if** $ov = nv$ **or** $x < win.xmin$ **or** $x > win.xmax$
      **or** $y < win.ymin$ **or** $y > win.ymax$ **then**
    **return**;
  $push(y, x, x, 1);$                                             *needed in some cases*
  $push(y + 1, x, x, -1);$                                        *seed segment (popped 1st)*

  **while** $sp > 0$ **do**
    *pop segment off stack and fill a neighboring scan line*
    $pop(y, x1, x2, dy);$
    *segment of scan line* $y - dy$ *for* $x1 \leq x \leq x2$ *was previously filled,*
    *now explore adjacent pixels in scan line* $y$
    $x \leftarrow x1;$
    **while** $x \geq win.xmin$ **and** $pixelread(x, y) = ov$ **do**
      $pixelwrite(x, y, nv);$
      $x \leftarrow x - 1;$
    **endloop**;

    **if** $x \geq xl$ **then goto** $skip;$
    $start \leftarrow x + 1;$
    **if** $start < xl$ **then** $push(y, start, x1 - 1, -dy);$          *leak on left?*
    $x \leftarrow xl + 1;$
    **loop do**
      **while** $x \leq win.xmax$ **and** $pixelread(x, y) = ov$ **do**
        $pixelwrite(x, y, nv);$
        $x \leftarrow x + 1;$
      **endloop**;
      $push(y, start, x - 1, dy);$
      **if** $x > x2 + 1$ **then** $push(y, x2 + 1, x - 1, -dy);$       *leak on right?*
      $skip: x \leftarrow x + 1;$
      **while** $x \leq x2$ **and** $pixelread(x, y) \neq ov$ **do**
        $x \leftarrow x + 1;$
      **endloop**;
      $start \leftarrow x;$
    **while** $x \leq x2;$
  **endloop**;
**endproc** $fill;$
```

4.5. 2D GABOR FILTER

A two dimensional (2D) Gabor filter is a linear filter used in particular in the field of texture analysis to detect the frequency of content in certain directions [19]. In other words, the image content extending to a certain direction would be detected by the help of a 2D Gabor filter. A 2D Gabor filter is as shown in the following equations (4.2).

$$g_{\sigma,\gamma,\theta,\lambda,\varphi}(x,y) = e^{-\frac{x'^2+\gamma^2 y'^2}{2\sigma^2}} e^{i\left(2\pi\frac{x'}{\lambda}+\varphi\right)}$$
$$x' = xcos\theta + ysin\theta \tag{4.2}$$
$$y' = -xsin\theta + ycos\theta$$

x and y denote a specific position in the n by n sized kernel matrix that is obtained by assigning values to both x and y in $\left[-\left(\frac{n-1}{2}\right),\left(\frac{n-1}{2}\right)\right]$ closed range. x' and y' values are formulated by the transformation below (4.3).

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{4.3}$$

θ is the orientation which determines the direction of the content to be detected. In case the θ value is 0^o, that means the content which is perpendicular to the x axis is detected. As θ is increased, the content with increasing angle is determined according to the orientation of the matrix. σ is the standard deviation of the Gaussian envelope and λ is the wavelength of the sinusoidal factor. The Gaussian function creates a bell curved graph and γ is the spatial aspect ratio of that curve. φ is the phase offset and when this value is 0^o or 180^o, the Gaussian curve is located in the center of x axis.

The g function returns a complex number. Furthermore, the real and the imaginary components would be expressed as follows (4.4) [20].

$$g_{\sigma,\gamma,\theta,\lambda,\varphi}(x,y) = e^{-\frac{x'^2+\gamma^2 y'^2}{2\sigma^2}} \cos\left(2\pi\frac{x'}{\lambda}+\varphi\right) \qquad Real$$
$$g_{\sigma,\gamma,\theta,\lambda,\varphi}(x,y) = e^{-\frac{x'^2+\gamma^2 y'^2}{2\sigma^2}} \sin\left(2\pi\frac{x'}{\lambda}+\varphi\right) \qquad Imaginary \tag{4.4}$$

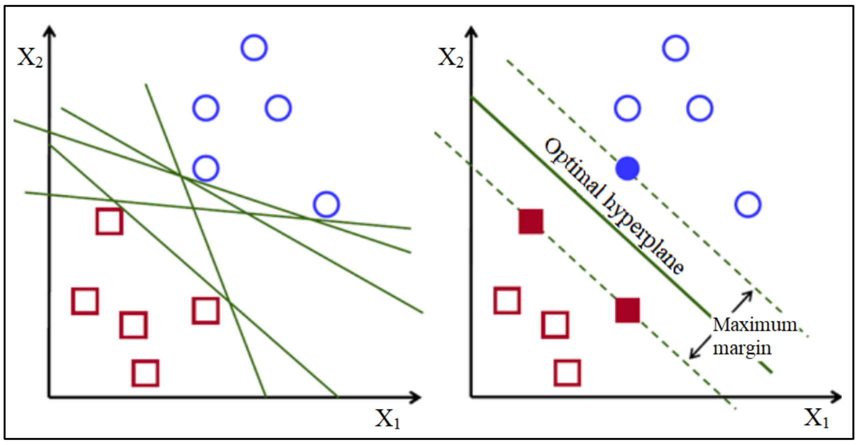
The return value of the Gabor function would be obtained by taking the square root of the sum of the squares of the return values of both the real and imaginary functions above. Nevertheless, skipping imaginary g function and taking only the real part into consideration is usually satisfactory.

To sum up, the Gabor filter analyzes the structures of given components and depending on that generates structural inferences, which would be useful to be able to classify the observations made for a set of components.

4.6. SVM CLASSIFIER

Support Vector Machine (SVM) is a supervised learning method with associated learning algorithms that analyze data and recognize patterns, which is often used for classification problems and regression analysis [21]. SVM is one of the most useful classification models to separate different observations. Classification process would be performed by finding the hyperplane that best segregates two classes. The main idea is to define a set of rules and to classify the population as the most likely segments according to the maximal margin principle. Maximal margin representation on a scatter plot might be defined as the distance of the optimal separating hyperplane to both training observations (Figure 4.4).

The real world problems might not be solved always in a linear way. Normally, the dimension size of an SVM classifier is defined as equal to the number of features obtained in prior. Moreover, adding more dimensions to be able to separate handled segments in an optimal way by utilizing some transformation techniques is rather possible. SVM provides more robustness and accuracy with this flexibility characteristic which is called kernel trick (Figure 4.5).



(a) (b)

Figure 4.4. (a) Infinite hyperplane space, (b) optimal hyperplane with maximal margin

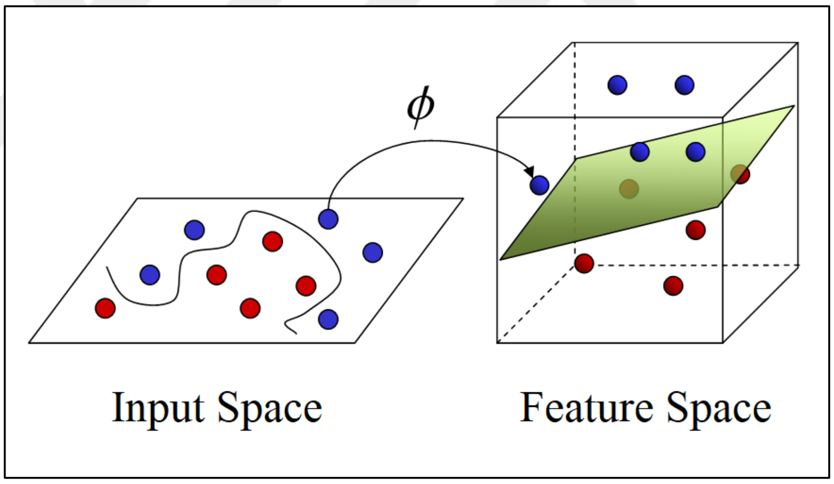


Figure 4.5. An example for kernel trick

In this study, an SVM classifier is employed to be able to achieve the segmentation of Hamparsum notes by using a set of 2D Gabor filters that generate a number of structural features for each component.

4.7. BINARY DILATION

Following the extraction of individual symbols and notes, the corresponding time indicators, the accidental markers and the octave differentiator signs should be associated with the related note symbols to identify what exactly that note represents to. In Hamparsum notation, these relatively small indicators are located at the top or bottom of the note character and neither are connected to the actual body nor too distant. To be able to connect these related symbols which are positioned top to bottom, a kind of vertical stretching method would be a solution.

Dilation is the name of a morphological transformation operator which expands the foreground pixels and it may shrink inner holes of these components if exists [22]. In detail, dilation is a non-complex method that takes only two inputs; an image to be dilated and a structuring element (dilation kernel) to define how to apply dilation in terms of direction.

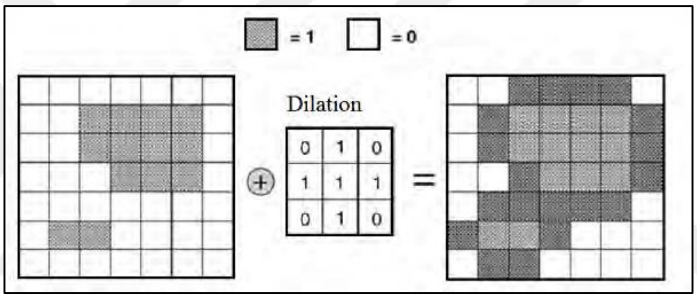


Figure 4.6. An example for dilation operation

As an explanation of how to apply dilation operation, at first the center point of the kernel would be superimposition on the pixel of the currently operating image, and then the adjacent pixel values which were defined in the dilation kernel in terms of the distance from the center point would be overwritten with the kernel pixel values. An example for this operation is as shown above (Figure 4.6). The most commonly used dimension size of the dilation kernel is 3x3 as it is presented, however this does not mean that this is a strictly defined rule. The kernel size and the values of kernel pixels are able to be modified according to the requirements and expected results. A larger kernel would show more dilation effect or vice versa. As a summary, binary dilation is rather simple, but effective way of applying mathematical morphology to the binary images.

4.8. CROSS VALIDATION

There are two main categories of cross validation (CV) techniques: non-exhaustive cross validation and exhaustive cross validation. Major non-exhaustive methods are might be called as Holdout cross validation, K-fold cross validation and Stratified K-fold cross validation. The most commonly used exhaustive cross validation method is Leave p-out cross validation [23].

In general cross validation attitude, the data set is divided into two separated sets called training set and testing set. In Holdout CV, the training set is divided into another small set which is used for validation. The validation set is different from testing set, but it is a part of original training set. However the problem is how to choose this small validation set, which might end up with high variance. Different sets would give different results. To avoid that, K-fold CV method might be used. K-fold cross validation is dividing the training set into k subsets, which means the hold out method is repeated k times. Each time, a distinct subset is chosen as validation set, and the training set comprises the remaining k-1 subsets. The fold error rates are averaged as a result. In case of having a bias in the training data set, it is not as easy to select a validation set randomly. To avoid that kind of variance, Stratified K-fold CV might be applied. In this method, an arrangement is made for making sure that there are equally likely number of results of all categories in addition to K-fold method.

Leave p-out CV method might be described as, if there are n data points, p of them are used for validation and n-p of them are taken for the training in each iteration. This loop goes on for all possible combinations of p from original data set. The average error rates of all the iterations gives the final success rate. The p value might be chosen according to the suitability for the data set, however, usually higher p value provides more exhaustive validation.

5. ANALYSIS AND DESIGN

In this chapter, the data set used in this study is introduced briefly. After that, the analysis of the data and the design of the Hamparsum OMR system will be presented in detail. (Figure 5.1) shows the general schema for the Hamparsum OMR system.

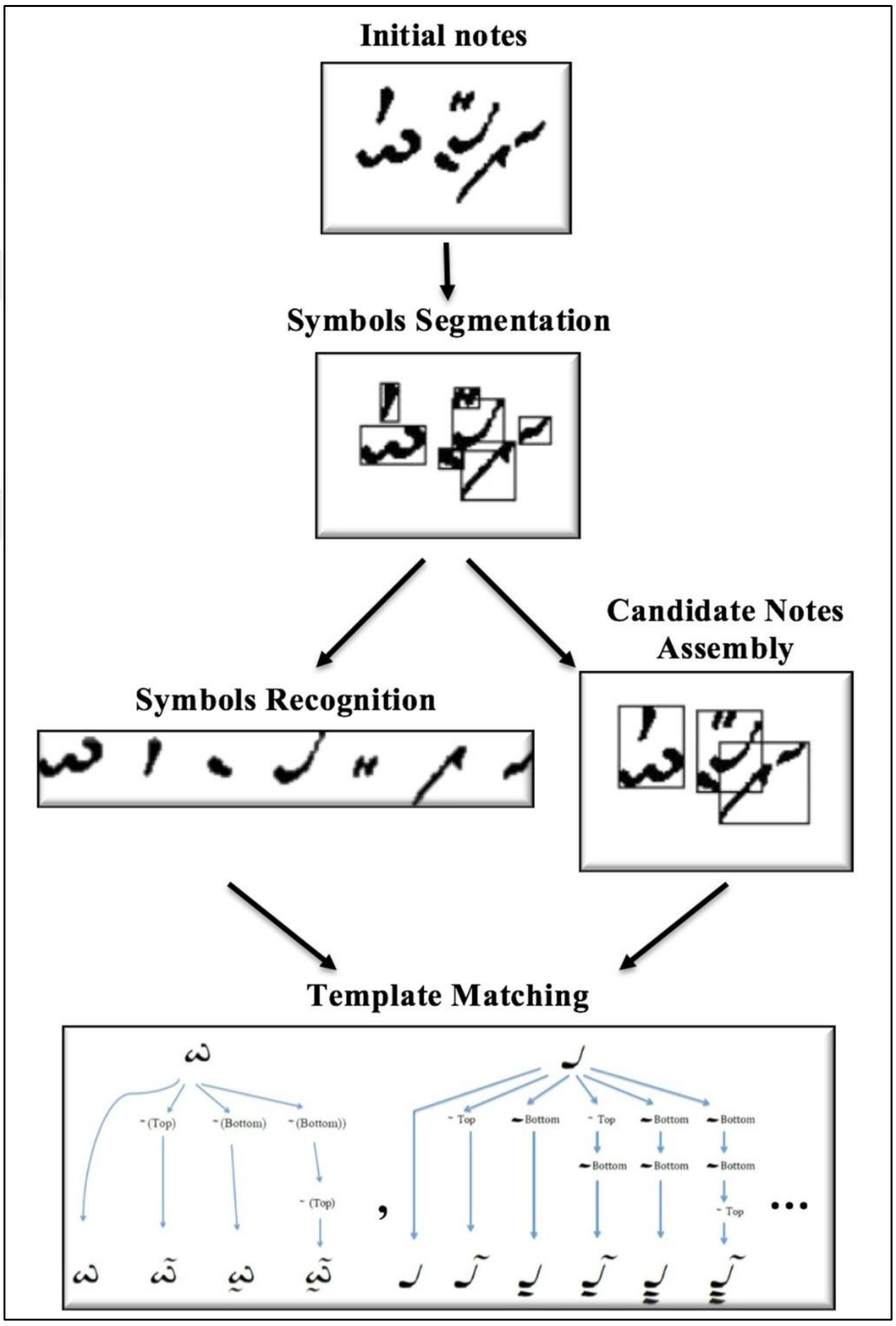


Figure 5.1. General schema for the Hamparsum OMR system

5.1. THE HAMPARSUM DATA SET

In this study, a set of manuscript Hamparsum notation images was used, which has been provided in the form of an already scanned digital copies from Turkish Radio and Television (TRT) Corporation. The database contains ancient and original Hamparsum manuscripts written by different authors. During this study, TRT archives were transferred to Ottoman Archives Department in İstanbul, therefore the owner of the data is Ottoman Archives at the moment. The digital copies has not been scanned very high quality, and there was a brand name printed on the background of each pages with transparent red color. The brand name was “Pecya” which is the name of an old online library and archiving system. Pecya was not active when this study kicks off, however probably their databases must have been transferred to TRT in advance. During this journey of the data set, the quality of the resolution might have been reduced couple of times, it is not possible to be sure about it at the moment.

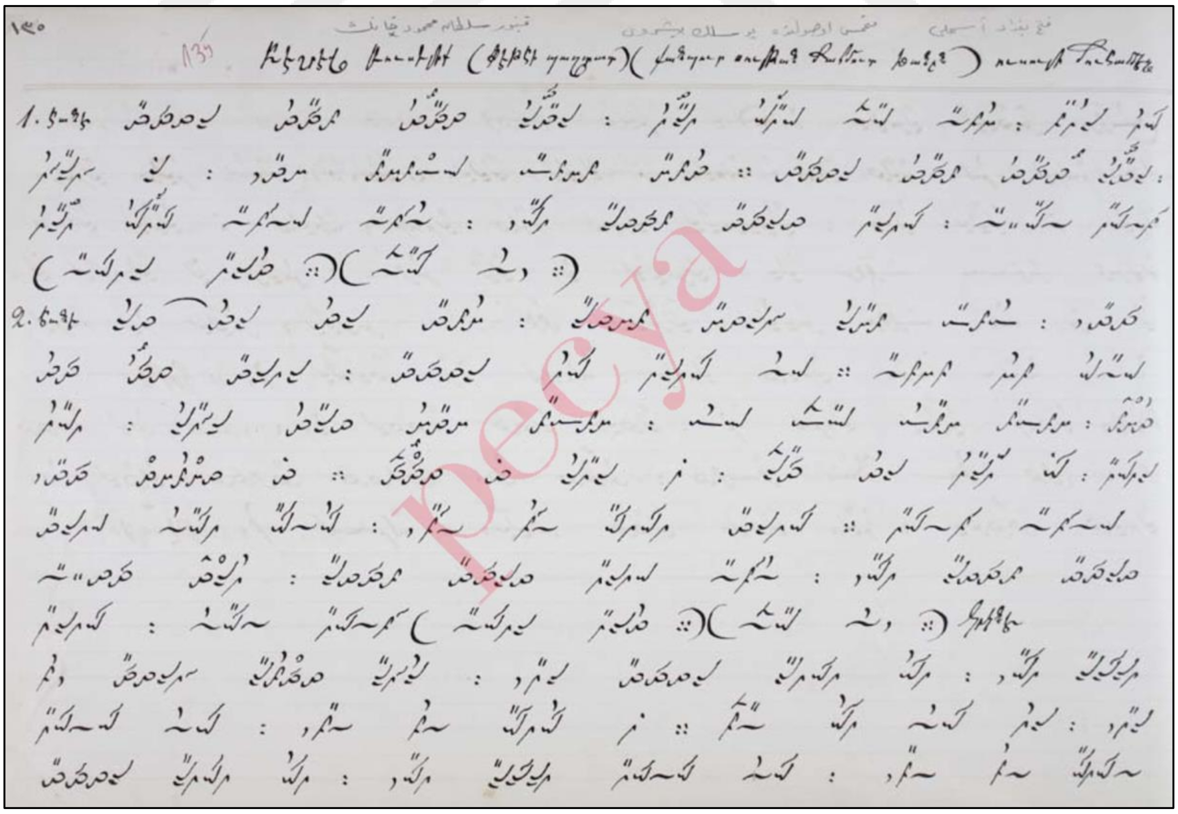


Figure 5.2. A sample Hamparsum notation score

The pages of the data set does not involve only Hamparsum symbols, there are also some irrelevant postscripts to be ignored that had been taken in Armenian and Ottoman alphabets. Against all odds, writings are readable by human eye and the scores have already been clasified by academicians in terms of titles of the songs. The low image quality and the existance of irrelevant postscripts impediments require more preprocessing operation. On the other hand, while the names of the songs are known, it is possible to validate the results of the recognition. In summary, the content of the data set matches the purpose of this study. It contains good and clear samples to train and test. It is not easy to gather a Hamparsum data set of this content and size. For this reason, it was decided to go on the study with this data set.

5.2. IMAGE PRE-PROCESSING

Since the data are rather noisy and low quality and there are too many scribbles and lyrics in various languages unrelated with Hamparsum notation, a pre-processing prior to the recognition needs to be performed. Although the sample Hamparsum score above (Figure 5.2) is relatively clear compared to the whole data set, the segments that are not in Hamparsum notation would be noticed. Above all, the “pecya” tag stands out in very large fonts on the diagonal axis. The scripts written in Ottoman and Armenian letters appear at top of the score and at the beginning or end of each paragraph. A number of signs and inscriptions written during the classification can be seen in various sections of the image.

The above-mentioned segments are not related to the subject of this study, but might constitute a disparate study topic. Therefore, in part, these sections will be manually removed until the picture of almost the pure Hamparsum symbols is provided. Nevertheless, removing the pecya tag without any morphological process does not seem possible. Due to the fact that, a binarization process following a grayscale transformation will be utilized to obtain a rather clean data required for the recognition.

5.2.1. Grayscale Transformation

Although the original data set consists of color images, the variety of color values is not very high. However the most significant features to differentiate the “pecya” tag are its red color and transparency. Although the tag rides over the Hamparsum note symbols, the likelihood of being able to extract it from whole image by a following binarization threshold process seems high because of the difference on intensity, when the grayscale transform is obtained.

In detail, the intensity layer extraction method is employed to convert the image to a grayscale representation. The method is based on color model transformation from RGB to HSI. In fact, hue and saturation layers are not required to obtain a grayscale form. For example, in the 3D RGB cubic representation, the grayscale line is between the black and white color vertices and in the HSI representation this line fits in the intensity axis. Extracting only the intensity layer would be sufficient to be able to acquire the image only in the gray colors variance.

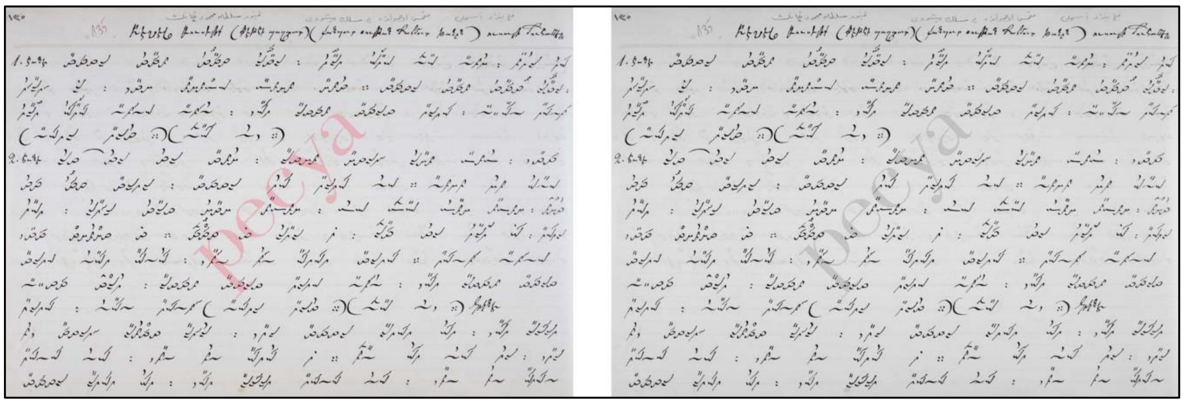


Figure 5.3. Grayscale transformation of a Hamparsum score

Figure 5.3 shows of a sample input and its output for the grayscale transformation. As it is clear, whilst the text symbols look near to black, the “pecya” tag and some irrelevant scribbles are almost converging to background.

5.2.2. Extraction of Hamparsum Letters

As far as the grayscale image is obtained, to remove noises and some irrelevant sections, simply the selected method is to apply a binary threshold mechanism for the extraction of Hamparsum letters. However the threshold value may differ according to the content of the input image. Therefore the threshold parameter might be a dynamic variable. In detail, the threshold space is between 0 and 255 in grayscale representation where 0 is black and 255 is white, the pixel values lower than the threshold parameter would be updated with 0 value namely black color and the pixel values higher than the threshold parameter would be updated with 1 value namely white color.

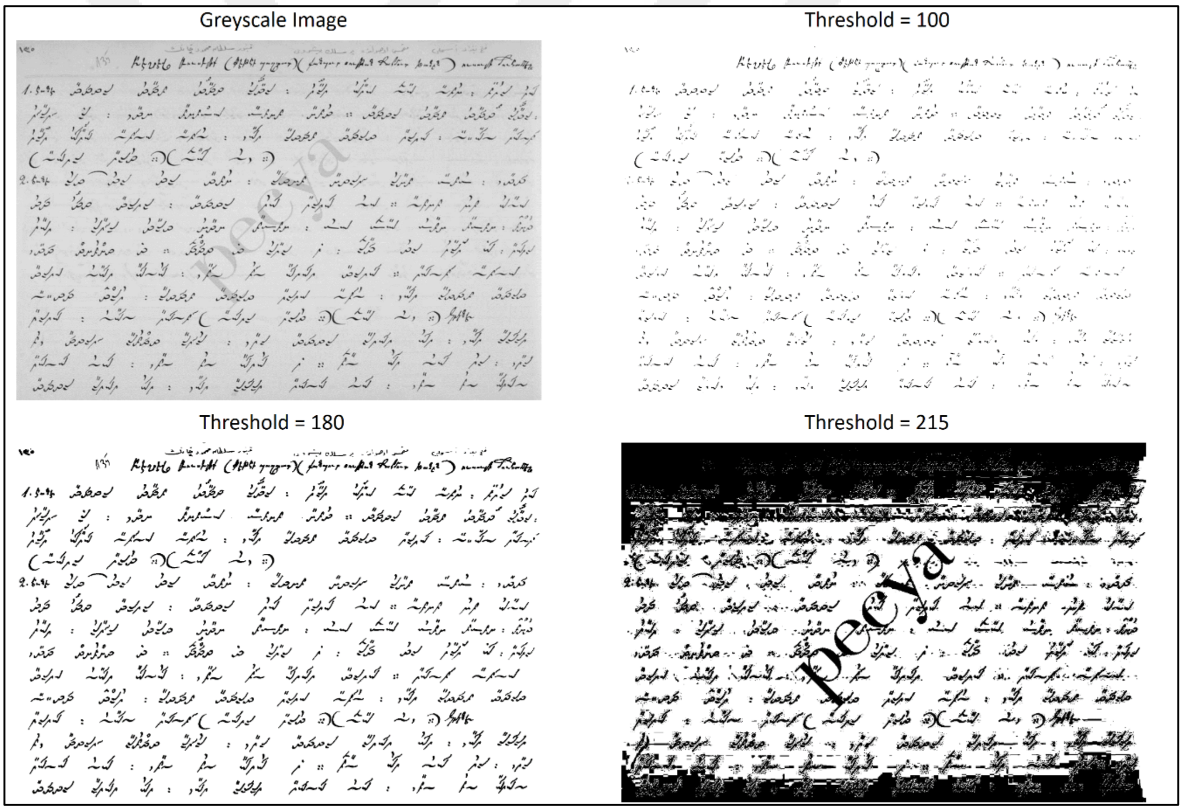


Figure 5.4. Binarization with various values of threshold parameters

Figure 5.4 shows outputs of a binarization process for various threshold values. For the sample above, threshold value of 100 removes too many information including Hamparsum symbols, 215 does not sufficient to disappear “pecya” tag and creates some dark sections at the top and the bottom of the image. In the best case, threshold value of

180 yields better, however notice that this does not mean 180 is the generic threshold value for whole data set. It depends on the pixel values of each component in the grayscale image. Therefore, threshold parameter must be set as adjustable in the implementation.

5.2.3. Removal of Remaining Redundant Sections

At the end of binarization, there still exists some irrelevant sections in the output. Usually, these sections consist of postscripts or head titles about the musical piece in the score or they might include some notes that were priorly taken during the classification of the notes. This study does not concern about the sections other than Hamparsum symbols, so that these remaining redundant sections are removed by manual.

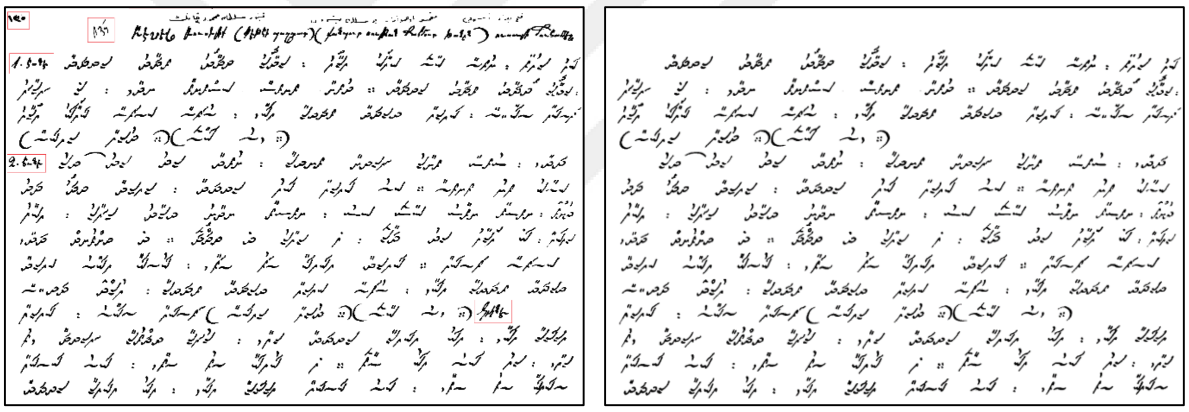

(a) (b)

Figure 5.5. Manual removing of irrelevant sections; (a) before removal (b) after removal

5.3. SYMBOL SEGMENTATION

The remaining data after pre-processing contain only Hamparsum symbols. However each individual components should be isolated and labeled separately. Therefore the contours of each non-connecting symbols or auxiliaries, and hence their bounding boxes will be detected by utilizing a border following method [17]. Although symbols do not touch each other, because of the author’s italic handwriting style, bounding boxes might intersect. Even after the bounding box detection, there still might be multiple separated components into the same frame. A seed fill algorithm [18] is utilized to be able to detect and label each components into the frame. In this manner mapping the component that has the highest area within the boundaries and remove the outliers will be achieved.

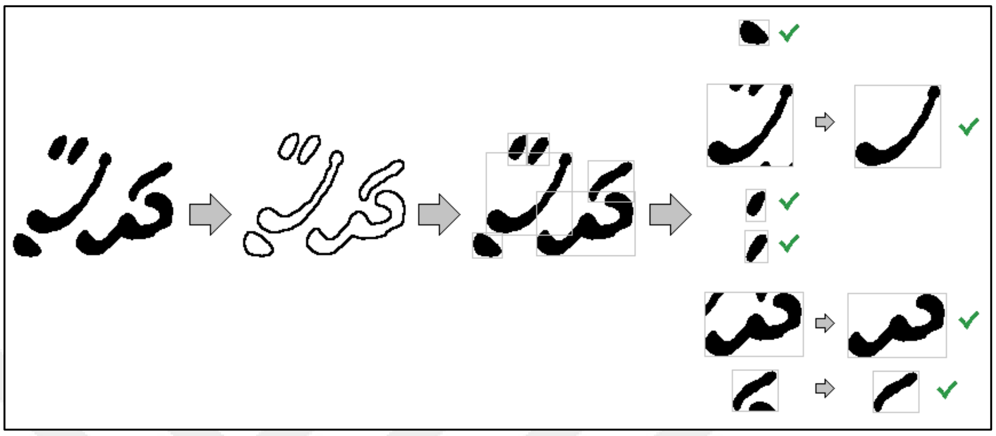


Figure 5.6. Segmentation of Hamparsum symbols

Figure 5.6 shows the segmentation process in a nutshell. First of all each individual shape was emphasized with a surrounding frame which represents the bounding boxes. Second, each frame was isolated from the image and checked for the sections in order to decide they belong to a single or multiple symbols. In case there is only one component in the boundaries, that means the segmentation of that frame is achieved. However, if there are multiple segments, the seed fill algorithm labels each components pixels with an identifier number. Every single pixel of the same component will have been labeled with the same value at the end of the operation. In this manner, it is possible to detect the dominant segment in the frame by taking the number of pixels of each component and selecting the maximum.

5.4. FEATURE EXTRACTION

As well as the symbols are isolated from the image, each one of them should be analyzed in order to obtain optimum amount of information to identify them by utilizing their structural inferences. A bank of 2D Gabor filters was employed to gather the necessary knowledge for the recognition. In this study, only real part of the 2D Gabor filters has been employed. Depending on the experiments, 11 parameter sets were defined corresponding to $s = 11$ different scales of the 2D Gabor filter by modifying the φ, σ, and λ values with an incremental factor. Then for each scale parameter set, $o = 13$ different orientations were defined by modifying the θ argument with an incremental factor, thus corresponding to different angles. The parameter sets define 143 Gabor filters $G(s, o)$ with different scale and orientation values that constitute the 2D Gabor filter bank. The number of filters were specified via testing different sizes of 2D Gabor filter banks having as a criterion the success rate of the supervised learning system. This will be discussed in Chapter 7.

$$R_{s,o} = I * real(G_{s,o}) \tag{5.1}$$

For each result $R_{s,o}$, where $s = [1..11]$ and $o = [1..13]$ as shown above (5.1), the sum of pixels is holded as the amplitude A_k of the image, where $k = [1..K]$ and $K = 143$ and M, N respectively are the width and height of $R_{s,o}$ is as shown below (5.2).

$$A_k = \sum_{i=0}^{M} \sum_{j=0}^{N} R_{s,o}(i,j) \tag{5.2}$$

These 143 2D Gabor filter values constitutes the feature vector of the image I.

$$F_I = (A_1, A_2, .. A_K) \qquad where\ K = 143 \tag{5.3}$$

5.5. SYMBOL RECOGNITION

The size of the provided Hamparsum data set is relatively small, and therefore the use of traditional machine learning techniques is thought to perform better than more complex techniques such as artificial neural networks (Figure 5.7). There may be a risk of bias and overfitting for complex methods, and it is often not easy to explain what the learning model does on the training data.

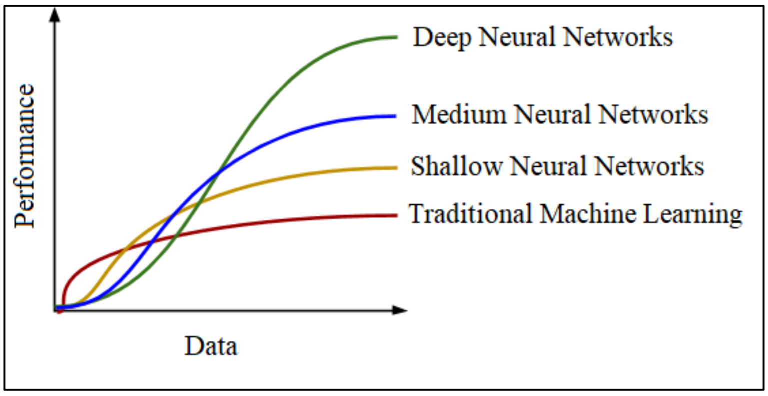


Figure 5.7. Comparison of neural networks and traditional machine learning methods

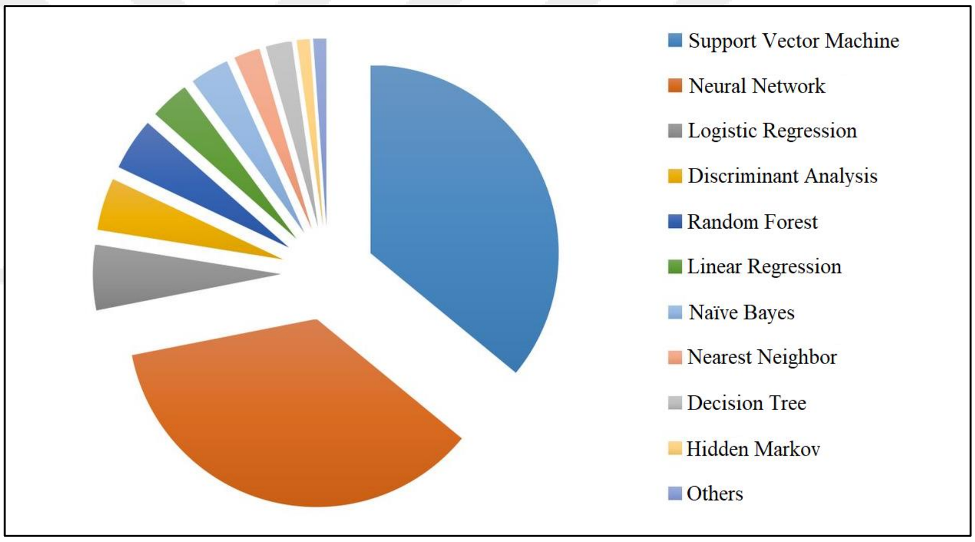


Figure 5.8. Trends in the use of machine learning methods

Various supervised learning methods such as decision tree, random forest classifier and so on have been researched, however depending on the feedback gathered from prior studies, especially including OMR/OCR subjects, the SVM is expected to perform well in such conditions. Thus, Support Vector Machine (SVM) is used in this study to classify the Hamparsum components (Figure 5.8).

There are 1576 instances which were extracted from more than 50 Hamparsum score sheets. In case the note symbols and the auxiliary symbols were segmented successfully in prior steps, then there are no problems with that. However, in reality, nothing should be thought to be perfect, and there are some of symbols and auxiliary components that touch on each other exceptionally (Figure 5.9). These samples were also appended to the training set as if they were a different symbol. Eventually, there are 38 instances on average for a set of 37 distinct patterns.

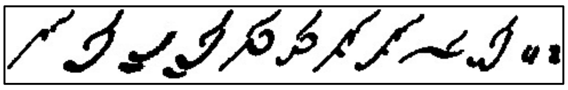


Figure 5.9. Examples of faulty touching symbols

The data set is splitted into 2 parts of 90 percent and 10 percent respectively for training and testing. The testing data are completely separated from the training set. Cross validation is employed to assess the prediction success of the SVM learning model. The scale and orientation parameters of the 2D Gabor filter bank were chosen as 11 and 13, respectively, as these optimum values are detected by the cross validation test on various values. Thus, the 2D Gabor filter bank produces 143 features. In other words, the SVM classifier has 143 dimensions for each individual symbol. The cross validation test results and the success rate of SVM classifier will be shared in further.

5.6. CANDIDATE NOTES ASSEMBLY

After the individual symbols are segmented and successfully classified by using generated features, the next stage of the study is to determine which auxiliary belongs to which symbol. This operation is required to detect the candidate notes for the next template matching phase. A candidate note constitutes a semantic musical note. However the problem is not to have a specified template for the Hamparsum score sheet in terms of positioning of symbols uniformly, and that is why this process was developed.

First of all, the symbols are sorted in a line from left to right by using their bounding box centers. However, the writing styles in this dataset are italic, which means there is a slope of orientation for the association of symbols. Depending on the observations and empirical tests, the slope angle varies around 45°. For that reason, a dilation filter with an inclination of 45° is applied to be able to associate semantically related symbols. The dilation kernel is the transpose of a $5x5$ identity matrix (5.4), so that the slope of the filter will be $45°$ and dilation operation will be achieved with the desired angle. The dimensions of the kernel was determined by the average dimensions of the digital Hamparsum scores and also depending on the observations by experimental operations. The dilation is employed to be able to match the auxiliaries with their belonging individual note symbols.

$$Kernel = \begin{bmatrix} 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.4}$$

Algorithm 5.1. An angle based symbol association algorithm

```
$V$ : {arraylist for the candidate notes}
$\boldsymbol{for}$ each individual symbol $S$ line by line, starting from the leftmost on the top line $\boldsymbol{do}$
    $R_1$ : initial angle of search area is $50°$
    $R_2$ : end line of the search area with an angle of $140°$
    $C$ : Center of $S$
    $NV$ : {arraylist for the symbols contained by the same candidate note}
    Add $S$ to $NV$
    $\boldsymbol{for}$ each individual new symbol $NS$ in the range be $R_1$ and $R_2$ degrees from the $C$ $\boldsymbol{do}$
        Add $NS$ to $NV$
    $\boldsymbol{end}$
    Add $NV$ to $V$
$\boldsymbol{end}$
```

The kernel works well and can often match the most common Hamparsum components. However, in some cases, the respective symbols are positioned too far from each other, and the size of the dilatation kernel does not sufficiently connect between them. For that reason, an angle based symbol association algorithm was developed.

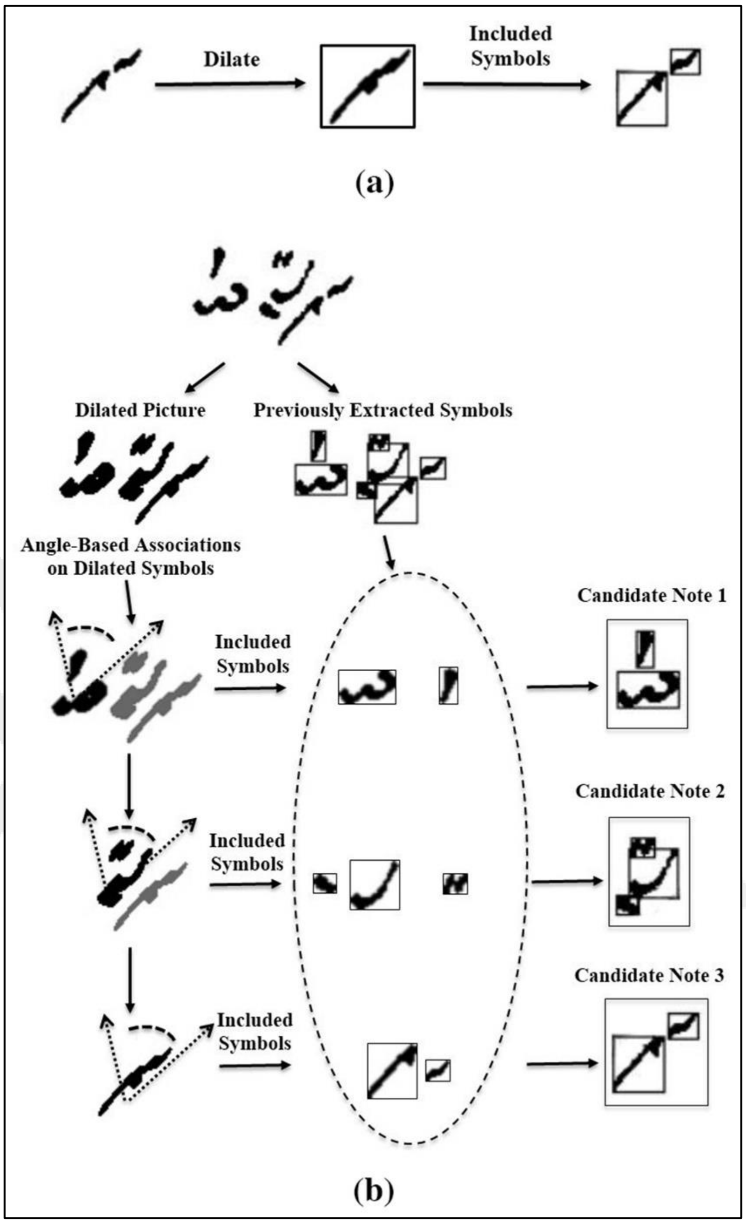


Figure 5.10. (a) Association of individual symbols, (b) association of dilated symbols

On a specific line, by taking the center of the bounding box of the leftmost symbol as the origin, an area based angle looks for the related symbols. The angle range is in between 50° and 140° which is determined upon experiments. Then the symbol is associated to all dilated symbols that the centers of bounding boxes are inside this region. When the association of the leftmost symbol was done, the corresponding individual symbols are extracted from the dilated symbols as shown in figure (Figure 5.10). Then, the next dilated symbol in the same line which was not operated prior has been taken, and so on. The procedure iterates until the rightmost symbol at the bottom line.

5.7. TEMPLATE MATCHING

After the candidate note unities are detected, the next step is the determination of their corresponding Hamparsum notes. A main note symbol would have some variances on correspondant Hamparsum note by using of the auxiliaries depending on the type and the position of the auxiliary symbol. Since the positions and identities of the symbols were found, the templates of look up trees are defined to be able to recognize the notes as shown in figure (Figure 5.11).

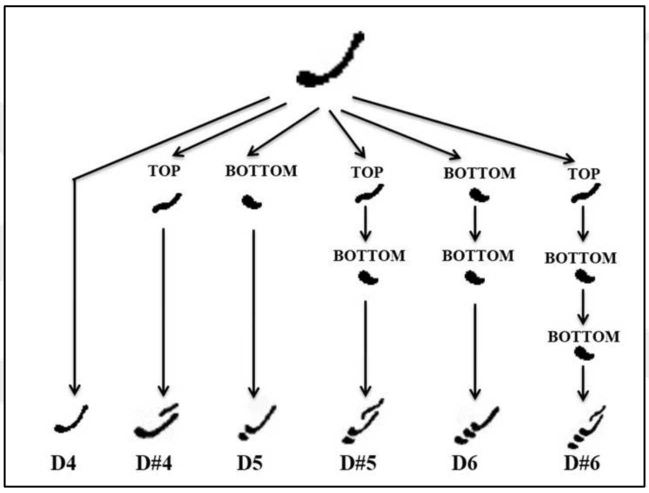


Figure 5.11. Pattern template example

Figure 5.11 is an example for the look up tree of D (Re) note. Each note has a similar predefined tree. Moreover, the duration symbols are also located into the bounding boxes of the assembled note symbol groups. However, assignment of the durations are achieved after the tree structure operates and identifies the note. If any duration symbol is found in the bounding box of the candidate note, then it is assigned, or else if there is no duration symbol on the candidate note then the duration of the previous adjacent note is assigned this note too.

6. IMPLEMENTATION

In this study, a Hamparsum notation OMR system is implemented in C++ programming language. Core libraries of C++ are not adequate for image processing operations, so OpenCV 3.2.0 application programming interface (API) is employed for that purpose. The Makefile for OpenCV library has been created and adjusted by using CMake 3.7.2 tool in Windows 10 environment. In order to compile the code, Qt 5.9.3 with MinGW 5.3.0 release is used. The compiler comes with the Qt Creator 4.8.2 package release, which has been also utilized to design a graphical user interface (GUI) to be able to test the implemented code. Weka software is used for machine learning operations such as the creation of a Hamparsum model with the SVM classifier and cross validation of training and testing data sets.

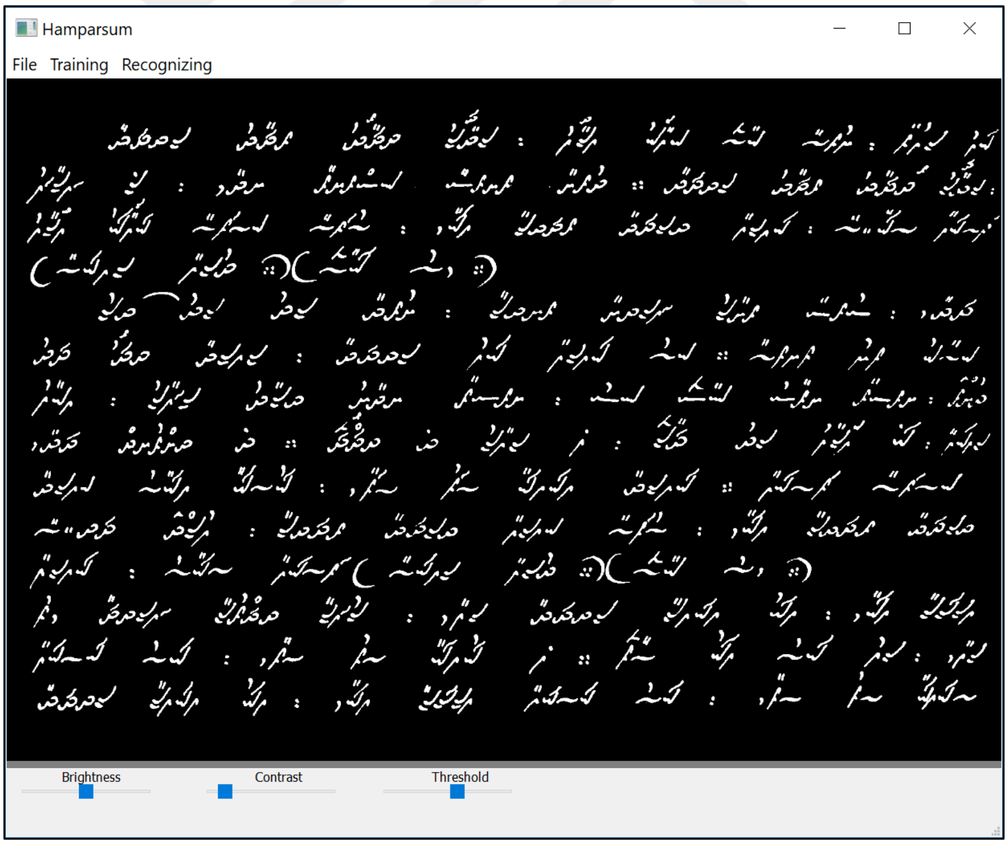


Figure 6.1. The graphical user interface

In Figure 6.1, the graphical user interface of Hamparsum OMR system is shown. This screen is designed to test the system in terms of recognition of individual symbols and semantic musical groups. A sample scanned Hamparsum image would be imported to the label on the screen. In case the binarization and thresholding operations does not produce sufficiently clear image then it would be adjusted by using the slider bars at the bottom. The GUI provides a facility of selecting a specific region on the image by the movement of the mouse cursor. A sample selection might be seen as a rectangle with white frame on sixth line of Figure 6.2.

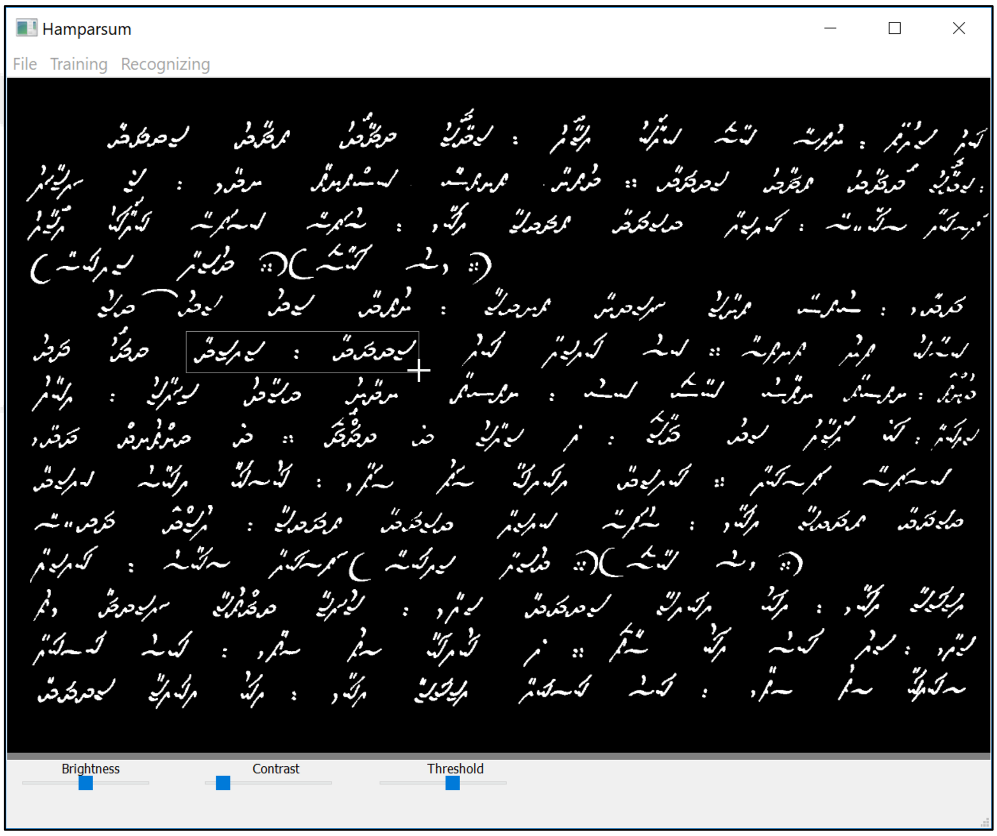


Figure 6.2. Selection of a region on Hamparsum score

After that the region is selected by the user, the recognition process finds the musical notes located into. These recognized musical notes are expressed into separated white rectangle frames as shown below (Figure 6.3).

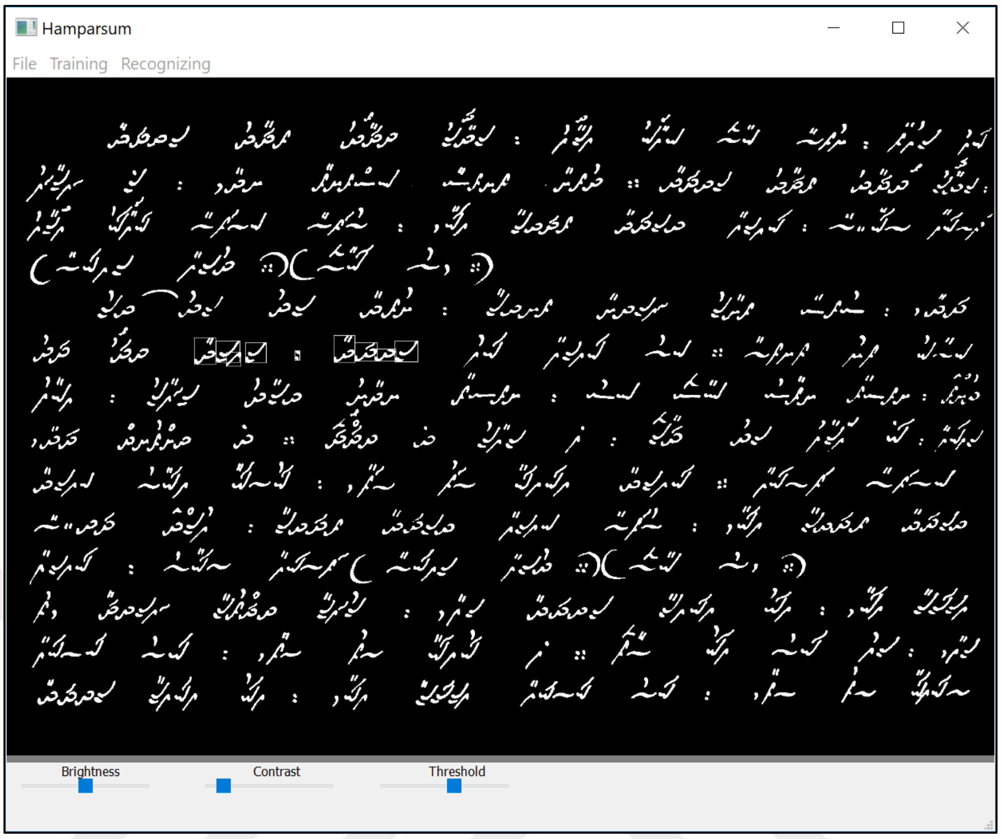


Figure 6.3. Recognized musical notes

In case of erroneous recognition of some components, there is a chance to adjust the threshold parameters and then to run the recognition again by reselecting the notes. The functionalities described above suffices for the main idea of this study.

The proposed method in this study is implemented in a manner of object oriented approach. It consists of a group of classes in a hierarchical relationship as shown in the UML class diagram of the Hamparsum OMR system in Figure 6.4. The functionality of graphical user interface is managed by the MainWindow class. In other words, the user interacts with GUI and instructions of the user are gathered by MainWindow class. Once the recognition of the notes is done then the results are printed into a txt file.

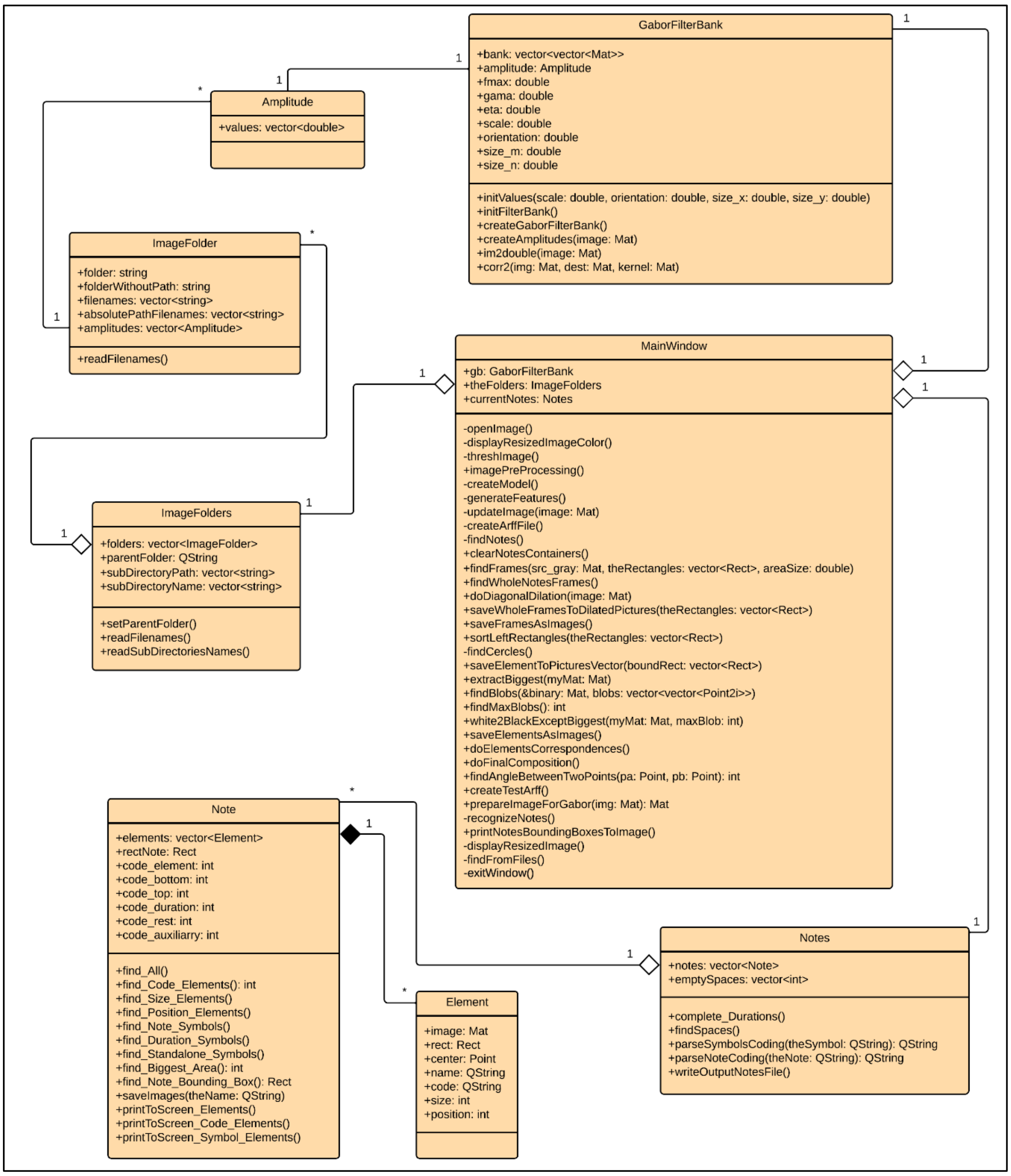

GaborFilterBank
+bank: vector<vector<Mat>>
+amplitude: Amplitude
+fmax: double
+gama: double
+eta: double
+scale: double
+orientation: double
+size_m: double
+size_n: double
+initValues(scale: double, orientation: double, size_x: double, size_y: double)
+initFilterBank()
+createGaborFilterBank()
+createAmplitudes(image: Mat)
+im2double(image: Mat)
+corr2(img: Mat, dest: Mat, kernel: Mat)
Amplitude
+values: vector<double>
ImageFolder
+folder: string
+folderWithoutPath: string
+filenames: vector<string>
+absolutePathFilenames: vector<string>
+amplitudes: vector<Amplitude>
+readFilenames()
MainWindow
+gb: GaborFilterBank
+theFolders: ImageFolders
+currentNotes: Notes
-openImage()
-displayResizedImageColor()
-threshImage()
+imagePreProcessing()
-createModel()
-generateFeatures()
-updateImage(image: Mat)
-createArffFile()
-findNotes()
+clearNotesContainers()
+findFrames(src_gray: Mat, theRectangles: vector<Rect>, areaSize: double)
+findWholeNotesFrames()
+doDiagonalDilation(image: Mat)
+saveWholeFramesToDilatedPictures(theRectangles: vector<Rect>)
+saveFramesAsImages()
+sortLeftRectangles(theRectangles: vector<Rect>)
-findCercles()
+saveElementToPicturesVector(boundRect: vector<Rect>)
+extractBiggest(myMat: Mat)
+findBlobs(&binary: Mat, blobs: vector<vector<Point2i>>)
+findMaxBlobs(): int
+white2BlackExceptBiggest(myMat: Mat, maxBlob: int)
+saveElementsAsImages()
+doElementsCorrespondences()
+doFinalComposition()
+findAngleBetweenTwoPoints(pa: Point, pb: Point): int
+createTestArff()
+prepareImageForGabor(img: Mat): Mat
-recognizeNotes()
+printNotesBoundingBoxesToImage()
-displayResizedImage()
-findFromFiles()
-exitWindow()
ImageFolders
+folders: vector<ImageFolder>
+parentFolder: QString
+subDirectoryPath: vector<string>
+subDirectoryName: vector<string>
+setParentFolder()
+readFilenames()
+readSubDirectoriesNames()
Note
+elements: vector<Element>
+rectNote: Rect
+code_element: int
+code_bottom: int
+code_top: int
+code_duration: int
+code_rest: int
+code_auxiliarry: int
+find_All()
+find_Code_Elements(): int
+find_Size_Elements()
+find_Position_Elements()
+find_Note_Symbols()
+find_Duration_Symbols()
+find_Standalone_Symbols()
+find_Biggest_Area(): int
+find_Note_Bounding_Box(): Rect
+saveImages(theName: QString)
+printToScreen_Elements()
+printToScreen_Code_Elements()
+printToScreen_Symbol_Elements()
Element
+image: Mat
+rect: Rect
+center: Point
+name: QString
+code: QString
+size: int
+position: int
Notes
+notes: vector<Note>
+emptySpaces: vector<int>
+complete_Durations()
+findSpaces()
+parseSymbolsCoding(theSymbol: QString): QString
+parseNoteCoding(theNote: QString): QString
+writeOutputNotesFile()
1
1
1
*
1
*
1
1
1
1
1
1
*
1
*
1
1

Figure 6.4. UML class diagram for Hamparsum OMR system

The MainWindow class organizes all the user interactions. An instance of the MainWindow class is created by starting of the program. One of the most important functionalities of this class is to create a Hamparsum SVM model file and a classification training and testing output file with the extension of Attribute-Relation File Format (ARFF). It also allows user to import and display Hamparsum images in a dimension which is resized depending on the fixed size of the central layer. Another mission of this class is to handle image pre-processing operations such as thresholding, to clean the image from noises a little bit more and to extract Hamparsum symbols in a black and white contrast of the binarized image. Also the recognition processes are organized in the MainWindow class. The GaborFilterBank class creates a bank of 2D Gabor filters with given parameters and applies these filters on a given image then stores their amplitudes as a feature vector. The Amplitude class is used to store aforementioned features and pass them to an instance of the ImageFolder class, so that the features will be stored in an indexed and ordered structure with the other descriptive attributes of images. The ImageFolders class organizes the ImageFolder instances by using a vector structure in order to locate each one of the ImageFolder instance objects and gather their feature values when they are needed. The Element class is defined to describe and handle the attributes such as position and size of every single component such as auxiliary symbols, main note symbols, durations, octave changers etc. that are found in the selected region of the processed Hamparsum image. The Note class holds a vector of the Element instances in order to store each component that belongs to a specific note. In this manner, an instance of the Note class collects all the required information about a specific Hamparsum musical note extracted from a handwritten Hamparsum score. The Notes class organizes the Note class instances in an indexed vector structure to iterate them for the classification operation. The MainWindow class creates instances of the GaborFilterBank, the ImageFolders and the Notes classes during the initialization itself.

The program starts with a window including an empty frame and the user might select to open an image by clicking the “Open” subitem of the “File” item in the menu on the top of the window. Then the imported image is displayed in a resized form in order to fit into the image frame. At first sight, image is displayed with the colors it has. When the user clicks somewhere on the window, then the image is binarized into black and white colors. There are three slidebars at the bottom of the window as respectively “Brightness”, “Contrast” and “Threshold” which might be used to obtain a clear and sharp symbols. An image pre-processing method is invoked each time the user changes the slidebar parameters and the processed image is redisplayed at the end of the pre-processing. In case the user decides that the image is satisfiably clear for the recognition then he/she might select a region of interest (ROI) consists of Hamparsum characters by clicking, moving and releasing the mouse cursor. During the movement of cursor, the graphical user interface will emphasize the ROI in a rectangular white frame. It is possible to select a new ROI in case of an incorrect selection by clicking, moving and releasing the cursor again. The erroneous selection will be removed and the new selection will be displayed for sure. Once the ROI has been determined, the recognition process starts if the user selects the “Find Notes” subitem under the “Recognizing” menu item. First of all, the container objects are cleared to ensure that they do not contain any irrelevant data. Bounding boxes of each element in the ROI is detected and saved into a container vector instance. Moreover, a dilation operation is applied to unify the main note symbols and relevant auxiliary symbols, then the bounding boxes of each unified components are detected. The biggest element areas are holded into the frame and the remainings are excluded. Those frames of the dilated elements are also saved in another container vector instance. The dot symbols are such problematic shapes compared to others, because they are tiny and might not be unified with the related notes. Therefore, a special process is applied to find those dot symbols and to make it possible to match with their correspondant note elements if exists. After that moment, both container vectors are sorted internally and all the matching elements are determined via applying special angle based operation. Feature amplitudes of each notes are obtained by applying 2D Gabor filter bank and those values are saved into another container vector instance, besides, they are written into a file in arff format which is compatible with Weka. The created arff file will be used as an input parameter of the recognition by Weka cross validation function. After the recognition is achieved, then the note definitions are written into a result file. At the same time on the GUI, each note will be emphasized in distinct rectangular frames.

In order to achieve the operation of Hamparsum notes recognition described above, training data should be analyzed in prior and also a model file must be created. The Hamparsum training data set which consists of isolated sample individual symbols has been prepared before and organized in a nested folder structure. When the user selects the “Create File” subitem of “Training” item on the top menu, the training data set is processed and the amplitudes of 2D Gabor filter bank features for each notes are written into a training file in arff format. If the user selects the “Create Model” subitem of “Training” item on the top menu, depending on the training data set feature amplitudes, an SVM model for Hamparsum OMR system is created and saved into a file. Finally, when the user selects the “Exit” subitem of “File” item on the top menu, the GUI window is closed and the program quits successfully. The use case diagram of the Hamparsum OMR system is described in the following Figure 6.5.

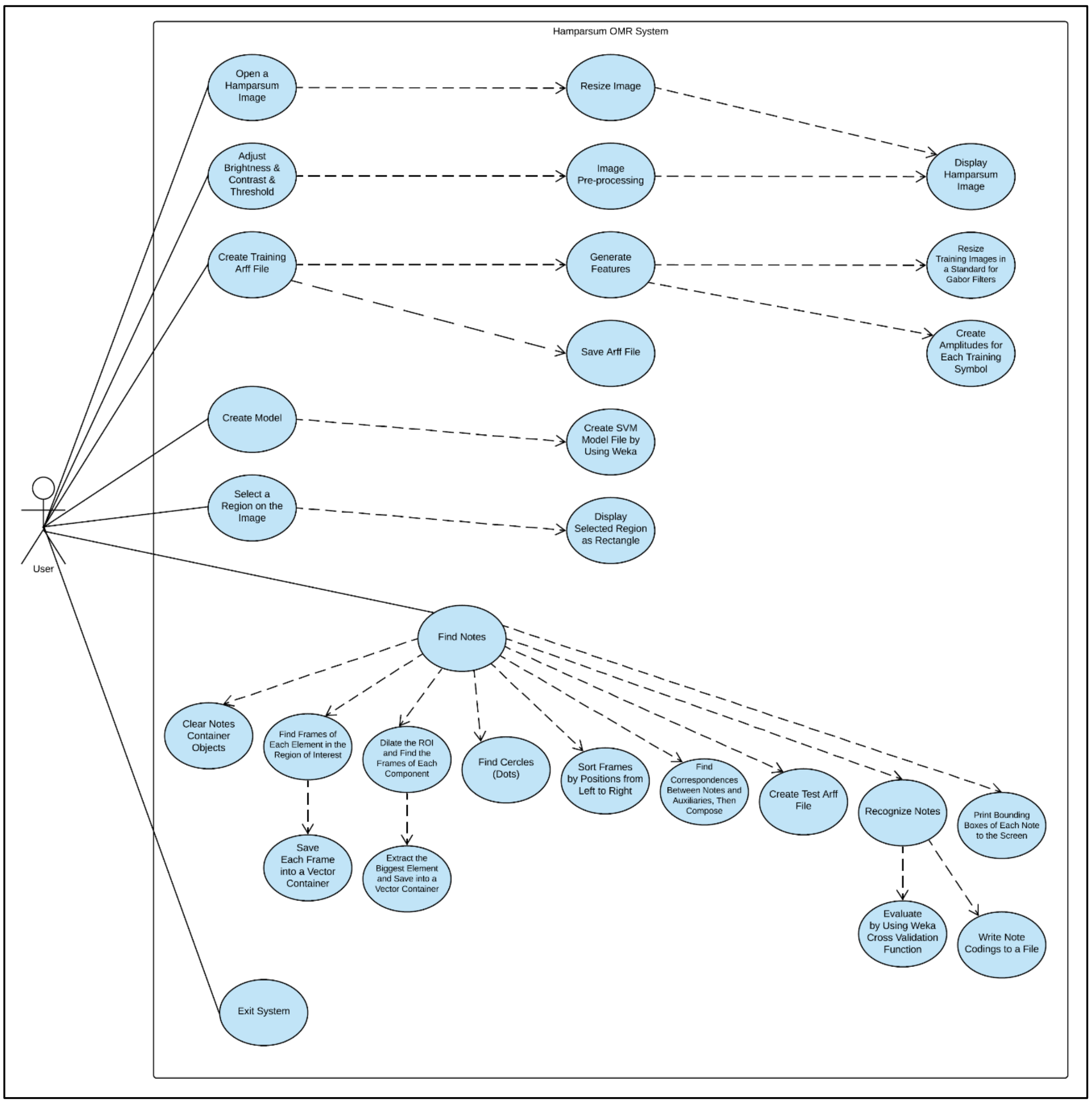

Hamparsum OMR System
User
Open a Hamparsum Image
Resize Image
Display Hamparsum Image
Adjust Brightness & Contrast & Threshold
Image Pre-processing
Create Training Arff File
Generate Features
Resize Training Images in a Standard for Gabor Filters
Save Arff File
Create Amplitudes for Each Training Symbol
Create Model
Create SVM Model File by Using Weka
Select a Region on the Image
Display Selected Region as Rectangle
Find Notes
Clear Notes Container Objects
Find Frames of Each Element in the Region of Interest
Dilate the ROI and Find the Frames of Each Component
Find Cercles (Dots)
Sort Frames by Positions from Left to Right
Find Correspondences Between Notes and Auxiliaries, Then Compose
Create Test Arff File
Recognize Notes
Print Bounding Boxes of Each Note to the Screen
Save Each Frame into a Vector Container
Extract the Biggest Element and Save into a Vector Container
Evaluate by Using Weka Cross Validation Function
Write Note Codings to a File
Exit System

Figure 6.5. UML use case diagram for Hamparsum OMR system

7. TEST AND EVALUATION

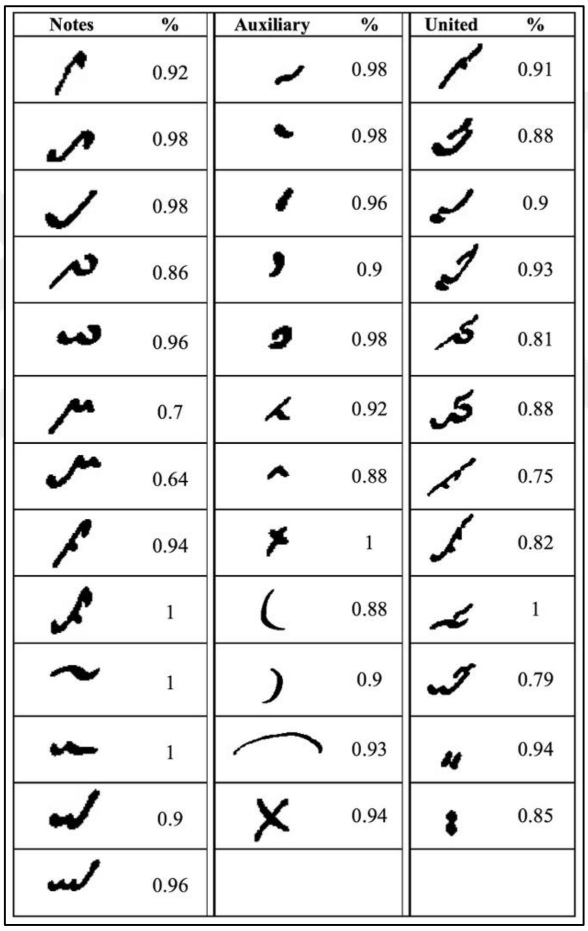
7.1. RECOGNITION OF SYMBOLS

The total size of training dataset was more than 22000 symbols, extracted from 50 representative scores in terms of various handwriting styles and different levels of clarity. Three lines were randomly selected from each score, which corresponds to about 100 symbols per score. Whilst 1576 instances of 37 distinct symbols including main symbols and auxiliary symbols are randomly selected by considering representability, and the data set is splitted into 90 percent for training and 10 percent for testing partitions, a bank of 2D Gabor filters is generated to evaluate the SVM classifier. The success rate of the SVM classifier varies depending on the content and size of the feature set. The content and size of 2D Gabor filter bank changes by the input parameters of scale and orientation. In other words, the selection process of these two filtering parameter values directly manipulates the results. The multiplication of these two parameter values gives the total number of features. Therefore, the selection of them was made by applying a leave-ten-out (LTO) cross validation test and detected the best pair of values with the best success rate accordingly. For the training and validation process, Weka software [24] with an SVM classifier has been utilized.

Table 7.1. Cross validation test results of 2D Gabor filter bank parameters

Scale \ Orientation	**9**	**10**	**11**	**12**	**13**	**14**
9	90.6	90.6	91.0	90.9	90.8	90.8
10	91.0	91.0	91.0	91.0	91.0	91.0
11	90.8	91.2	91.2	91.2	90.7	91.2
12	90.8	90.6	90.6	90.8	90.7	90.7
13	90.8	91.0	**91.3**	90.8	90.8	90.8
14	91.0	90.8	90.8	90.8	90.9	90.8

The results of the cross validation test to detect the success rates of the SVM classifier depending on the scale and orientation parameters are seen in the Table 7.1. The values less than 9 for both gave worse results, however all the values above are higher than 90 percent and they are very close to each other. Clearly, the best pair of values are $s = 11$ for the scale and $o = 13$ for the orientation, which has the success rate of 91.3 percent. Whilst the size of 2D Gabor filter bank is 143, the success rates of individual symbols might be seen in following table (Figure 7.1).



Notes	%	Auxiliary	%	United	%
	0.92		0.98		0.91
	0.98		0.98		0.88
	0.98		0.96		0.9
	0.86		0.9		0.93
	0.96		0.98		0.81
	0.7		0.92		0.88
	0.64		0.88		0.75
	0.94		1		0.82
	1		0.88		1
	1		0.9		0.79
	1		0.93		0.94
	0.9		0.94		0.85
	0.96				

Figure 7.1. Success rates of individual symbols for the SVM classifier

As a note, the definition of success rate so far is the number of correctly identified instances divided by the total number of instances individually. However, the frequency of some symbols usage is more than some others. Therefore depending on that observation, the weighted accuracy of the success rates in the data set is calculated as 97.7 percent.

7.2. RECOGNITION OF SEMANTIC MUSICAL GROUPS

Rather than recognizing only individual symbols, it is also necessary to evaluate the recognition of semantic musical groups. Because, for instance, a note symbol alone can only specify the frequency of the corresponding tone, but it may only be obtained how long the duration of this note should continue, by the utilization of auxiliary symbols. In this case, more than one symbol must be logically associated by the system. Furthermore, even if there are no auxiliary duration symbols over the note symbol, there might be a duration defined automatically due to the duration of the previous note. Moreover, some notes reflect octave changes or accidental tone markers by means of auxiliary symbols. In these cases it is expected that the semantic integrity of the groups consisting of more than one symbol can be captured.

In order to evaluate the success rate of the recognition of Hamparsum notes as semantic musical groups where each note comprises one or more symbols, 50 representative scores were selected from the Leon Hanciyan’s collection that were not used in the training process. Eventually, more than 5000 semantic musical groups were tested by comparing manually the automatically recognized note with the true note one by one. For that reason, as for the recognition of the semantic musical groups, a pattern matching procedure was followed, a cross validation method is not applicable. The success rate was 75 percent. In the following subsection the types of errors produced by the system will be analyzed.

7.2.1. Types of Errors

During the recognition process it is considered that a note or a symbol to be successfully recognized only if a valid recognition of the whole candidate note is performed. For example if the note is recognized but not the duration or the accidental then the recognition is classified as erroneous. Most of the errors produced by the system fall into one of the following categories.

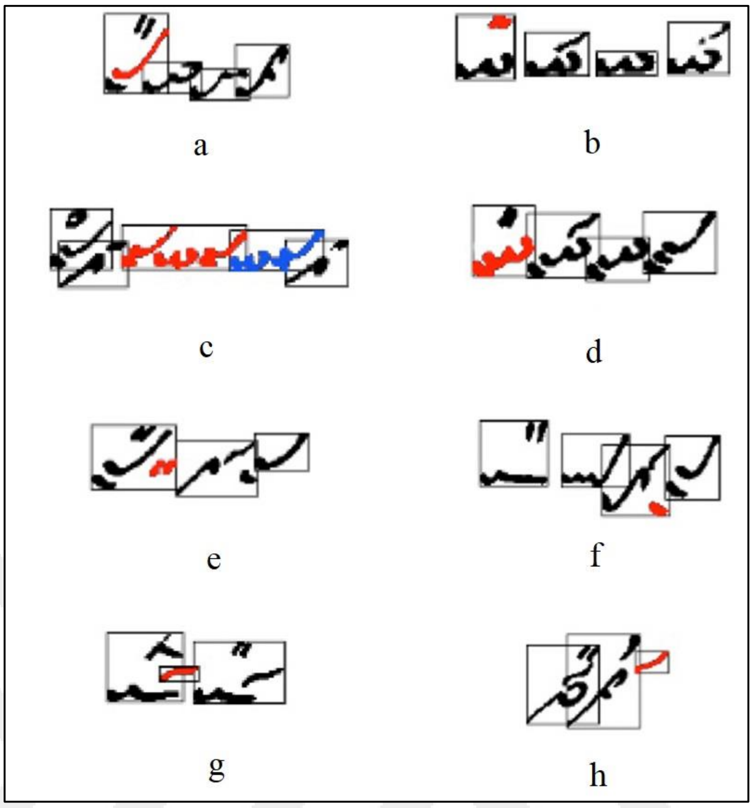


Figure 7.2. Recognition errors examples: (a) (b) erroneous symbol recognition, (c) (d) proximity errors, (e) (f) bounding box errors, (g) (h) slope range errors

7.2.1.1. Erroneous Symbol Recognition

In case of an individual symbol is not recognized correctly, the whole candidate note is evaluated as incorrect. In Figure 7.2.a, the first candidate note consists of four individual symbols and if one of them (e.g. *Push* in red) is faultly recognized then the candidate note is as incorrectly recognized. This is relatively a tolerable case, because the error affects only the related note. However, the worst case occurs when the recognition of duration symbol over the first candidate note was incorrect. Because, it manipulates to whole subsequent notes without a duration symbol. For example, in Figure 7.2.b the duration symbol in the first note (in red) is not correctly recognized. Consequently, even if all the symbols in the rest of this sequence are correctly recognized, the whole sequence is categorized as incorrect. This type of error is frequent as the duration symbols are small and similar. This is the major reason for the significant decrease of the success rate of the system compared to the individual symbols success rates of the SVM classifier.

7.2.1.2. Proximity Errors

Some symbols are touching each other by the fault of the author. In such cases, these subsequent symbols are associated incorrectly with each other. In Figure 7.2.c it might be seen that the notes are recognized together (the red and the blue group) because they are touching or almost touching to each other. The first note in Figure 7.2.d cannot be recognized because of the two sticked symbols (in red), as it is a rare case that was not defined in the classifier.

7.2.1.3. Bounding Box Errors

This kind of error corresponds to the case when a rest or note symbol is completely inside the bounding box of a candidate note that it doesn’t belong to. In that case the symbols are wrongly associated with the same candidate note. In Figure 7.2.e, the bounding box of the second note includes a rest eighth symbol (in red) that should be treated separately. Figure 7.2.f shows a similar case where the bounding box of the third note includes a symbol (in red) belonging to the fourth note.

7.2.1.4. Slope Range Errors

In some cases, a symbol is written with a different slope than usual. In that case some symbols are excluded from the candidate note leading thus to errors. Figure 7.2.g and Figure 7.2.h display symbols in red color that were excluded from the candidate note to which they should belong because they were found outside the usual slope region.

7.3. COMPARISON WITH OTHER OMR SYSTEMS

As emphasized in Chapter 2, no other study has been published on the OMR of the Hamparsum notation. As far as it is known, this work is the first effort on the OMR of the handwritten Hamparsum notation. Also, concerning the comparison of a Hamparsum note in terms of the semantic musical group methodology, none of the investigated studies focus on musical notes as groups of symbols. Therefore, it is not possible to compare the results and methods directly with another study. However, in terms of feature extraction and classification, it is worth sharing by comparing the results of similar approaches in other traditional OMR studies.

European notation OMR studies are in a little bit different concept, because of having a dissimilar structure with Hamparsum notation. The stave entity is utilized to determine the position of the character, that identifies decisive information about the notes. However, there are some similarities in terms of binarization and noise reduction processes in [3] and [4]. Nevertheless, it is not very meaningful to compare them with the OMR of Hamparsum as a whole, since their data sets consist of printed scores, not manuscripts.

The systems of Thai sheet music (TSM) [5], Chinese Kunqu opera scores with Gong-Che notation (GCN) [6] and Hellenic Byzantine music (HBM) notation [7] are traditional samples, and the OMR studies on these domanis are more related with this work to compare in terms of feature extraction, classification and semantic musical groups structure.

From the feature extraction methods point of view, the study for the HBM notation uses a set of features based on wavelets. The utilized method in this Hamparsum OMR study is rather similar as it is based on Gabor features. In TSM study, the feature extraction method is not explicitly defined. Nevertheless, the study is dealt with only seven printed versions of Thai notes, thus with a problem that it is quite straightforward compared to the recognition of Hamparsum symbols. On the other hand in GCN, the feature extraction method is based on statistical features extracted from a two-dimensional grid containing notes. That type of features are inadequate for Hamparsum symbols as GCN symbols are based on symbols like the Asian logograms that have fairly large size and complex shape.

Regarding the classifiers used in the above studies, in [7] a nearest neighbor classifier is used with a success rate of 99.4 percent. However, it is worth mentioning that Hamparsum data set comprises handwritten symbols whereas in [7] the data set is composed of printed symbols. Hamparsum data set was tested with a nearest neighbor classifier using a cross-validation method, and the performance was very close (95.4 percent) but slightly worse than SVM classifier (97.7 percent). In [5], as in this study, an SVM classifier is used, with an average success rate of 79 percent. In [6], Bayesian, genetic algorithm, and K-nearest neighbor classifiers are utilized but with success rates that are lower than 68 percent.

8. CONCLUSION

In this study, a method for the recognition of handwritten Hamparsum music notation is proposed. In order to kick the first effort off on this subject, a collection of Leon Hanciyan in the Ottoman Archives in Istanbul was obtained. The presented method initially aims to detect each individual symbol elements including main notes, auxiliaries, durations and rests. Support vector machine classifier method was employed to the 2D Gabor filter bank features of every single element in order to identify those symbols depending on a model which was prepared by using a training data set. In other words, a supervised learning method has been preferred because the handwriting style of the Hamparsum notation authors vary exceedingly. After the first step is achieved, then the relations among the recognized elements need to be established. By utilizing the standard positionings and proximities among the auxiliaries and the main note symbols, applying a special dilation process often satisfies to make them unify and achieve to establish a relation. Some special methods were developed to handle some exceptional issues, however some rarely occurring cases were ignored. When all the elements are collected and organized together according to their relations, then a template matching method was applied to identify the detected musical groups in terms of musical sounds. The Hamparsum training model file was created by using the 2D Gabor filter bank feature amplitudes obtained from the training data set which has been manually preprocessed.

The scale and the orientation parameter sizes of the 2D Gabor filter bank were specified depending on the test results obtained during the analysis phase. Also a special angle based method was developed to match the relevant auxiliaries with their main note symbols. The angle of the interval to be scanned was also determined by trying different inputs values. There are still some erroneous cases which might be possible to eliminate by applying various approaches such as heuristic algorithms like finding general duration for each block of notes or detecting errors based on bigram frequencies of notes. By concerning the usage weights of the notes, the success rate has been computed as 97.7 percent as a result of cross validation testing.

There are not any other OMR study based on handwritten Hamparsum music notation to compare the results, therefore the proposed method is a precessor attempt for the aforementioned subject. This study might contribute to promote this unique cultural heritage as it deserves, and help someones who interested in translating Hamparsum manuscripts to the European notation which is used world wide.

REFERENCES

1. Murrock CJ. Music and mood. *Psychology of Moods*. 2005;1(8):141–55.

2. Güngör K. Historical development of notation. *Sahne ve Müzik Eğitim - Araştırma Dergisi*. 2015;1(1):81–9.

3. Rossant F. A global method for music symbol recognition in typeset music sheets. *Pattern Recognition Letters*. 2002;23(10):1129-41.

4. Rebelo A, Fujinaga I, Paszkiewicz F, Marcal ARS, Guedes C, Cardoso JS. Optical music recognition: state-of-the-art and open issues. *International Journal of Multimedia Information Retrieval*. 2012;1(3):173-90.

5. Kusakunniran W, Prempanichnukul A, Maneesutham A, Chocksawud K, Tongsamui S, Thongkanchorn K. Optical music recognition for traditional Thai sheet music. *2014 International Computer Science and Engineering Conference (ICSEC)*, 2014:157-62. IEEE.

6. Chen GF, Sheu JS. An optical music recognition system for traditional Chinese Kunqu Opera scores written in Gong-Che Notation. *EURASIP Journal on Audio, Speech, and Music Processing*, 2014;1(7):1-12.

7. Gezerlis VG, Theodoridis S. Optical character recognition of the Orthodox Hellenic Byzantine Music notation. *Pattern Recognition*. 2002;35(4):895-914.

8. Asebriy Z, Raghay S, Bencharef O, Chihab Y. Comparative systems of handwriting Arabic character recognition. *Second World Conference on Complex Systems (WCCS)*, 2014:90-3. IEEE.

9. AlKhateeb JH, Ren J, Jiang J, Al-Muhtaseb H. Offline handwritten Arabic cursive text recognition using Hidden Markov Models and re-ranking. *Pattern Recognition Letters*. 2011;32(8):1081-8.

10. AlKhateeb JH, Pauplin O, Ren J, Jiang J. Performance of hidden Markov model and dynamic Bayesian network classifiers on handwritten Arabic word recognition. *Knowledge-Based Systems*. 2011;24(5):680-8.

11. Parvez MT, Mahmoud SA. Arabic handwriting recognition using structural and syntactic pattern attributes. *Pattern Recognition*. 2013;46(1):141-54.

12. Kessentini Y, Paquet T, Hamadou AB. Off-line handwritten word recognition using multi-stream hidden Markov models. *Pattern Recognition Letters.* 2010;31(1):60-70.

13. Hatipoğlu E. The ahenks in Turkish music. *İSTEM*. 2013;22:131–44.

14. Karamahmutoğlu G. Hamparsum Limonciyan and his musical notation system. *Journal of Music and Science*. 2009;6(5):73–90.

15. Carron T, Lambert P. Color edge detector using jointly hue, saturation and intensity. *Proceedings of 1st International Conference on Image Processing*, 1994:977-81. IEEE.

16. Qiao Y, Hu Q, Qian G, Luo S, Nowinski WL. Thresholding based on variance and intensity contrast. *Pattern Recognition.* 2007;40(2):596-608.

17. Suzuki S. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 1985;30(1):32-46.

18. Heckbert PS. A seed fill algorithm. *Graphics Gems*. 1990;1(4):275–7.

19. Fogel I, Sagi D. Gabor filters as texture discriminator. *Biological Cybernetics*. 1989;61(2):103-13.

20. Adak C. Gabor filter and rough clustering based edge detection. *2013 International Conference on Human Computer Interactions (ICHCI)*, 2013:1-5. IEEE.

21. Roy K, Das RN, Kar S. *Understanding the basics of QSAR for applications in pharmaceutical sciences and risk assessment*. Amsterdam: Academic Press; 2015.

22. Liang JI, Piper J, Tang JY. Erosion and dilation of binary images by arbitrary structuring elements using interval coding. *Pattern Recognition Letters*. 1989;9(3):201-9.

23. Browne MW. Cross-validation methods. *Journal of Mathematical Psychology.* 2000;44(1):108-32.

24. Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH. The WEKA data mining software: an update. *ACM SIGKDD Explorations Newsletter*. 2009;11(1):10-8.