Politecnico di Milano
Facoltà di Ingegneria dell'Informazione

Polo Regionale di Como
Master of Science in Computer Engineering
Sound Engineering and Design

OMRJX
A framework for piano scores optical music recognition

Supervisor
Ing. Matteo Matteucci

Author
Gianmarco Gozzi
ID: 734129

Academic Year 2009/2010

Politecnico di Milano
Facoltà di Ingegneria dell'Informazione

Polo Regionale di Como
Corso di Laurea Specialistica in Ingegneria Informatica
Ingegneria e Design del Suono

OMRJX
Un sistema per il riconoscimento ottico di spartiti per pianoforte

Relatore
Ing. Matteo Matteucci

Autore
Gianmarco Gozzi
ID: 734129

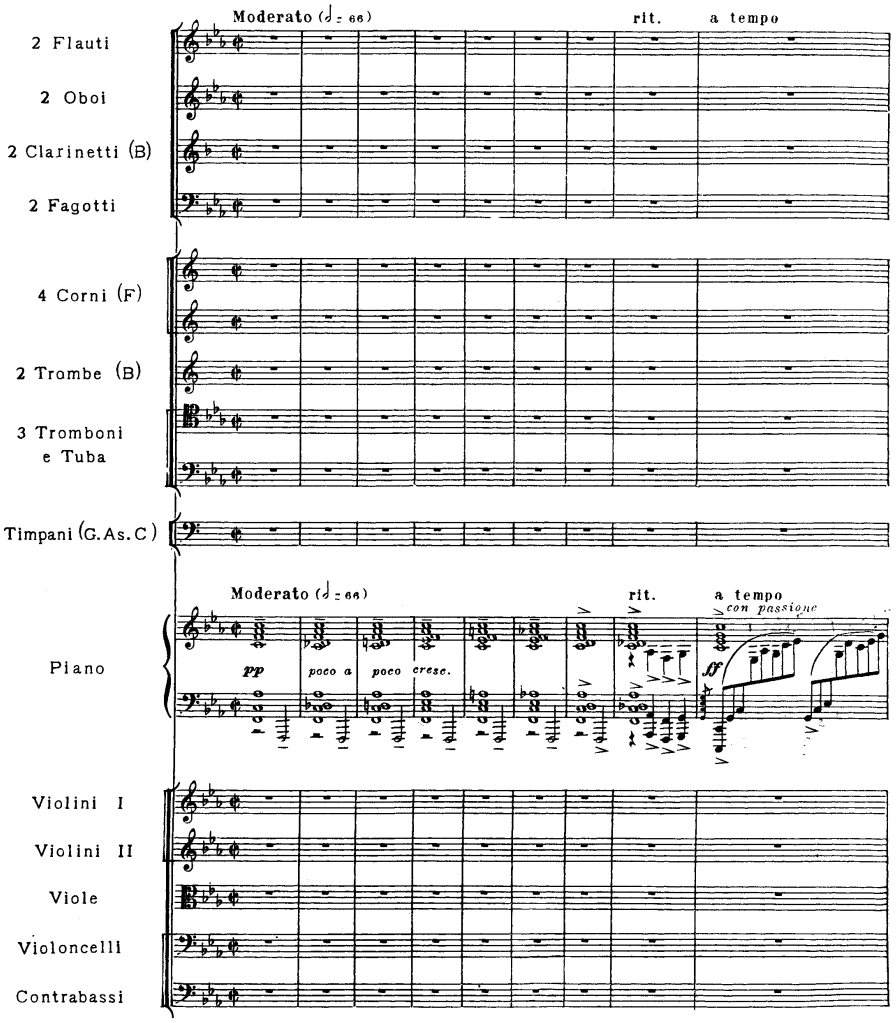
Anno Accademico 2009/2010

To my Music and my Pianos

Piano Concerto No. 2

in C Minor, Op. 18

I



Ringraziamenti

Un sentito ringraziamento va al prof. Matteo Matteucci per la serietà e la disponibilità offerta durante lo svolgimento di questo lavoro.

Questa tesi è il risultato di anni di sacrifici. Sacrifici che ho fatto con e per la mia famiglia. Dedico, infatti, a loro, e non ai miei pianoforti, i grandi obiettivi che siamo riusciti a raggiungere insieme. Con questo traguardo siamo riusciti a dimostrare ancora una volta che siamo una famiglia molto unita. Quindi un grande grazie va ad Andrea, Bea, Gaia, Marisa e Rossi. Una dedica speciale va a chi ha lasciato una meravigliosa impronta dentro di me e che oggi purtroppo non può condividere questa gioia con noi, grazie Pino.

Grazie ad Edda e Italo per il continuo supporto morale e non solo!

Questa tesi è dedicata anche a due persone molto speciali che hanno fatto di me ciò che sono oggi. Due esempi di saggezza e luce nella mia vita. Due grandissimi esempi di umiltà da cui prendere esempio e a cui tendere asintoticamente. Senza di voi oggi sarei solo un quarto di quello che sono e questa tesi forse non esisterebbe neanche. Due fratelli. Grazie al Maestro Monsi Dave e a Filippo Carlo Azeglio Cona 2 in numero romano.

Arigatou Gozaimasu Mika Satake. Grazie per essere stata sempre paziente con me, per avermi aiutato nei momenti difficili e per avermi fatto avvicinare ancora di più alla musica. Grazie anche per l'ospitalità e per l'aiuto nel calcolare le teste delle note :D Kisu Rabu!

Grazie alla mia famiglia Comasca. Mi avete fatto crescere tantissimo e vi ringrazio di cuore per questi due anni passati insieme. Un ringraziamento speciale va a Dani, Luca e Davide.

Quindi, un ringraziamento particolare va alle stupende amicizie romagnole a cui penso ogni giorno. Così lontane, così vicine (da 26 anni). Grafazifiefe.

Per concludere, grazie a tutti quelli che non sono stati nominati ma che hanno contribuito alla mia crescita personale.

Gianmarco Gozzi

Sommario

Il riconoscimento automatico di partiture, detto OMR (dall'inglese optical music recognition), è una branca del riconoscimento ottico dei caratteri (OCR). OMR è un processo che acquisisce automaticamente le informazione di uno spartito elaborandole da una immagine digitalizzata di tale spartito. Possibili applicazioni possono coinvolgere l'ambito dell'archiviazione, applicazioni per non vedenti o armonizzazioni automatiche.

Negli anni, molti sono stati gli sforzi affrontati nel campo dell'OMR. I pionieri furono Pruslin e Prerau che, negli anni '70 presso il MIT di Boston, gettarono le prime basi nel riconoscimento ottico di spartiti musicali. Fino alla fine degli anni '90 l'approccio base fu quello di dividere il problema in due parti, una prima estrazione del pentagramma, componente principale dello spartito, e poi un conseguente riconoscimento dei vari simboli musicali (note, pause, etc). Recentemente l'approccio sta cambiando direzione e si fanno tentativi di riconoscimento senza una iniziale rimozione dei pentagrammi, utilizzando strumenti quali catene di markov nascoste (HMM) e reti neurali.

Questa tesi getta le basi per un futuro lavoro sulla tassonomia di autori classici di pianoforte. Per poter discriminare tra vari autori è neccessario ottenere delle informazioni preziose che sono contenute nello spartito stesso. *OMRJX*, nome del sistema creato in questo lavoro di tesi, nasce come strumento preliminare per tale raccoglimento di informazioni[1]. Di fatto, il sistema prevede un riconoscimento totale della maggior parte delle componenti che si possono rilevare su uno spartito classico per pianoforte. Esso non pretende di risolvere il problema OMR relativo a spartiti degradati ma cerca invece una soluzione ottimale per spartiti generati da un programma di notazione musicale (e.g., *Finale*). Prima di affrontare l'implementazione del sistema, si è studiato lo stato dell'arte in campo OMR per poter scegliere gli strumenti più adeguati a certe determinate situazioni.

OMRJX prevede un ampio utilizzo della maggior parte degli strumenti utilizzati per la visione e intelligenza artificiale. Per esempio, il riconoscimento del pentagramma e delle barre delle battute è stato effettuato utilizzando la trasformata di hough sullo spartito binarizzato. Algoritmi di clustering sono stati, invece, utilizzati per il riconoscimento di un possibile titolo dello spartito[2] e per il riconoscimento delle teste delle note piene. Inoltre, la tesi propone due nuovi e validi algoritmi per l'identificazione del valore di tali note piene; il primo fa uso del processo di

[1] Nonostante l'esistenza di software commerciali che affrontano il problema OMR, *OMRJX* è stato creato per poter ottenere delle informazioni che tali software non sono in grado di restituire.

[2] Informazione che nessun software commerciale restituisce e che potrebbe essere utile per il riconoscimento dell'autore.

scheletrizzazione morfologica e l'altro utilizza il metodo delle proiezioni. Infine sono state ampiamente utilizzate tutte le operazioni morfologiche e le tecniche di *template matching* per il riconoscimento dei restanti simboli.

Il sistema ha ottenuto ottimi risultati e dai test effettuati sono emerse comunque delle falle. Quindi, per tali lacune sono state pensate delle possibili soluzioni che verranno applicate in un lavoro futuro. *OMRJX* è stato quindi testato su una serie di possibili spartiti per pianoforte e poi confrontato anche con le prestazioni di altri software commerciali ottenendo dei risultati competitivi.

In conclusione, il sistema, dal nome *OMRJX*, creato per questa tesi è uno software che fa uso di nuovi algoritmi per la risoluzione del problema OMR su spartiti per pianoforte, spartiti che presentano innumerevoli difficoltà viste le possibili complicazioni che la scrittura pianistica può offrire.

Abstract

Optical Music Recognition (OMR) is the OCR branch oriented to musical documents. First works on OMR are dated back to the early '70s and till nowadays, the scientific community has exerted to obtain the best from this open issue. An OMR process tries to recognize, from a scanned real page of music, all musical symbols that live that page. Unfortunately, it is, often, a very arduous task to accomplish. This master thesis will cover all past works concerning the OMR problems, emphasizing on the tools and methodologies used for a final solution. Hence, a MATLAB® system framework will be developed and explained into details. The software will make use of most of the computer vision and artificial intelligence tools, adding a new solution for the OMR issue on scorewriters' not deteriorated scores. Such program is meant not only to be a simple OMR application but also the starting point for a future work about classical piano author discrimination.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Music is enough for a lifetime, but a lifetime is not enough for music.

Sergei Vasilievich Rachmaninoff

Optical Music Recognition (OMR) was born at the MIT laboratories during the very end of the '60s. It is a branch of OCR (optical character recognition) and is meant to recognize objects on a music sheet. In a score live notes and music. Not only. As music developed throughout the centuries, also scores had to change their content. The frequency of dynamics, accents, words, accidentals significantly increased. Hence, a score can contain a large variety of objects to be recognized. Furthermore, each instrument has its own music notation and score formatting. To conclude, flawless OMR is a very hard target to reach and infact, nowadays there exist several applications that face the OMR challenge but never achieving perfect accuracy.

1.1 Goals and Motivations

As a pianist and a composer, I lived (still live) surrounded by music and music sheets.

As a pianist, I always wanted to enhance my studying tools. While playing pieces by composers such as Liszt, Rachmaninov, Scriabin, Ligeti, there's no time to turn a page. It would be really interesting if the composition could interact with the player (and not necessarily viceversa). A sort of living score that can listen to the player and react to player's stimuli.

As a composer, I have the natural tendency to imagine what really doesn't exist. Flows of notes, structures, musical landscapes that interleave continuously. What if these streams could be taught to a computer. What if a computer could learn thousands of scores. With this thesis, my goal is to create the starting point for a future development of *instruments* for pianists and composers. Especially, the basis for classical piano author recognition has been jotted down, but they need the support of an ad-hoc OMR system. **OMRJX** is a complex architectural system that challenges optical music recognition and try to collect as much information as possible from a piano music page.

1.2 Thesis Outline

The structure of the dissertation is the following:

- **Chapter 2** provides the basis of piano music notation in order to easily understand each part involved in the project.
- **Chapter 3** is a complete overview of the state of the art about OMR issue. In this chapter the methods and tools used in the past and useful for the building of *OMRJX* are listed and deeply analyzed.
- **Chapter 4** approaches the system framework logic and explains its main structure.
- **Chapter 5** describes in detail all processes that have been presented in chapter 4.
- **Chapter 6** shows the system results after being tested among a series of scores and against other commercial softwares.
- **Chapter 7** and **Chapter 8** expose conclusions and future works.

Chapter 2

Music Notation

This chapter will briefly introduce most of the musical symbols involved in piano scores.

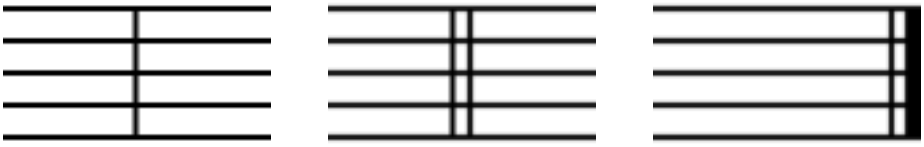
2.1 Lines

All music sheets are centered on staves. A staff consists of five parallel lines[1] (same length) and four intervening spaces (same height). These two components are the musical translation of note pitches. Music objects can lie on the staff but can also lie above or under the staff limits. Hence, the musical notation makes use of a ledger to extend the staff to pitches. Such ledger lines are placed behind the note heads, and extend a small distance to each side. In Fig. 2.1 are shown raw examples of a staff and a ledger. As far as the temporal dimension of a staff is concerned, the time measure is delimited with vertical bars which connect the upper and lower staffs of a double staff (*Grand Staff*). Sometimes these bars can be coupled with other near bars to determine a changing in the score (e.g. tempo or key signature changes). Fig. 2.2 shows a list of possible combination of bars. Finally, in scores which need multiple staves, each staff couple is embraced with the accolade[2] (brace). An example from a piano score is shown in Fig. 2.3.

(a) The staff. **(b)** The ledger.

Figure 2.1: Staff elements.

[1] In some music genre, lines can be more than 5.
[2] Braces can change in style depending on the instrument.



(a) The single bar. **(b)** The double bar. **(c)** The bold double bar.

Figure 2.2: Staff elements.

2.2 Clefs

Clefs determine the *tessitura* of the staff they refer to. Hence, clefs are the leftmost objects of a staff and initialize the reading. Sometimes they can appear in the middle of the staff indicating a change in register. The piano score can contain just two types of clef – *G-clef*[3] and *F-clef*[4]. The G-clef spiral center defines the line upon which it rests as the pitch G above middle C. The F-clef line between the dots in this clef denotes F below middle C. In Fig. 2.3 are shown two example of clefs.

2.3 Notes and Rests

Notes and rests are the most common symbols of a score. Notes are signs representing the pitch and duration of a musical sound. The pitch changes with the vertical position while duration with the shape of the note. A note is made of a head, a stem and a beam and can be single or grouped together (see Fig. 2.4). A small dot near a note head increases the note value by half of its real value[5] as shown in Fig. 2.5. Rests are silence that is they take place of notes when notes have to be mute. So, there's a bijection between notes and rests as shown in Fig. 2.6.

2.4 Accidentals and key signature

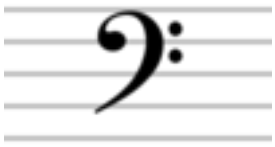
Accidentals are signs that change the pitch of the note they refer to. The *sharp* raises the pitch of a note by one semitone. The *flat* decreases the pitch of a note by one semitone. The *natural* cancels previous accidentals. *Double sharp* and *double flat* are similar to sharp and flat except that they change the pitch of two chromatic semitones. In Fig. 2.7 are listed all common accidentals for piano scores.

Key signature denotes the tonality in which the musical piece is set and is placed most of the time at the very beginning of staves (Fig. 2.8). Sometimes they are also placed in the middle of a staff and usually follow a double bar. Accidentals in the key signatures are applied to all notes in the score (until another accidental is encountered).

[3] Also treble clef.
[4] Also bass clef.
[5] There can be more than one dot next to a note.

(a) The Accolades link staffs of a grand staff.

(b) The *Treble Clef*.



(c) The *Bass Clef*.

Figure 2.3: Piano accolade and clefs.

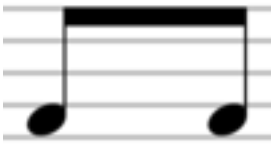


Figure 2.4: Notes can be grouped together with a beam.

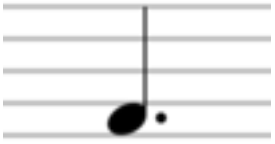


Figure 2.5: Note values changes if there's a dot next to it.

(a) Whole note/rest.

(b) Half note/rest.

(c) Quarter note/rest.

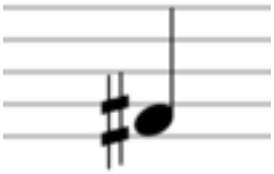
(d) Eighth note/rest.

(e) Sixteenth note/rest.

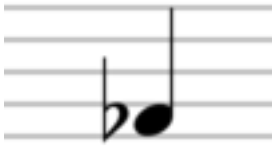
(f) Thirt-second note/rest.

(g) Sixty-fourth note/rest.

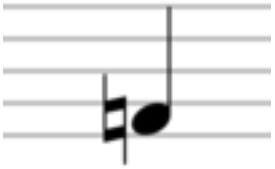
Figure 2.6: Complete set of notes and rests.



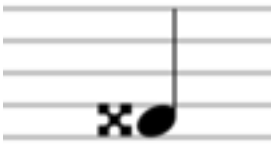
(a) The sharp.



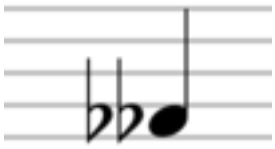
(b) The flat.



(c) The natural.

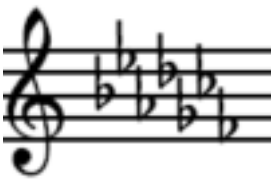


(d) The double sharp.

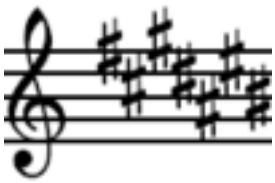


(e) The double flat.

Figure 2.7: Complete set of piano accidentals.



(a) C flat major example.



(b) C sharp major example.

Figure 2.8: Two key signature examples.

2.5 Time Signature

The time signature (TS) identifies how many beats are in measures and which note value constitutes one beat. They can be represented with numbers (numerator and denominator) or with the letter C (common time)[6]. Usually, TSs are placed at the very beginning just in between the clef and the key signature (if it exists) but in some cases a mid-score time signature is placed after a bar line to indicate a change of meter. In Fig. 2.9 are shown some common time signature notations.

2.6 Note Relationships

The basic features of music are counterpoint and harmony. The first one specifies horizontal relationships between melodic lines while the second one focuses on the vertical relationships in chords[7]. Counterpoint music notations are slurs and ties as they refer to the horizontal dynamic of notes. Slurs indicate the beginning and the end of a musical phrase, while ties interconnect two identical pitched notes that will be played as one note. Instead, harmony is based on chords that is a vertical series of notes that are played simultaneously.

Figure 2.9: Two time signature examples.

[6]There also exist the cut common time represented with a vertically cut C.

[7]Counterpoint and harmony are so close concepts that in some cases are difficult to distinguish from each other.

Chapter 3

State of The Art

OMR issue has been heavily addressed since the '60 and it is currently a provocative challenge for lots of researchers and software developers. It is a huge task to accomplish and most of the approaches decompose the problem into two main parts and eventually further subparts[1].

At a first glance staves stand as the main skeleton of the music. These groundsills consist of parallel lines all equidistant one from each other and in piano scores are coupled five by five with a brace. Actually, their dimensions (Fig. 3.1) and positions contain lots of information about the geometry of the score; for example, notes height is equal to the distance between two staff lines while the bass clef height is three times that distance. Again, a staff includes perfectly all numeric time signatures as far as height is concerned. On the other side, staves block all symbols within their length (but not height). So, once staves have been detected, their coordinates can be used to give orientations and proportions of the remaining objects on the image.

After detection follows their extraction which is a critical task. As a matter of fact, if staves are completely extracted, there is the possibility to erase pieces of other objects that are crossed by the line themselves. This could arouse significant problems on symbols recognition, depending on the exploited method; indeed, if objects are broken, they are no more the objects the system was supposed to recognize and this leads to disambiguations and further complexity. What remains on the score, after staves extraction, has then to be recognized and categorized. Some object position may be limited between the staves, while other can be located everywhere. So, position is one possible discriminating feature but it is still not enough. Dimension is another important feature, for example, all notes should have the same size and

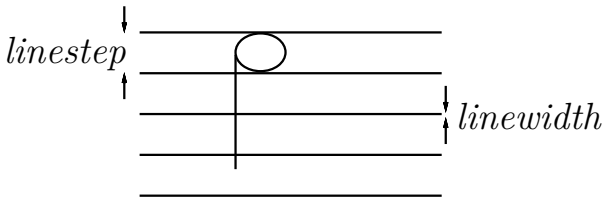


Figure 3.1: *linestep* and *linewidth* are really important through the whole process

[1]For example, value and pitch note recognition.

the same for accidentals, clefs and so on. Particular attention should be given to artifacts due to staves extraction or possible deterioration of the score.

Section 3.1 will cover the main methods used in the past to fulfill a good staves extraction while section 3.2 will go into details of the remaining object recognition issue; section 3.3 will briefly summarize the state of the art of OMR problem solving and section 3.4 will draw the landscape in OMR software developments.

3.1 Staff Line Recognition and Extraction

3.1.1 Projection Method

The projection method has been widely used in text and OMR recognition problem. It is very simple to implement but it has some shortcomings when the score is complex or too spoiled. Projection profiles are simply computed by summing up all pixels on the preferred dimension. Fig. 3.2 shows how the projection on the Y-axis discriminates staff lines in a reasonable manner, that is, each line corresponds to the longest run of black pixels. This method can be used also for vertical recognition of bar lines and stems in the same way as before, but on the X-dimension. The real problem for this approach is the complexity due to skewness and curvature artifacts that heavily affect the projection profiles. In some cases, one way to enhance staff lines detection is to use a combination of filters that erase outliers and determine proper thresholds to separate real lines from false lines. This method has been used for staff detection by Fujinaga in [7], Lee Sau Dan in [6], Kato and Inokuchi in [9], Randriamahefa in [15], Clarke in [2] and Carter-Bacon in [4].

3.1.2 Line Adjacency Graph

Line adjacency graph (LAG) is a well-known concept in graph theory. Given an undirected graph G, its line adjacency graph $L(G)$ is another graph that represents the adjacencies between edges[2] of G. So, each vertex of $L(G)$ represents an edge of G and two vertices of $L(G)$ are adjacent if and only if their corresponding edges share a common endpoint in G. In Fig. 3.3 the LAG building process is shown. Carter and Bacon approach in [4] relies on the line adjacency graph. In fact, they segment the image and each segment belongs or note to a staff line; their method has the advantage of not removing thin portions of symbols which tangentially intersect a staff line while lots of other methods do not consider this issue and remove also parts of symbols.

3.1.3 Hough Transform

The Hough transform is a well-known tool in computer vision and image processing. It allows recognition of lines and curves[3] through a voting system of objects in an image. As far as the standard Hough transform is concerned, all points in an image are projected in a parameter space in which ρ and θ are the coordinates with respect to the parametric representation of a line (see Fig. 3.4):

[2] An edge is the connecting arc between two consecutive nodes (vertices).
[3] The Hough transform has further applications also in the 3D space.

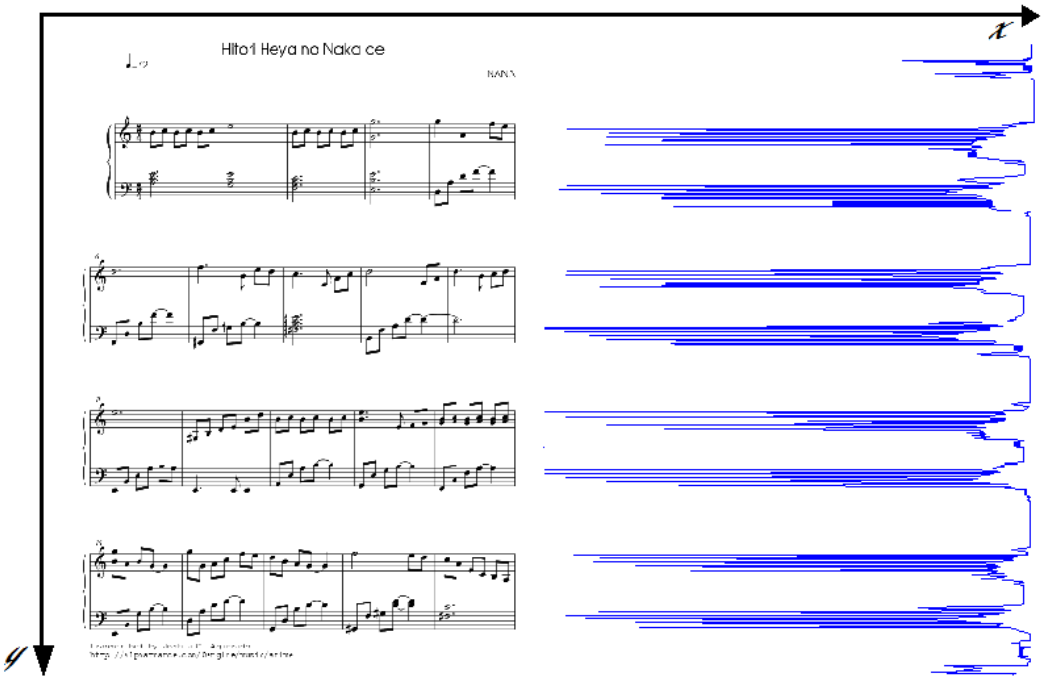
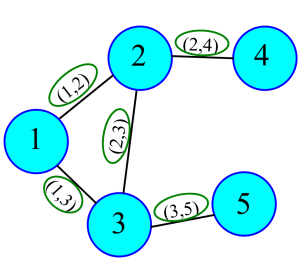
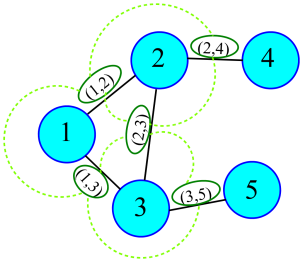


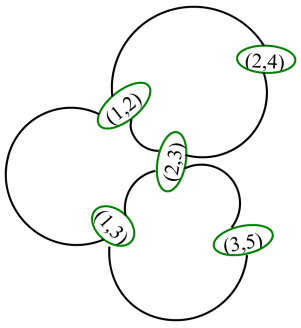
Figure 3.2: An example of projection profile on the Y-axis



(a) An undirected graph G. In blue its vertices, in green its edges



(b) Construct edges in $L(G)$



(c) The final adjacency graph $L(G)$

Figure 3.3: LAG process

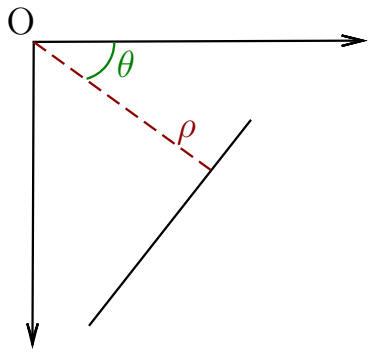


Figure 3.4: in the hough transform θ is the angle and ρ is the orthogonal distance to a line

$$y = -\left(\frac{\cos\theta}{\sin\theta}\right) + \left(\frac{\rho}{\sin\theta}\right) \tag{3.1}$$

and rearranging it into:

$$\rho(\theta) = x\cos\theta + y\sin\theta \tag{3.2}$$

with $\theta \in [0, \pi[$ and $\rho \in \mathrm{R}$ where $\rho > 0$ and R is the set of all possible distances from the origin to all lines. In this way each point (ρ, θ) in the Hough space corresponds to a line in the image. Fig. 3.5 shows a comparison between an image and itself after the Hough transformation. This tool has been used by Miyao in [12] for staves extraction; indeed, Hough transform can deal with discontinuities and inclinations but has some problem with curvatures.

3.1.4 Line Tracking

Line tracking is one of the most intuitive ways to accomplish staff line extraction after being detected. The lines are *tracked* to establish whether some of the pixels need to be removed according to some criteria. This method will, then, remove, with some approximation, pixels belonging to the staff lines while maintaining those pixels which are both part of a symbol and part of the line. Fig. 3.6 shows a simple example: Line tracking was used in [13], [1], [15] and each work uses different criteria upon pixel extraction.

3.1.5 Skeletonization

As in human anatomy, the skeleton in image processing is the substructure of a binary image. A binary image is made of *true* and *false* pixels[4] as in Fig. 3.7a: Skeletonization of such images will reduce the foreground image regions to a skeletal remnant that largely preserves the extent and connectivity of the original region while throwing away most of the original foreground pixels. It can be achieved in two main ways. First, the image is processed with some kind of morphological thinning

[4] *True* as foreground and *false* as background.

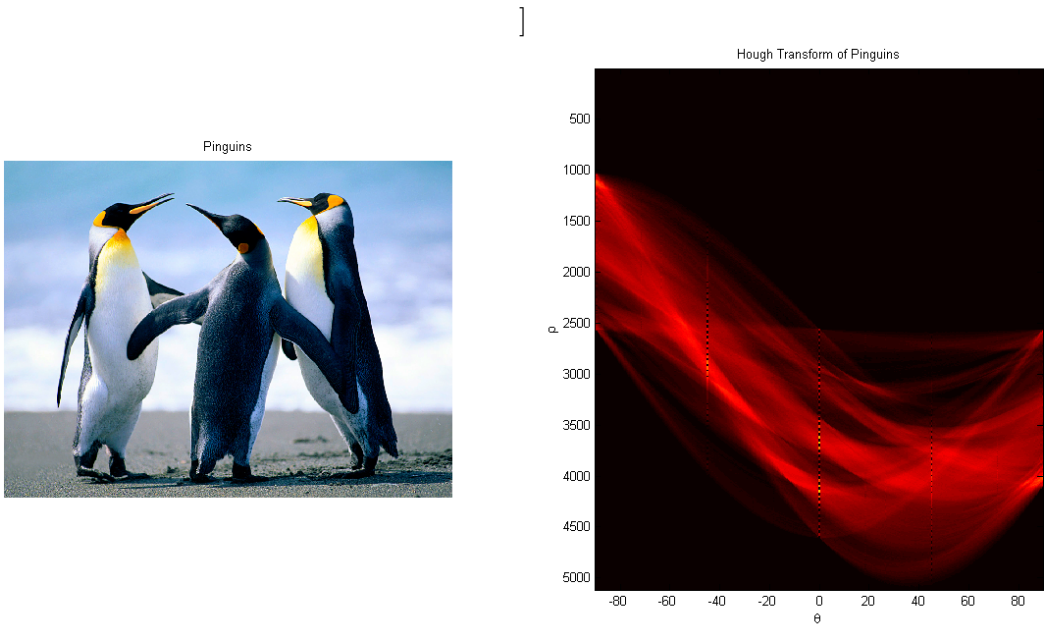


Figure 3.5: The Hough transform on a common image

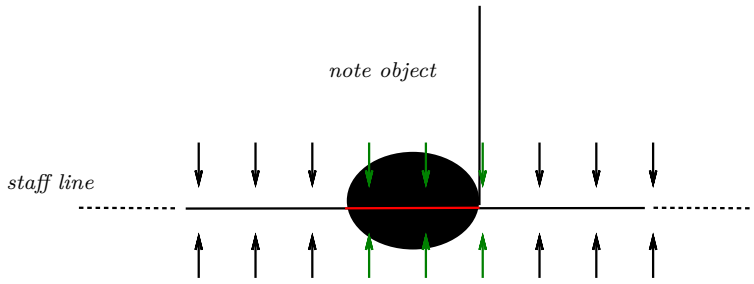
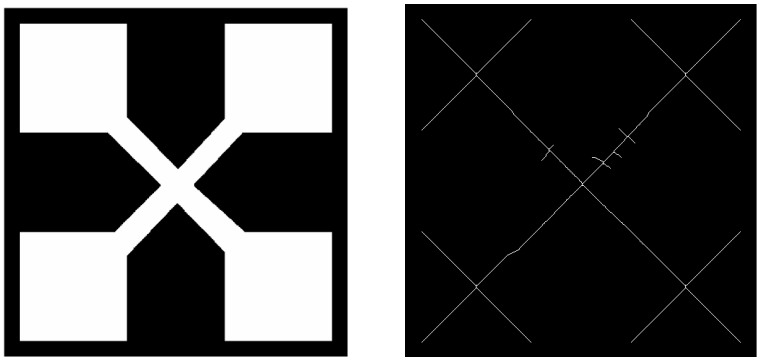


Figure 3.6: An example of line tracking. The red segment corresponds to a set of pixels which are the result of the intersection between the staff line and the note.



(a) A binary representation of a double sharp.

(b) Skeletonization of a double sharp.

Figure 3.7: Processing on a double sharp.

that erodes away pixels from the boundary (while preserving the end points of line segments) until no more thinning is possible, at which point what is left approximates the skeleton. A canonical formulation of skeletonization on a binary image is shown in Eq.3.3

$$S_n(X) = (X \ominus nB) \setminus (X \ominus nB) \circ B \tag{3.3}$$

where $\ominus$ and $\circ$ are respectively erosion and opening, X is a binary image with $X \subset \mathbb{Z}^2$, $S(X)$ is the union of the skeleton subsets and

$$nB = \underbrace{B \oplus \ldots \oplus B}_{n \text{ times}} \tag{3.4}$$

with $\oplus$ as dilation.

The second approach relies on the distance transform of the image whose result is a gray level image that looks similar to the input image, except that the gray level intensities of points inside foreground regions are changed to show the distance to the closest boundary from each point.

An important point in the skeletonization process is that it is possible to reconstruct the original image from the skeletonized one. In Fig. 3.7b follows a skeletonization of the binary image in Fig. 3.7a:

This morphological tool has been adopted in [5] where the authors explain a new method that can reconstruct staff lines thanks to the orientation of segments as output of the skeletonization.

3.2 Symbol Extraction and Taxonomy

Before starting to list all methodologies used in musical symbol recognition, there's to point out that not all approaches extract staff lines before symbols extraction. For example, Pugin in [14] uses Hidden Markov Models (HMM) to accomplish the task[5] while the famous Japanese robot Wabot-2 [10] finds symbols through a template

[5]The domain for this approach are early music prints.

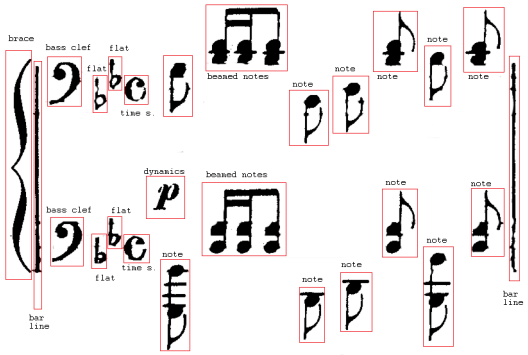


Figure 3.8: Bounded box on each object used in [3] and [4] (Rachmaninov prelude Op.23 n.5).

matching process[6]. Consider, now, symbol recognition after staff detection and removal. Indeed, template matching is one of the most immediate approaches and has been used by most of the pioneers in this field. Thus, as far as template-based approach is concerned, in [17] the matching is used to discover touching symbols in piano music scores while in [16] a fuzzy model is exploited to give more flexibility to the basic template matching, adding a certain degree of membership for the object to a certain class. On the feature-based side, [13] compares the dimensions of all connected regions to those of the staves and scout a precomputed LUT for satisfying matching conditions, [2] compares, instead, just some particular areas of selected regions in order to avoid lots of computation due to full template matching. [3] and [4] surround with bounding box all the objects and classify them with respect to their dimension and position on the score (see an example on Fig:3.8).

Other systems are based on much more sophisticated tools such as neural network (NN) in [11] that are used to extract head notes and beams in piano scores or decision tree in [8] where a 278 feature linear tree is exploited to discriminate among 5 classes of symbols. Finally, in [9] a multi-layer system is adopted; the score is explored bar by bar and each layer corresponds to a certain degree of abstraction (pixel, primitive, symbol, meaning and context).

3.2.1 Hidden Markov Model (HMM)

A Markov model is a probabilistic model that can accurately capture the effects of order dependent component failure and of changing failure rates resulting from stress or other factors. In general, Markov modeling is used to evaluate system reliability as a function of time by mapping out the states of the system - fully operational, degraded, failed - and the probability of moving from one state to another. Markov models are most useful for modeling complex behavior associated with fault-tolerant systems, degraded modes of operation, repairable systems, sequence-dependent behavior and time-varying failure rate.

The simplest Markov model is the Markov chain which satisfies the Markov

[6]It uses hardware for staff recognition and uses information gained in this phase to calibrate dimension and geometry of the score.

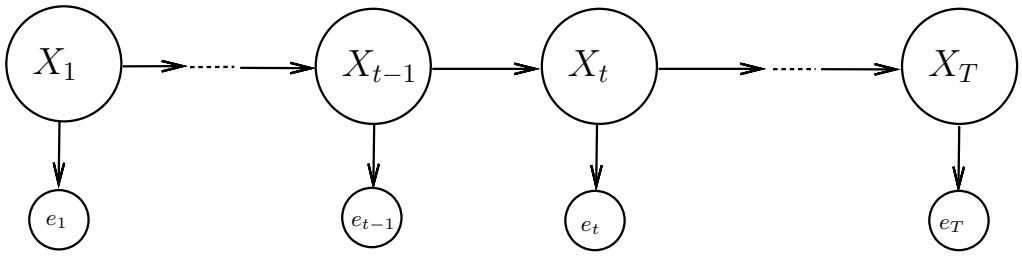


Figure 3.9: A schematic representation of a HMM.

property:

$$\mathbb{P}(X_n = x_n | X_{n-1} = x_{n-1} \dots X_0 = x_0) = \mathbb{P}(X_n = x_n | X_{n-1} = x_{n-1}) \tag{3.5}$$

this property is referred to a stochastic process X_d with discrete values on a probability space $(\Omega, \mathcal{F}, \mathbb{P})$. In a nutshell, it states that the conditional probability distribution of future states of the process, given the present state and the past states, depend only upon the present state.

Finally, a hidden Markov model[7] is a Markov chain for which the state is only partially observable. It is described by a quintuple (S, E, P, A, B):

1. $S : \{s_1, \dots, s_N\}$ are the values for the hidden states
2. $E : \{e_1, \dots, e_N\}$ are the values for the observations
3. P: probability distribution of the initial state
4. A: transition probability matrix
5. B: emission probability matrix

HMM are widely used in several fields, moving from weather casting to cryptanalysis and it was used in OMR to detect old typographical music symbols before staff removal (see [14]).

3.2.2 Feature-based template matching: contours with FFT

In order to spare computation on the matching process, it is, sometimes, useful to reduce it by selecting just a particular feature to match. Contour is a valid feature in OMR problems, but scores must not be affected to overloaded degradation. The FFT is one of the best candidates for the purpose. Immediately, notice that one aim is to obtain independence from translations and rotations. So, first operation is contour extraction:

1. Binarize images: 1 for objects and 0 for background
2. Find each object on the image and label it

[7] It is a Bayesian network.

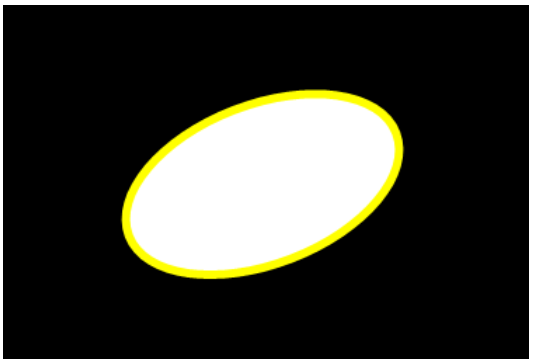


Figure 3.10: In yellow is marked the contour of a binary image.

3. For each label, extract the object contour (adopt always the same direction (clockwise or anti-clockwise))

Now, the n-th contour is characterized by two vectors $c_{x,n}$ and $c_{y,n}$, each of length N, including its coordinates. It is important that all contours must have the same number of elements $NFFT$ because when FFT will be applied, all contours will have the same period[8]. Next, obtain a complex vector representing the contour as:

$$c_n = c_{x,n} + jc_{y,n} \tag{3.6}$$

Consider $C_n = FFT(c_n)$ and its polar representation as $C(\omega) = \rho(\omega)e^{j\phi(\omega)}$[9]. The same procedure is obviously applied to the matching object and will return

$$M(\omega) = k(\omega)e^{j\theta(\omega)} \tag{3.7}$$

Normalization of the vectors is required, and henceforth $\bar{C}$ and $\bar{M}$ are the normalization of C, M where

$$\bar{\rho}(\omega) = \frac{\rho(\omega)}{\sqrt{\sum_\omega \rho(\omega)^2}} \tag{3.8}$$

and

$$\bar{k}(\omega) = \frac{k(\omega)}{\sqrt{\sum_\omega k(\omega)^2}} \tag{3.9}$$

Recalling the two important conditions:

1. Translation independence: $\frac{k(\omega)}{\rho(\omega)} = K_1\,,\ \forall\omega \in \Omega$
2. Rotation independence: $\theta(\omega) - \phi(\omega) = K_2\omega\,,\ \forall\omega \in \Omega$

Thanks to the normalization, K_2 is the only parameter to be computed. In fact,

$$\bar{M}(\omega)\bar{C}(\omega)' = \bar{\rho}(\omega)\bar{k}(\omega)e^{j(\theta(\omega)-\phi(\omega))} \tag{3.10}$$

which leads to

$$\bar{P}(\omega) = \bar{M}(\omega)\bar{C}(\omega)' = \bar{k}(\omega)^2 e^{jK_2(\omega)} \tag{3.11}$$

[8]This is possible with simple interpolation.
[9]Hereafter n is neglected.

Figure 3.11: An example of full template matching. In this case, the eye is the template (filter mask).

As a matter of fact, $\bar{P}(\omega)$ can be considered as a discrete signal whose principal component has to be computed. From Eq.3.11 can be easily derived:

$$\langle \bar{M}(\omega), \bar{C}(\omega)e^{jK_2}(\omega)\rangle_H = 1 \tag{3.12}$$

In general, $V = \mid \langle \bar{M}(\omega), \bar{C}(\omega)e^{jK_2}(\omega)\rangle_H \mid \leq 1$, where equality exists just in very lucky cases (i.e. same dimensions and rotation of $\pi/2$ multiples) and so, V is the final measure of similarity of objects with $V \in [0,1]$[10].

3.2.3 Template matching and convolution

Template matching is a costly operation that scouts the entire image looking for a target pattern, which in this case has no strong features (otherwise feature-based template matching should be applied). The most common approach uses a convolution mask (target) which is spread over the image containing the target. The convolution output will be highest where the mask best fit the image. Often, this method and the template are called respectively linear spatial filtering and filter mask. Hence, the idea is that of moving the template $T(x_t, y_t)$ on the searched image $S(x, y)$ and calculate the sum of products between the coefficients in $S(x, y)$ and $T(x_t, y_t)$ over the whole area spanned by the template. Now, considering that a pixel in the searched image has intensity $I_s(x_s, y_s)$ while a pixel in the template has intensity $I_t(x_t, y_t)$, it is possible to evaluate their absolute difference at each intensity position as:

$$AD(x_s, y_s, x_t, y_t) = \mid I_s(x_s, y_s) - I_t(x_t, y_t) \mid \tag{3.13}$$

so

$$SAD(x, y) = \sum_{i=0}^{T_{rows}} \sum_{j=0}^{T_{cols}} \text{Diff}(x + i, y + j, i, j) \tag{3.14}$$

and finally the mathematical representation of the idea about looping through the pixels in the search image as we translate the origin of the template at every pixel

[10] Maximum similarity = 1.

and take the SAD measure is the following:

$$\sum_{x=0}^{S_{rows}} \sum_{y=0}^{S_{cols}} SAD(x, y) \tag{3.15}$$

S_{rows} and S_{cols} denote the rows and the columns of the search image and T_{rows} and T_{cols} denote the rows and the columns of the template image, respectively. The lower the SAD, the higher the template score.

3.3 Brief summary

In sections 3.1 and 3.2, a list of past works has been described, emphasizing on the tools they used for the main purposes. Staff line detection and extraction are really important for the OMR problem solving in that once staff lines are flushed away from the music sheet, what remains on the score is simpler to detect and classify. But there's always a tradeoff between the applied method and results. As far as object recognition is concerned, the main tool is template matching- both feature-based and template-based. Then, recent works built up frameworks based on higher levels of abstraction, subdividing the problem into smaller (and on different levels) sub problems as in [9]. There follow Tab. 3.1 and Tab. 3.2 that summarize authors and proposed methods.

Staff detection and removal	
Techniques	*Authors*
Projection method	Fujinaga [7], Dan [6], Kato, Inokuchi [9], Randriamahefa [15], Clarke [2], Carter-Bacon [4], Bainbridge [3]
LAG	Carter-Bacon, Bainbridge [3]
Hough Transform	Miyao [12]
Line Tracking	Prerau [13], Bell [1], Randriamahefa [15]
Skeletonization	Dalitz-Fujinaga [7]

Table 3.1: Main approaches for staff recognition and removal

Object recognition	
Techniques	*Authors*
HMM	Pugin [14]
Template Matching (T-B)	Matsushima [10], Rossant [16], Toyama [17]
Template Matching (F-B)	Prerau [13], Bainbridge, Carter, Bacon [3]
Bounding Box	Bainbridge, Carter, Bacon [3]
Neural Network	Miyao [12]
Decision Tree	Homenda [8]
Multi-layer, syntax analysis	Kato [9]

Table 3.2: Main approaches for object recognition

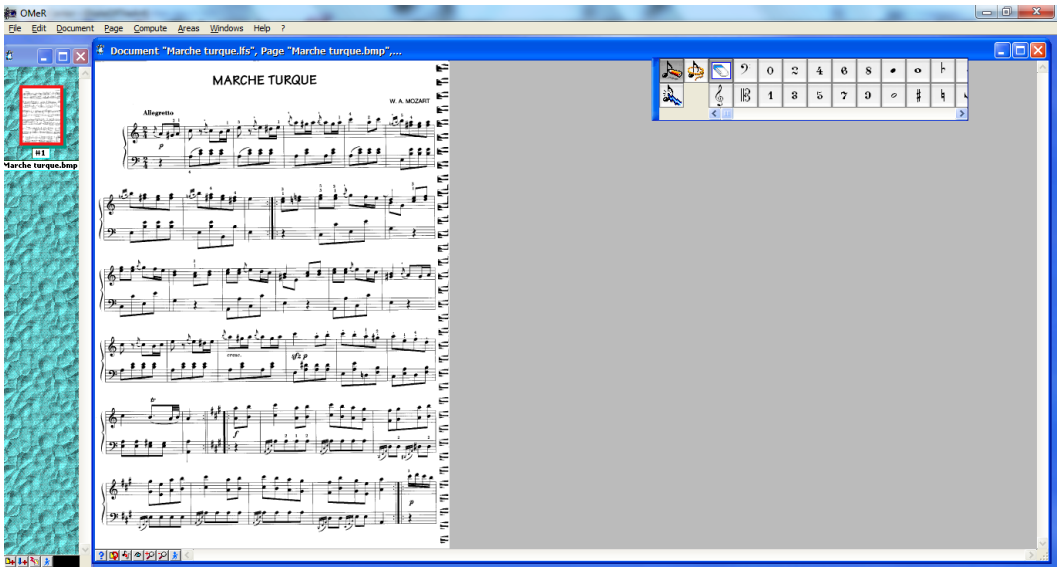


Figure 3.12: OMeR SW screenshot.

3.4 OMR Softwares

OMR issue has been heavily addressed since the '60 and it is currently a provocative challenge for lots of researchers and software developers. As a matter of fact, there exist different kinds of OMR softwares and each one gives its contribution for a good solution to the problem. First five softwares listed below are priced programs; each one has a GUI, some user friendly other not, and each one gives a set of possible actions to perform on the scanned music score. Beside the common OMR transcripition of the page, some of these SWs allow special actions such as editing, listening to the output result (MIDI format) or transposing the whole score to a certain pitch. However, there still exist some limitations due to the OMR complexity, especially when very old and corrupted music transcriptions are analyzed. It follows a list of the most famous OMR SWs:

- **OMeR** by Myriad
  From a score picture, OMeR will locate music symbols and generate a document that will be loaded automatically by Melody Assistant or Harmony Assistant which is the main program of Myriad. OMeR can manage documents made of several pages. For further information visit `http://www.myriad-online.com/en/products/omer.htm`. Fig. 3.12 is a screenshot of the program while testing a Mozart score.

- **SharpEye** by Visiv
  SharpEye Music Reader converts a scanned image of printed music into a MIDI file, a NIFF file, or a MusicXML file. It allows direct scanning from TWAIN compatible scanners. (Most scanners are TWAIN compatible). Used at its simplest, the user drags an image file into a window, clicks on a button and wait for the conversion to take place. The output is shown in conventional music notation in another window. Click on another button to save the result

as a MIDI file. SharpEye has a built in editor for correcting error of conversion. Almost all the editing can be done with the mouse and delete key. The input image window automatically scrolls to the right place in the image as you edit. SharpEye also shows warnings for each bar which doesn't make musical sense. SharpEye does not cope with handwritten music. More information on the manual at `http://visiv.co.uk/manualv2.pdf`.

- **Vivaldi Scan** by Vivaldi Studio
  Vivaldi scan recognizes symbols like beams, stems, notes, time signature, clefs, grace notes, texts, lyrics. It recognizes polyphonic voices and (MIDI) plays them back. Then, it gives access to a wide library of music symbols to modify and insert them more rapidly into the score. In addition, it gives the possibility to export the scanned scores in several formats (Vivaldi, XML, MIDI).

- **Photo Score** by Neuraton
  Photoscore is presented into two main products, a lite and an ultimate edition. The ultimate edition is capable of opening PDF files, saving MIDI files, sending scores directly to Sibelius and G7[11], saving MusicXML, NIFF and PhotoScore files for opening in Finale and other notation programs and so on. A complete list of its features is found on `http://www.neuratron.com/photoscore.htm`.

- **Smart Score** by Musitek
  Smart Score by Musitek is divided into many sub products and the Pro edition is absolutely the most complete one[12]. It is provided with notation editing, text and lyric recognition (editing), digital audio libraries (VST and AU), MIDI editing and recording. In addition, the SW can export into files for Finale, MusicXML, MIDI, WAV and CD Burning. There's an interesting demo on youtube at `http://www.youtube.com/watch?v=7pjLsQJHwdM`.

- **OpenOMR**
  OpenOMR is a 2006 open source project built on java language and whose founder is Arnaud F. Desaedeleer;

- **Gamera** Gamera is an academic system started developing in 2001 by Michael Droettboom, Karl Mac Millan and Ichiro Fujinaga at the Digital Knowledge Center of the Johns Hopkins University (USA). This last system is a toolkit written in Python (possible extensions in C++), open source coded and portable on most OS.

[11] Together with Finale, it is one of the most famous and powerful software for writing, playing, printing and publishing music notation.
[12] Pro edition is about 400$.

Chapter 4

Logic of OMRJX

OMRJX is a complex framework. It consists of blocks and each one is dedicated to a specific element in the score. In most cases, the order of such blocks is fixed and cannot be swapped; this order has been chosen with the experience, while moving inside the OMR problem and understanding that, in some cases, some processing is much more better than another one at a different step of the whole process. In a first attempt, the adopted approach tried to detect objects without staves removal through the *Viola-Jones* algorithm[1]. Unfortunately, this algorithm requires lots of time for the training phase and it can happen to wait more than a month for output results (which were all failures). Hence, the approach changed and the final system considers the removal of staves before the global recognition of all the other symbols[2]. After staves removal it would be possible also to change the skewness of the image thanks to the Hough transform that in any cases detects the lines with their inclination; if the image won't be deskewed, all the processings involving the projection method should fail. Then, it follows the recognition and removal of title, bar lines, clefs, time signatures, full note heads, full note values, rests, half and whole notes, accidentals, dots and key signature.

The full note heads and their values detection are the innovative parts of this thesis. Indeed, to my knowledge, no one has addressed such problems in the way I chose. The clustering model and the skeletonization approaches used in step *Full note heads extraction* and *Full note values* (See Fig. 4.1) are really powerful and give very good results.

In Fig.4.1 all *OMRJX* steps are sketched neglecting minor passages and in the next chapter they will be explained in detail.

[1] OpenCV functions has been studied and then implemented.

[2] This is not really true. Brace recognition precedes the staves removal as braces can alter Hough perception about lines. In addition, the title is computed and removed before because it doesn't affect the core of the page.

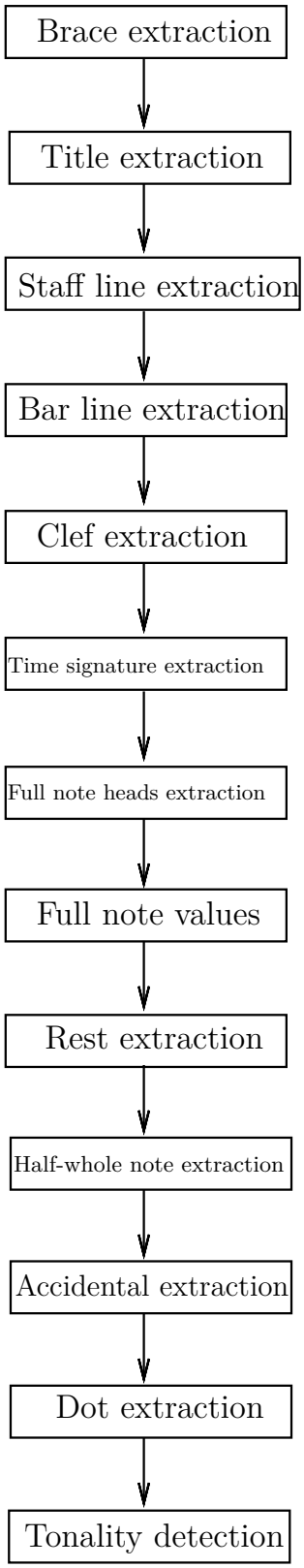


Figure 4.1: Main steps of OMRJX architecture.

Chapter 5

Deep in the OMRJX FrameWork

In the following sections, all procedures adopted in the framework will be shown and explained in detail. Each section will be provided with pseudocode and images to best illustrate the methods and obstacles for each issue. Sections are listed following the same order proposed in Chap.4 which is, by the way, a mandatory flush of instruction.

5.1 Number Of Braces

The number of braces is the first important thing to detect because it allows previous information about the content of the page. Actually, this number it's equal to the number of the biggest connected regions on the score. Pseudocode 5.1.1 describes the proposed method.

Algorithm 5.1.1: NUMBEROFBRACES(I)

comment: Compute the number of braces in a piano score

I = binary input image
$CN \leftarrow connectedregions(I)$
$RegionaArea \leftarrow Area(CN)$
Cluster RegionaArea with *k-means*
$nbrace \leftarrow$ number of biggest areas
return ($nbrace$)

Fig. 5.1 shows that, in fact, the greatest connected regions are effectively corresponding to the number of braces. These connected regions are, actually, the staves populated by all notes and symbols. Next step is the extraction of such braces.

5.2 Brace Extraction

Brace extraction is advisable before the staff lines removal. In fact, some points of the braces can be on the prosecution of a staff line as shown in Fig. 5.2. It is visible

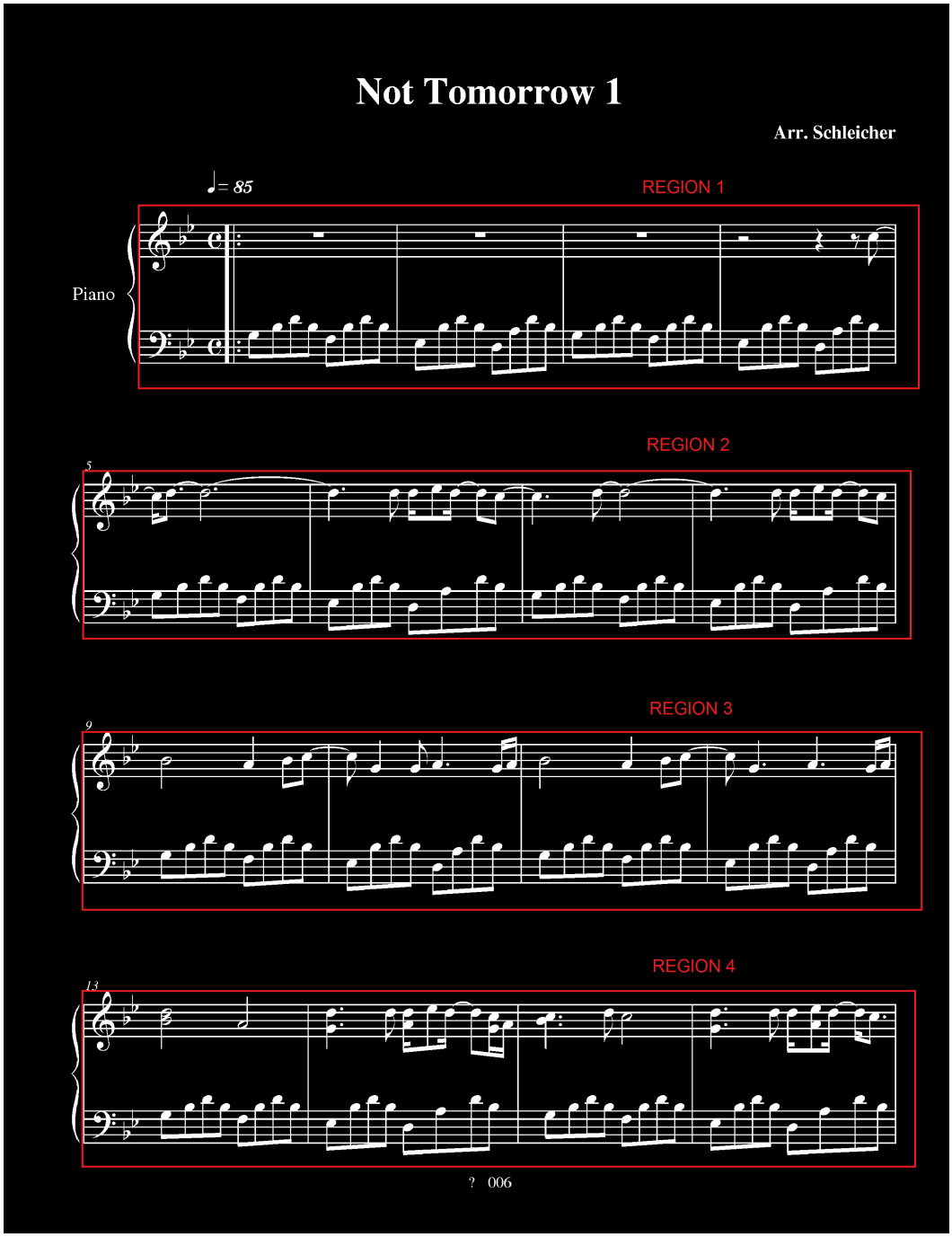


Figure 5.1: The number of braces corresponds to the number of the greatest connected regions

that the brace and the staff lines are not perfectly aligned, hence, this is a "problem" for the Hough process in the staff removal step. If not removed, those points which are not aligned will be considered as making part of the staff line, tilting the possible output line guessed by Hough. The adopted algorithm is listed in 5.2.1.

Algorithm 5.2.1: BRACEEXTRACTOR$(I, nbrace)$

comment: Extract braces

I = binary input image
$nbrace$ = total number of braces
$FFTtransbrace$ = FFT contour transform of a sample brace
$CN \leftarrow connectedregions(I)$
comment: for loop on CN is implied

$FFTcontCN$ = FFT contour transform of each CN
$Matches \leftarrow FFTcontourmatch(FFTtransbrace, FFTcontCN)$
$Brace$ = first $nbrace$ best matches
$BraceStruct \leftarrow position(Brace)$
$InoBraces = I(Brace = false)$
return $(InoBraces, BraceStruct)$

In this way, all objects on the page are compared through the FFT contour matching process.(see SubSec.3.2.2). The positions of each brace candidate are saved in *BraceStruct* and after, the input image is pruned, that is all braces pixels are set to logical $false$.

5.3 Title Extraction

Sometimes scores have a title at the top of the page or a footer at the bottom. In addition, the name of the author, the date, the arranger or the website can appear on some banners of the page[1]. Thus, all this information is associated with a number, sometimes huge, of characters and symbols. Fig. 5.3 shows an example in which there are a title, the name of the composer[2] and the website on top of the image.

At this step, the system allows the recognition of a title, if exists, and prunes the image erasing all characters belonging to this information banners. This removal is really important in that, once all banner symbols are removed, next functions will be much more light computing (i.e., less connected regions to evaluate); on the other side, in some rare cases, a note (or rest) won't be part of the largest connected region and will be considered as making part of the banner. So, the highest pixel of the highest and largest connected region (it is supposed to be a grand staff) it is used to delimitate the upper banner with the rest of the score. In this banner all connected regions are labeled and their centroids are computed. A K-nearest neighbor algorithm has been used to cluster those connected regions (letters of the title) that are close to each other and equidistant from the score margins. If a cluster

[1]Banners are always at the top or bottom of the page.
[2]There's also the Korean translation.

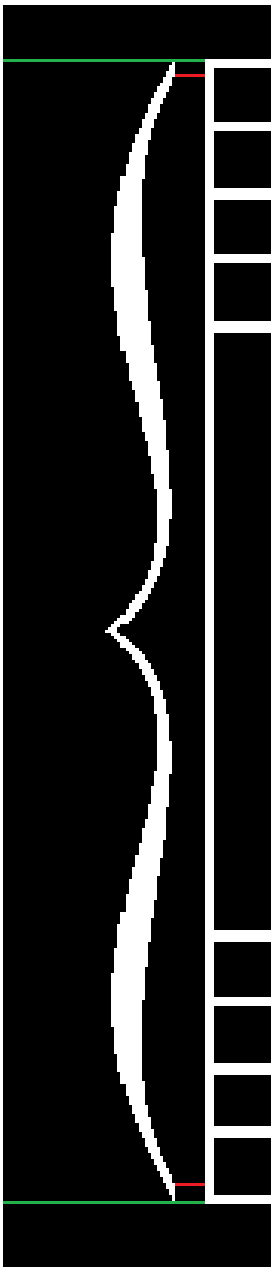


Figure 5.2: In green the upper and lower prosecution of the staff lines. In red the projection of the edges of the brace on the first bar line.

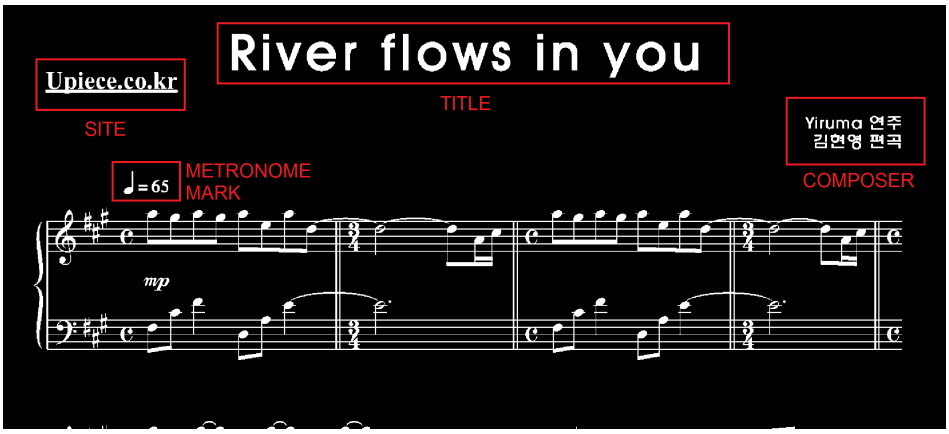


Figure 5.3: Additional information of the score can be printed besides the music itself – Title, author and website.

is found then the system will round with a box the possible title and then remove all objects from the banner. The algorithm is shown in 5.3.1.

Algorithm 5.3.1: TITLEEXTRACTOR($InoBraces$)

---

comment: Recognize and extract the title

$$\begin{cases} InoBraces = \text{binary input image without braces} \\ BigCN = \text{largest connected regions of } InoBraces \\ MaxBCN = \text{highest pixel of the highest BigCN} \\ MinBCN = \text{lowest pixel of the lowest BigCN} \\ HighBanner = InoBraces(1 : MaxBCN) \\ LowBanner = InoBraces(MinBCN : end) \end{cases}$$

comment: Set to false all lower banner pixels

$Inobraces(LowBanner = false)$
$TitleCN = connectedregions(HighBanner)$
for each $TitleCN$

$$\textbf{do} \begin{cases} \text{Compute Centroid} \\ \text{Compute Diagonal} \end{cases}$$

K-NN clustering on TitleCN
if ∃ a central cluster ∈ HighBanner

$$\textbf{then} \begin{cases} istitle \leftarrow true \\ titlebox \leftarrow position(\text{central-cluster}) \\ InoTitle = InoBraces(HighrBanner == false) \\ Ititle = HighBanner \end{cases}$$

$$\textbf{else} \begin{cases} istitle \leftarrow false \\ titlebox \leftarrow void \\ InoTitle = InoBraces(HighBanner == false) \\ Ititle = void \end{cases}$$

return $(InoTitle, istitle, Ititle, titlebox)$

(a) OMRJX result when there's a title in the image

(b) In the absence of a central title, OMRJX will save a flag $istitle = false$

Figure 5.4: Title extraction possible output.

The K-NN clustering is based on rules about dimension of the object and distance between centroids; i.e., the distance between two centroids must be lower than the sum of the diagonals of the two objects. Be c_1, c_2 rispectively the centroids of objects O_1, O_2, and d_1, d_2 their diagonals, one of the clustering condition is:

$$\sqrt{(c_1 - c_2)^2} < d_1 + d_2 \tag{5.1}$$

where c_1, c_2 are mapped in the image as (x_1, y_1) and (x_2, y_2). In Fig. 5.4 two possible examples of title extraction are shown.

5.4 Staff Lines Extraction

Staff line extraction is a critical step. Indeed, if staff lines are removed badly, next steps will be really difficult and messy. If so, parts of objects will be removed, deleting lots of useful information (pixels) and adding noise to the image. If staff lines are, instead, removed in a good way future work will be easier and will allow the use of useful tools (i.e., feature based template matching). However, staff line removal is nowadays still impossible to accomplish at best. For each adopted algorithm there will be a tradeoff to respect. OMRJX makes use of the Hough transform in such a way to surely find lines on the image. It, also, allows to detect the skewness of the score collecting all θ angles of the selected lines. On the contrary, very rough and spoiled images with curvatures[3] won't be recognized at all.

In order to improve the Hough approach, several additional operations have been done on the Hough domain. Hough domain contains votes. A threshold on votes is mandatory to select those regions in the Hough image that are associated to lines in the image domain. In Fig. 5.5 and in Fig. 5.6 are shown the Hough transform applied to a score and a detail of a maximum region. Each region of warm colours corresponds to a hypothetical line in the image domain. Each of these regions has votes which, unfortunately, are different for votes of other regions. So, thresholding has been applied imposing a $th = 0.9$ on the maximum value of the Hough domain[4]. Fig. 5.7 shows the Hough image after thresholding.

[3]Imagine to scan the score and part of the page folds when scanned.
[4]The threshold is normalized between zero and one.

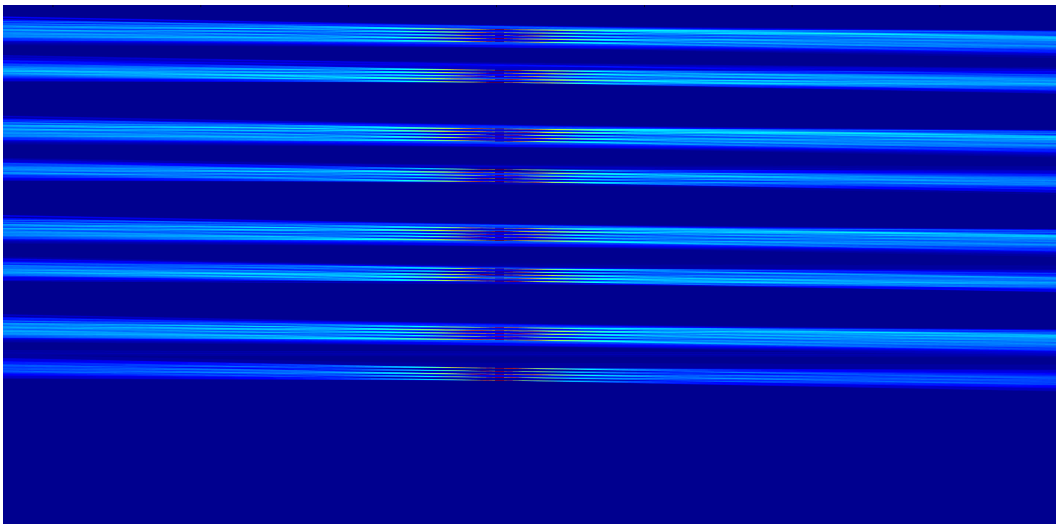


Figure 5.5: Hough transform of a piano score.

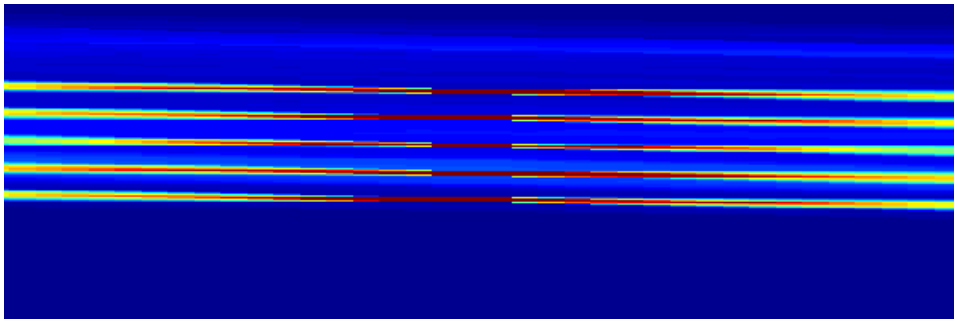


Figure 5.6: Zoom on a maximum vote region in the Hough transform. The redder, the higher the vote

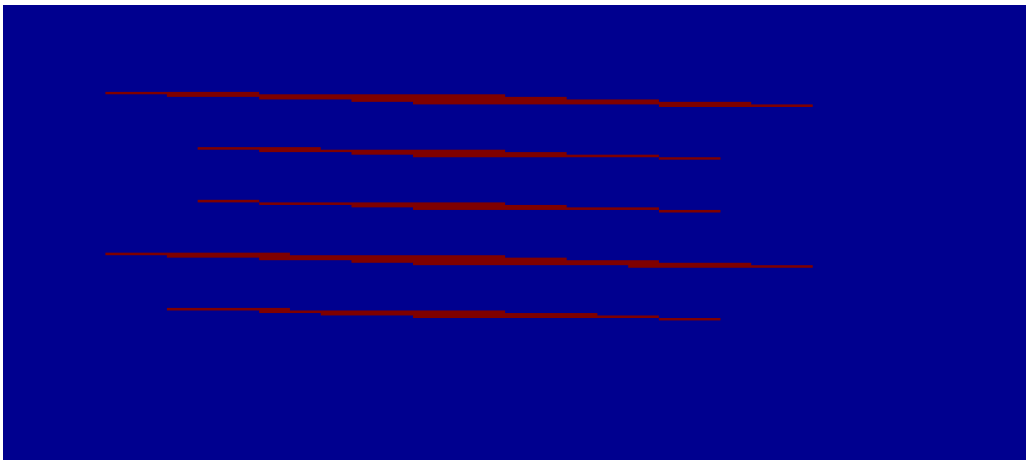


Figure 5.7: Hough maximum vote region after thresholding.

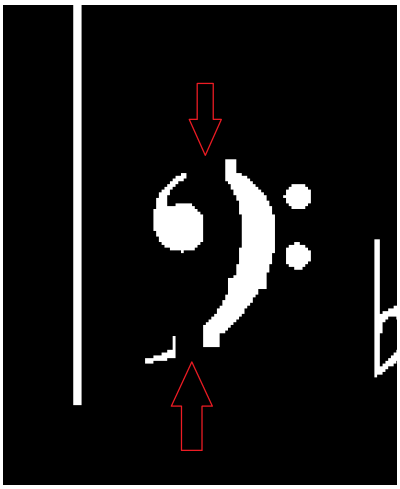


Figure 5.8: Part of the clef is tangent to the staff line and, hence, line tracking removes it.

In a sense, the thickness of the maximum regions after thresholding corresponds to the real thickness of a line on the image domain. This feature has been exploited to evaluate *linewidth*, which is one of the most important parameters used throughout the whole process, almost everywhere. *linewidth* is the width of a staff line. In order to compute its value, the projection method has been applied in the horizontal direction. The projection has been chosen for a simple reason. If the original image is corrupted, those maximum vote regions in the Hough domain won't be connected regions; this won't allow a simple measure of the height of those regions. What is sure is that those regions are very close to each other while belonging to the same line in the image. The projection fits well with this issue and a further correction is done comparing the number of connected regions of the projection of the maximum value regions with $(10 \cdot nbrace)$, where *nbrace* correspond to the number of double staff of the score[5]. If those values are not equal, the connected regions of the projections with smaller areas are discarded and considered as noise. Once *linewidth* has been computed, the function moves to the staff line removal step. Line tracking (see SubSec:3.1.4) has been adopted for such purpose, imposing an erosion window of 1.2 times the current line width. Line tracking gives good results except for those parts of objects that are tangent to the staff line. Those tangent points will be removed creating a hole in the symbol. Fig. 5.8 shows an instance of such problem that *OMRJX* won't consider. Hence, once a staff line has been deleted, points on the Hough transform referred to it are deleted too, in order to step on the next maximum region (on Hough image), hence, the next staff line to remove (on the binary image). All line positions are saved in a struct called $doublestaff$. The algorithm is summarized in 5.4.1.

[5]Just remember that a brace embraces two staffs.

Algorithm 5.4.1: STAFFEXTRACTOR($InoTitle, nbrace$)

comment: Estimate linewidth and remove staff lines

$InoTitle$ = binary input image from previous step
$nbrace$ = number of braces
H = Hough transform($InoTitle$)
Hth = 0.9 thresholding on H
$CNHth$ = connected regions on Hth
P = horizontal projection of $CNHth$
if Number of $CNHth > 10 \cdot nbrace$
then
neglect small regions
$linewidth = mean$(Area(P))
for each $CNHth$
{delete associated pixels on the image
save coordinates of the staff line in $doublestaff$
delete CNHth on Hough domain
return ($InoStaffs, doublestaff, linewidth$)

5.5 Linestep

As mentioned in Chap.3, linestep is the other relevant and required measure. This parameter, which is shown in Fig. 3.1, is pivotal in that it is used to give proportions to the musical objects. The value has been extracted from the $doublestaff$ struct, shown in Fig. 5.9 which contains all positions of the lines . In Matlab notation:

$$staffline = \begin{bmatrix} x_{1_i} & y_{1_i} & x_{1_f} & y_{1_f} \\ \vdots & \vdots & \vdots & \vdots \\ x_{5_i} & y_{5_i} & x_{5_f} & y_{5_f} \end{bmatrix}$$

where (x_i, y_i) and (x_f, y_f) are the initial and final point of a line. A psuedocode for line step evaluation is 5.5.1.

Algorithm 5.5.1: STAFFLINESTEP($doublestaff$)

comment: Estimate linestep

for each $doublestaff$
for each $singlestaff$
do Compute the mean among all linesteps
return ($linestep$)

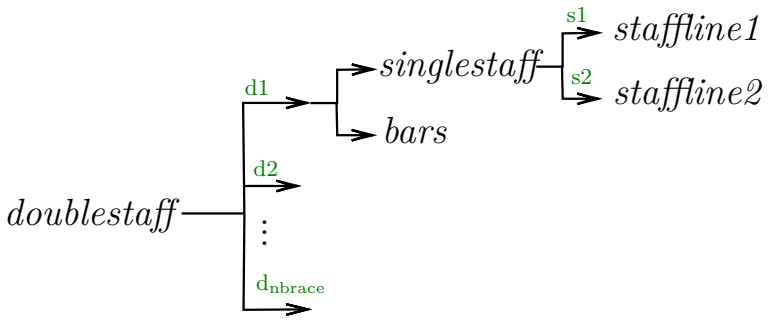


Figure 5.9: *doublestaff* structure

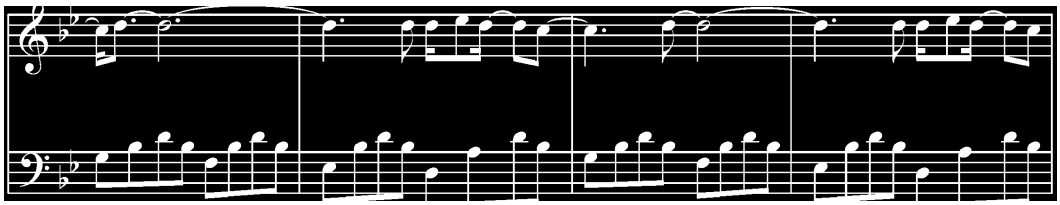


Figure 5.10: Region of the score containing the bar lines.

5.6 Hypothetical Note Area

At this point, it is possible to compute the hypothetical area of a full note. The head of a full note is elliptical but under good approximation can be considered as circular. Thus, as the head is fully contained in a *linstep*, with an additional touching *linewidth*, the area is simply

$$HypNoteArea = \frac{\pi(linestep + linewidth)^2}{4} \tag{5.2}$$

which is the canonical formula for the circle area.

5.7 Bar Line Extraction

What are bar lines? At a first glance, they are vertical segments in a wide page full of symbols. But if you zoom in, it is possible to see that these segments interconnect two single staffs that belong to the same double staff. Then, it is possible to cut out the regions in which they exist, i.e., the double staff region (see Fig. 5.10). The trick is straightforward. As in Sec.5.4 where the staff lines were detected by the Hough transform, now, bar lines can be considered as the longest lines with respect to their region. So, the procedure is the same as before with some attention about rotation of the sub image when applying Hough transform. The bar line removal function saves the coordinates of bars in the same struct *doublestaff* as the staff lines (see Fig. 5.9). In addition it is added a flag to see if the bar line is part of a double bar. A good example is shown in Fig. 5.11 and the algorithm is sketched in 5.7.1.

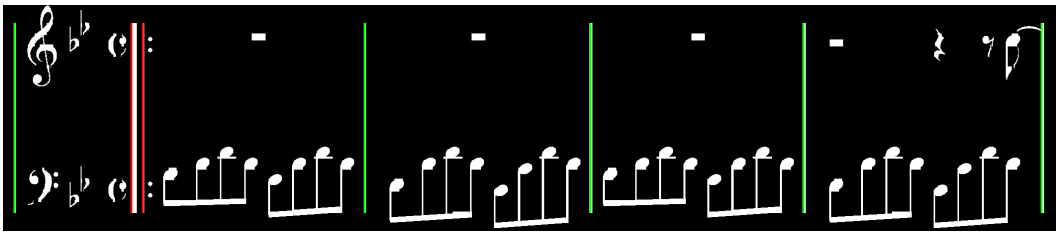


Figure 5.11: OMRJX result after bar line extraction. In red contiguous bars.

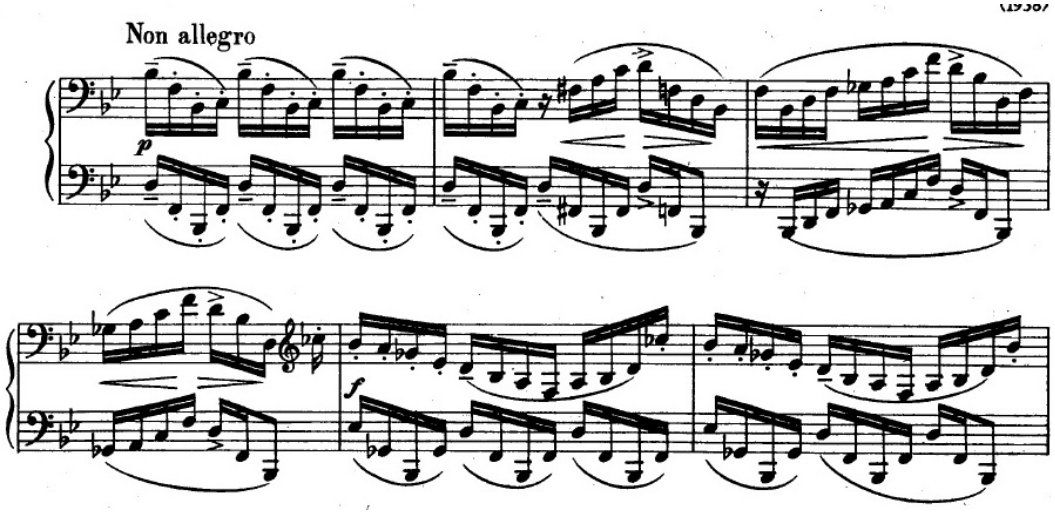


Figure 5.12: Excerpt from *Oriental Sketch* by S.Rachmaninov.

Algorithm 5.7.1: BARLINEEXTRACTOR($InoStaffs, doublestaff, linestep$)

---

comment: Save bar lines coordinates and remove them

for each *doublestaff*
crop the image
apply hough transform as for staves
save coordinates in *doublestaff*
remove the bars from the image
return $(InoBars, doublestaff)$

5.8 Clef Extraction

Clefs are a certainty of the piano score. Scanning from left to right, after the brace, there are always clefs anchored to the staves. Contrary to what most people think, the G-clef and the F-clef don't have a preferred staff. Indeed, most of the time the G-clef stays in the upper staff while the F-clef in the lower one. Fig. 5.12 shows an excerpt of a romantic piece where there are only F-clefs. Unfortunately, getting a closer look to Fig. 5.12, in the first bar of the second staff a G-clef appears in the

Figure 5.13: There exist also alphabetical time signatures.

middle of a musical passage. Clefs like that one are used in music when the hand is obliged to make a long jump on the keyboard or when particular melodic movements are woven through the lines. The OMRJX function at this step will recognize only the peripheral clefs for a faster initial computation. As clefs dimensions are strictly linked to the dimension of the staves, the clef extractor will force the system to find objects of that dimension in that particular place on the image. The tool used for this purpose is the full template matching (see SubSec.3.2.3) which applies the two resized clefs on a region blocked at the beginning of the staffs. Clefs are always detected. In 5.8.1 is listed the algorithm adopted for clefs extraction.

Algorithm 5.8.1: CLEFEXTRACTOR($InoBars, doublestaff$)

---

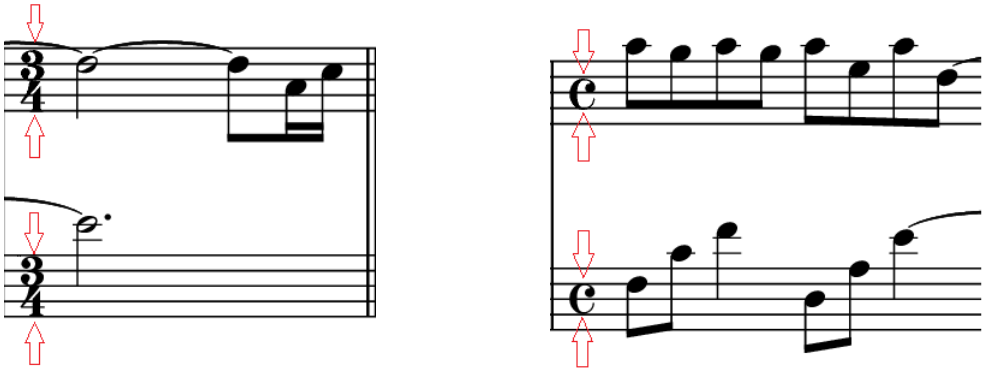
comment: Remove clefs and save their position

resize F-clef and G-clef images
for each $doublestaff$
for each $singlestaff$
- Convolve G-clef and F-clef in the beginning frame
- check on convolution results for maxima
- Associate the maximum to its clef
- save and remove

return $(InoClefs, gstruct, fstruct)$

5.9 Time Signature Extraction

Time signature behavior is a bit tricky, however, it is somehow predictable. A sheet music can have time signature (TS) or not. If exists, it can be also multiple. The main features of time signatures are that they have a fixed vertical position in the score and are replicated in both staffs of a same double staff (at the same horizontal position). Infact, they are always contained in a staff and never exceed its maximum height. A second observation about time signature is that it is not always a number. There are also symbols that can replace numbers such as the C time signature shown in Fig. 5.13. Thus, the problem has been split into two parts – numbers and C extraction. The main reason for this split is that number's vertical position in the staff is different from the C one, as shown in Fig. 5.14. For both notations has been adopted the same tool, again full template matching. As TS are confined to the staffs, the template will be reduced to these regions, avoiding useless computations outside their lattice. In addition, the system has been reinforced with rules such as the mirroring position of the TS in the coupled staves and some heuristics about the



(a) Position of a number time signature

(b) Position of a C time signature

Figure 5.14: Time signature positions are different.

frequency of TS. For number TS, the problem is divided, in turn, into numerator and denominator TS. The staff lines are so convolved with the number-C template, which is first resized as:

$$TS_{height} = 2 \cdot linestep \tag{5.3}$$

Once a convolution maximum is found, it is immediately scouted the mirrored part; i.e., if the maximum refers to a numerator number TS, it is searched the numerator region of the other staff to convalidate the convolution result. C TS procedure is much more simpler as there are just two coupled objects to detect, compared to the two by two objects for numbers. Basically, the principle is the same. The C template is moved between the second and the third space of the staff along its horizontal direction. For the sake of clarity, it follows an example for number TS detection. Imagine the situation described in Fig. 5.15 in which there are near sequential changes of TS (typical of Bartok or Stravinsky composition style). At first step the system will pick up a number from the denominator possibilities. Remember that some numbers never appear as denominator (e.g., a 3)[6]. The first denominator used as the template is 4 and it is convolved in the lower staff within its lower part. The convolution will give negative results as there are no 4 numbers as TS denominator. Next, an 8 will be rejected too. Picking up a 16, the system will found five maxima. The first move of the system is the horizontal position checking[7]. In fact, all TS, always, appear nearly after a bar line. If the maximum is far from a bar line it is, then, rejected. In this case all maxima are kept as possible denominator TS. Next step is to apply convolution in the upper staff, in its lower part, at the same horizontal coordinate. If convolution gives a maximum value then the denominator 16 is accepted as a real denominator TS. Moving on, a numerator TS has to be combined with the 16 denominator. Now, for each denominator is associated a set of possible numerators[8]. Those numerators are picked up and are convolved on the real denominator position. if there's a maximum, the numerator is selected as real

[6]This is not really true. Contemporary music is crowded with unusual TSs. For contemporary music the approach will be completely different.

[7]Optimization choice in order not to add convolution computations on failing positions.

[8]For example, a 8 is often coupled with $3, 5, 6, 7, 9$.

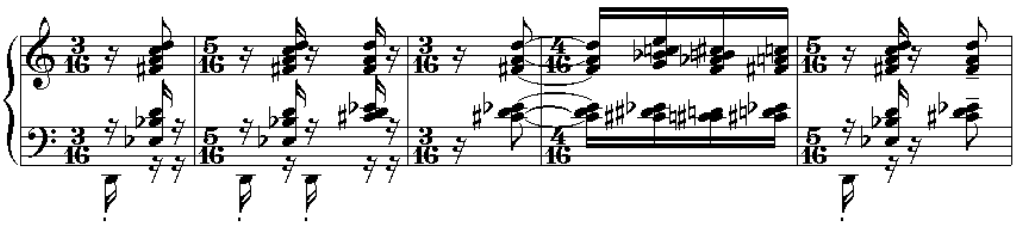


Figure 5.15: Frequent time signature changes.

numerator. Thus, all TS are saved in a struct with their values and their positions.

The fact that Ts values are stored is very important. In a sense, this is a first step toward giving semantics to low level pixels. In the future, this information will be useful to double-check, for example, if computed note values are correct or not, and in some way fix the misunderstanding. It follows two brief pseudocodes for number 5.9.1 and C 5.9.2 extraction.

Algorithm 5.9.1: TSNEXTRACTION($InoClefs, doublestaff, linestep$)

---

comment: TS number recognition and extraction

resize all possible number templates
for each $doublestaff$
for each $singlestaff$
{ subdivide the single staff in lower part and upper part
pick up a number from possible denominators
convolve the template in the lower region
if a maximum is found
then { check position
if good position
check in the lower part of the coupled singlestaff
if a maximum is found with the same template
store it as real denominator
Pick up possible numerator of such denominator
convolve until the maximum is found
save numerator and denominator
remove the time signature from the image
return $(InoTSN, FinalNumberTS)$

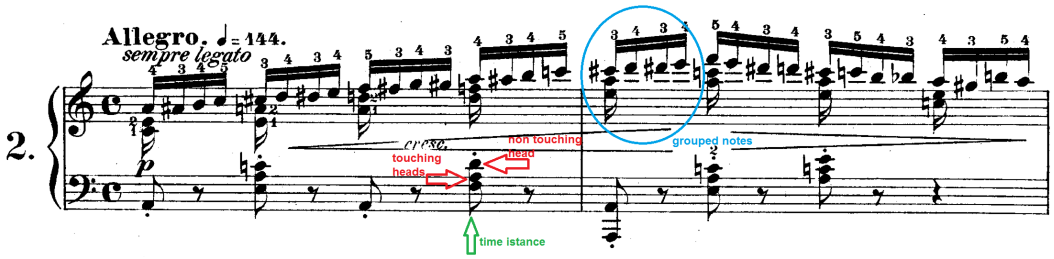


Figure 5.16: In piano scores, full heads are often in complex pattern. Excerpt from Chopin Etude Op.10 n.2

Algorithm 5.9.2: TSCEXTRACTION($InoTSN, doublestaff, linestep$)

---

comment: TS C recognition and extraction

resize all possible C templates
for each $doublestaff$
for each $singlestaff$
{ pick up a number from possible denominators
convolve the template in the central region
if a maximum is found
then { check position
if good position, check in the coupled singlestaff
if a maximum is found with the same template
store it
save position and name
remove the time signature from the image
return ($InoTSC, FinalCTS$)

5.10 Full Note Heads Extraction

Notes with full head are crucial objects on the score. They are the most common one and the most position free. In a common score for violin it won't be much difficult to find and extract them because of the monophonic nature of such instrument. In a piano score, things are slightly different as pianos can handle multiple voices at a time. So, it is possible to find head notes at the same vertical position of the same staff at different far pitches or head notes grouped in chords, really close to each other[9]. Moreover, in a same chord there can be a single head with a close tangent united group (see Fig. 5.16). Unluckily, due to degradation, full heads are most of the time corrupted and then of different size and shape.

Note head detection is really complex till heads are linked to their stem. Erosion is the chosen tool for the solution to this problem. If the image is eroded with a sufficiently big structuring element, all stems and remaining thin segments will be

[9]In contemporary music they are also grouped in clusters. This issue would be treated differently.

Figure 5.17: An example of erosion and dilation in the image. No more stems.

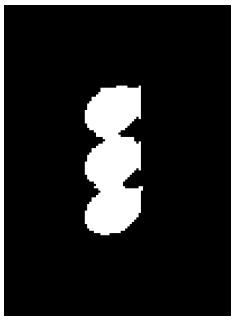


Figure 5.18: *NumNoteChord* = 3.

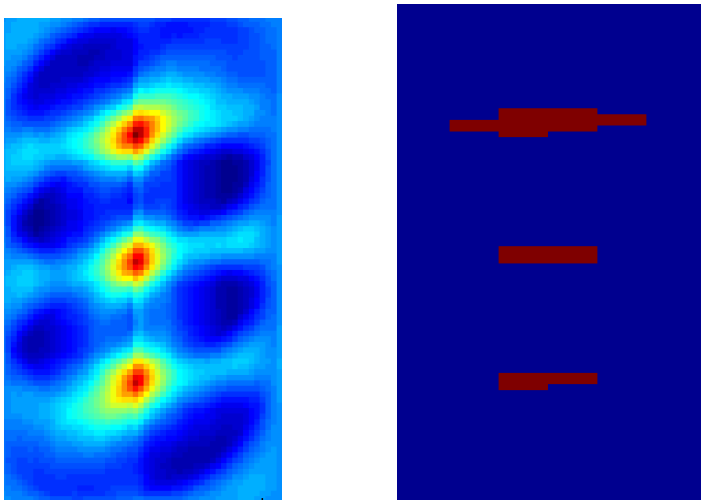
erased from the binary image. However, a compensating dilation[10] is requested to reconstruct parts of objects which were useful and have been removed by the erosion. At this point the process works basically on the dilated image after the erosion (see Fig. 5.17).

Then the system has to recognize full note heads. As note heads are sometimes touching to each other, the procedure adopted is iterated and self-trained. It is iterated because the two main functions seek in the image if there exist single, double-touching, three-touching and four-touching note heads[11] (this number is called *NumNoteChord* 5.18). It is self-trained in that each current step is somehow enhanced by the previous ones. The architecture is so divided into two main functions and both can be considered as filtering functions. Objects that are not full note heads must be rejected while note heads must be kept and later studied. How?

The first function starts with eliminating connected regions that exceed note head possible dimensions (remember that a head is contained in a *linestep*). If *NumNoteChord* = 1 all possible objects (after dimension check) are scoured and filtered through a contour matching process (a threshold is needed). If *NumNoteChord* > 1, the note convolver saved at step one is used to convolve on each possible connected region (*NNoteCollection*); when centroids of the maximum value of

[10]The structuring element for dilation is smaller than the erosion one. Too much dilation will cause objects to be transformed into touching elements.

[11]In more complex scores, there can exist also five-touching and higher number of such compound note heads. Since the beginning this issue has been estimated and it is possible to generalize the process to whatever number of touching symbols.



(a) Convolution with the convolver.

(b) Maximum of convolution.

Figure 5.19: Convolver control for chords.

Figure 5.20: The brighter region is a false negative example.

convolution are aligned, it means that those touching heads are a possible chord. In Fig. 5.19 are shown the convolution and its maximum values[12]. The convolution mask (the convolver) is computed as the mean of all good objects in *NNoteCollection*.

The output of function one is a collection of objects that resembles note heads but, actually, have a degree of uncertainty (e.g. Fig. 5.20). Function two implements a high profile algorithm. First filtering is about orientation of the *NNoteCollection* objects (if $NumNoteChord > 1$). Then, for each possible note head is computed a set of three features concerning the area, a convolution value and a contour matching condition. The possible heads are clustered and grouped if similar, through an ad-hoc K-NN algorithm, as shown in Fig. 5.21. As clustering may still contain false positives, a further filter is applied on note heads selection. Each possible note head is tested on its filling nature. If it's a region with a cavity, it is rejected, else it is the final choice on note head candidates. A pseudocode is shown in 5.10.1.

[12]Sometimes, this check is misleading because the accuracy decrease with the number of touching objects and valid chords are rejected.

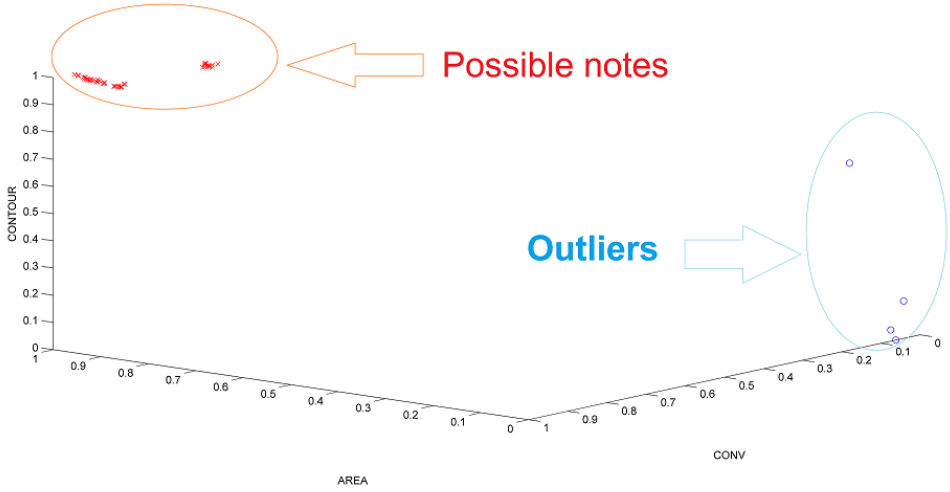


Figure 5.21: An output of the clustering process.

Algorithm 5.10.1: HEADEXTRACTION($FullNoteStruct, NumNoteChord$)

---

comment: Full note heads extraction

Remove stems
initialize FullNoteStruct
Compute connected regions
comment: NumNoteChord = number of possible touching symbols

NumNoteChord = 1,2,3,4
for each $NumNoteChord$
filter on size
if $NumNoteChord = 1$
  then contour matching
  else check on convolution maxima alignment
if $NumNoteChord > 1$
  then check orientation
area, convolution, contour features
K-NN clustering
Filling condition
return ($FullNoteStruct$)

5.11 Full Note Values Extraction

Full note values extraction is a sophisticated part of **OMRJX**. Indeed this part gives semantics to all heads extracted in Sec.5.10 and works on a higher level which is closer to the music content. A preliminary step involves the creation of a summarizing matrix, called *LabelCell*, that describes which full note heads are involved in the score. Remember that from the previous section, the *FullNoteStruct* includes n steps

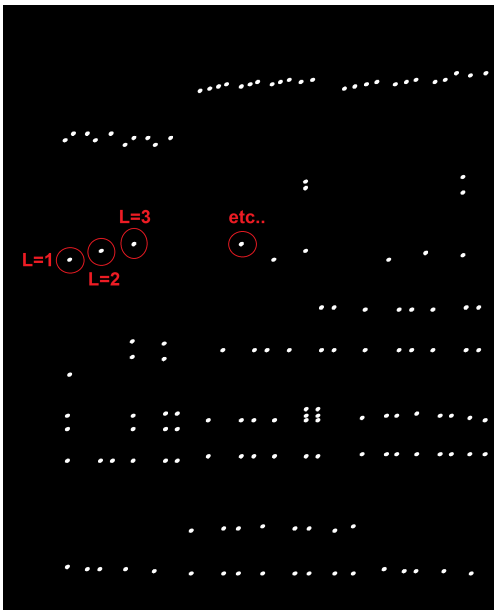


Figure 5.22: Labeled notes of step 1 in FullNoteStruct.

Label *InoTSC*	Label step1
2	[46 47 48 49 50 51]
4	[52 53]
⋮	⋮

Table 5.1: A possible example of *LabelCell* referred to the situation shown on Fig. 5.23

and each one is identified by the number of touching heads[13]. So, in each step there's a logical image that collects and labels such heads (each head corresponds to a connected region which has a certain label) (see Fig. 5.22).

In particular, *LabelCell* links each labeled note heads of each step (obviously with a number of heads greater than zero) to the last processed image in which all connected regions have been computed and labeled[14]. For better understanding in Fig. 5.23 and in table 5.1, there's a simple example of what *LabelCell* does.

At this point the *fullnoteheadvalueextractor* function comes into play. It allows to scout the labeled connected regions in InoTSC and try to assign to each head (of each step) involved in such connected region, its rhythmic value. As stated in Chap.2, notes can be alone or grouped with a beam and a relevant feature is that $\frac{1}{4}$ value notes are always not beamed notes. Moreover, notes can be included or lie outside the staff and in the second case there will be the presence of ledgers that will make the computation much more difficult.

The algorithm proposed will make use of two main tools for the values discrimination: skeletonization and projection method. Skeletonization is basically used

[13] $n = 1, 2, 3, 4 \Rightarrow$ one single head, two, three, four touching heads.
[14] Previous step image is *InoTSC*, the image after time signature extraction.

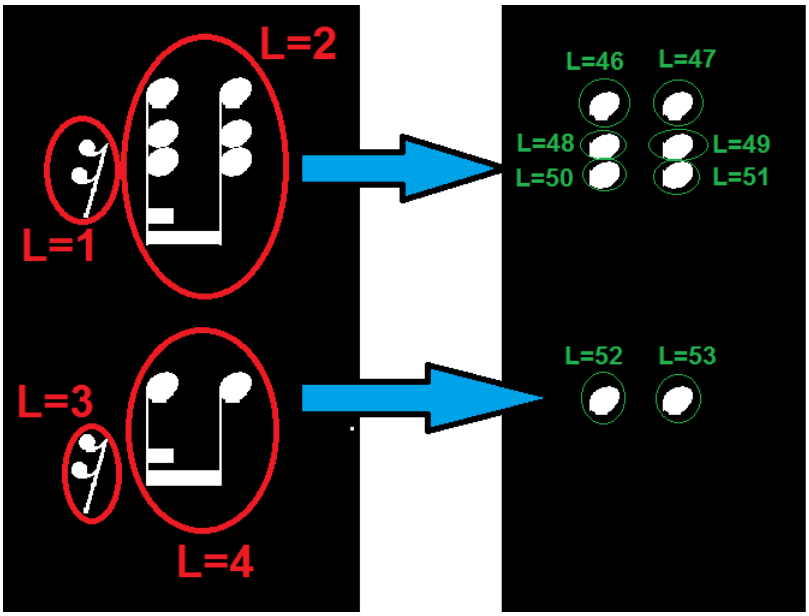


Figure 5.23: Each labeled connected region in InoTSC (red) is linked to an amount of labeled connected regions (green) in the FullNoteStruct image (in this example it is step1).

when heads (at whatever step) belong to a single stem and are not beamed. Instead, the projection method is used when heads are grouped and gathered with a beam. The process is the following. For each labeled connected region in *InoTSC* , remove heads of any steps of *FullNoteStruct* involved in such *InoTSC* connected region. On Fig. 5.24 are shown all possible results after this passage. For each situation a specific algorithm has been used. In situation A and B, the frame is cleaned of the ledgers, if exist, through erosion with a structuring element like:

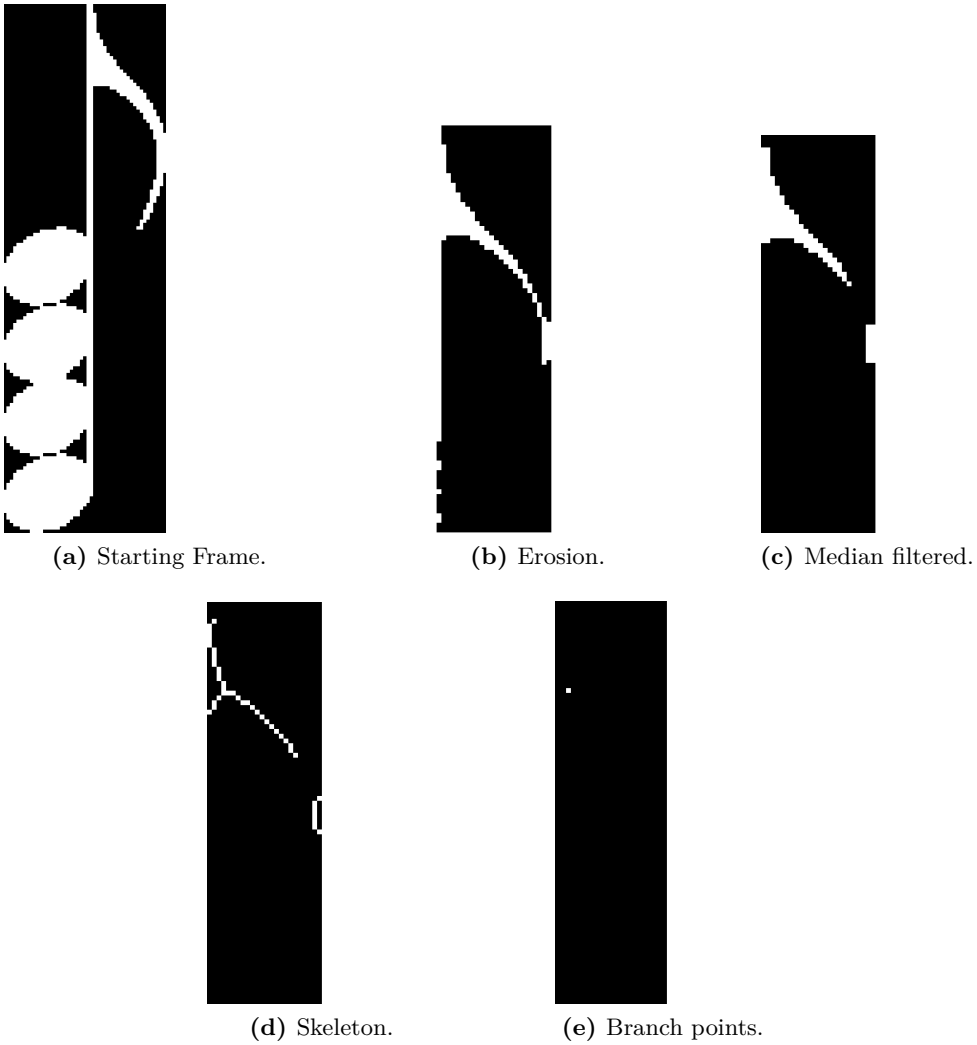
$$SE = \begin{bmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

where its length is equal to *linewidth*. After this, the frame is cropped and its width is evaluated. If it is minor than *linewidth* it falls in situation 1 and the heads of that connected region are assciated to a quarter value. Else it falls in situation B. In this situation, morphological operations have been heavily used. I.e., two consectuive erosions, median filtering and finally skeletonization. The conjunction between the stem and the beam is what characterizes the note. This portion of note contains enough pixels to undergo two consecutive erosions. The median filter is applied to smooth what remains of the note and at last, skeletonization is applied. Branch points after skeletonization correspond to the final value of the note[15]. In Fig. 5.25 is shown the process.

[15] 1 branch point $\Rightarrow \frac{1}{8}$ (♪), 2 branch points $\Rightarrow \frac{1}{16}$ (♬), and so on.

(a) Situation A. **(b)** Situation B. **(c)** Situation C.

Figure 5.24: Possible situations after heads removal.



(a) Starting Frame. **(b)** Erosion. **(c)** Median filtered.

(d) Skeleton. **(e)** Branch points.

Figure 5.25: Value extraction after skeletonization.

Figure 5.26: Just the beam. The small central and vertical hole on the beam is due to the stem removal.

As far as situation C is concerned, the projection method is the key. First, ledgers and stems are removed[16] in order to let just the beam in the frame as shown in Fig. 5.26. Then, the vertical projection of such frame is performed and finally, each centroid head neighbourhood is evaluated on the projection; the maximum value of that neighbourhood will correspond to the value of the head note. The following pseudocode broadly summarizes the whole process after the creation of *LabelCell*.

Algorithm 5.11.1: VALUEEXTRACTION($InoTSC, FullNoteStruct, LCell$)

comment: Full note heads value extraction

for each connected region in InoTSC
  do Detect situation
if situation A
  then { note heads correspond to a quaver
  else if situation B
  then { two consecutive erosions
           median filter
           skeletonization
           count branch points
  else { detect beam orientation (up, down)
           remove stems and ledgers
           projection on the beam
           for each head scout the neighbourhood
           assign a value to each head of the group
return ($FullNoteStruct$)

5.12 Full Note Remover

After values extraction, the image is cleaned and all full head notes must be removed. This is a simple operation because in *LabellCell* are stored all labels related to this

[16] Before stem removal, a localization process is done to recognize whether the stems are upturned or downturned

Figure 5.27: An output image after full notes removal.

notes. At this step the remaining image (*InoFullNotes*) should have just a few objects on it. A possible *InoFullNotes* is shown in Fig. 5.27.

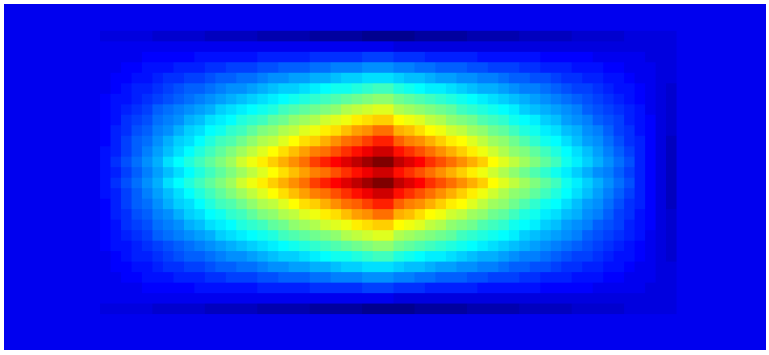
5.13 Rest Extraction

From now on, the image should be pruned enough to allow only the use of full template matching. So, all possible rests are stored and resized as in Tab. 5.2 to fit the image proportions. Then, template matching is applied[17] taking particular care to whole-half rests. In fact, when convolution is executed with such template, after imposing a threshold, each rest of this type may generates two close maxima (see Fig. 5.28). So, a dilation is applied on the convolution image after thresholding to obtain just one maximum for each whole-half rest. Then, rests are localized on the

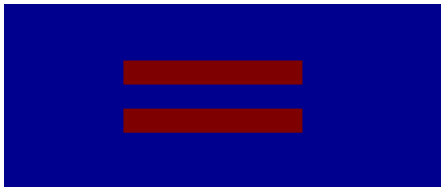
[17]Since some rests are contained in other, the order adopted is sixty-fourth, thirty-second, sixteenth, eighth, quarter and whole-half.

Rests	Scale Factor
whole-half	(linestep/2)+(linewidth/2)
quarter	3·linestep
eighth	2·linestep-linewidth/2
sixteenth	3·linestep
thirty-second	4·linstep
sixty-fourth	5·linestep

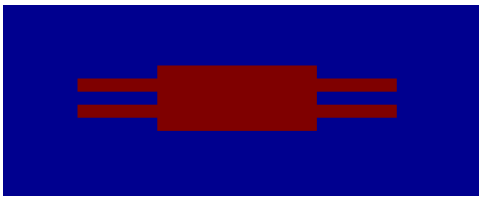
Table 5.2: Scale factors for each rest.



(a) Convolution result.



(b) Convolution after thresholding.



(c) After dilation the rest has just one maxima.

Figure 5.28: Process for detecting whole-half rests.

imaged and, after saving their position, removed.

Since now, whole-half rests have been saved in the same structure because, as a matter of fact, they are identical. A further function is built to distinguish whole to half rests according to their position or to the ledger on which they lie. If the rest is in the staff, its centroid distances to the third and fourth staff lines are computed and if closer to the fourth it is a whole rest else it is a half rest. Instead if the rest lies outside the staff, its extrema are computed. If the upper edge is longer than the lower, it means that it is a whole rest, else it is a half rest (see Fig. 5.29). Pseudocode 5.13.1 describes the rest extraction algorithm.

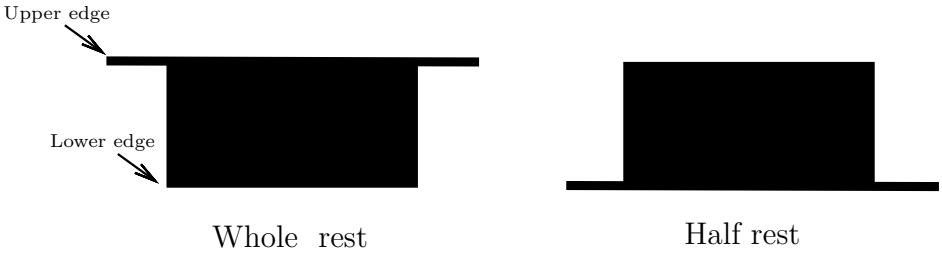


Figure 5.29: The difference between whole-half rests outside a staff.

Algorithm 5.13.1: RESTEXTRACTION($InoFullNotes, linewidth, linestep$)

comment: Rest extraction
for each rest template
  do Resize it
convolve with InoFullNotes image
find maxima after thresholding
if whole-half rest
  then { dilation on the thresholded convolution
save position of each rest
erase from the image
discriminate whole-half rests
return ($RestStruct, InoRests$)

5.14 Half Notes Extraction

Half notes are quite difficult to discover. Indeed, their void circular shape recalls lots of objects of the scores such as small ties, slurs, fermata or the coda sign. In addition, their contour is broken most of the time. For this reason it is quite hard to tune a thresholding value for template matching. However, template matching still works and an algorithm has been developed to remove not only the head, which is the template, but also the stem related to it. This algorithm ensures not only the stem removal but also the removal of distant and detached parts of the head. First, maxima of convolution are found and then their centroids are computed. Each maximum corresponds to a head note but, as stated before, heads can shows discontinuities on the contour. So, a neighbourhood of such centroid is extracted and an iterative morphological closing, with an increasing structural element, is applied until only one connected region remains in such frame as shown in Fig. 5.30. After the closing process there will appear lots of new pixels but those pixels are not redundant and are useful to best erase detached part of the note. At the end, the head notes and the stems are saved and completely removed. Pseudocode5.14.1 recalls the half note extraction process.

(a) Frame near the centroid of a convolution maximum.

(b) After iterative closing the discontinuities are reconnected.

Figure 5.30: Closing process on a half note.

Algorithm 5.14.1: HALFNOTEEXTRACTION($InoRests, linestep$)

comment: Half note extraction

load half head note template
Resize it
convolve with InoRests image
find maxima after thresholding
for each centroid of previous maxima
do { extract a frame from InoRests sorrounding the centroid
compute connected regions of the frame
morphological closing until
number of connected region equal to 1
save position of each half note
erase from the image
return $(HalfStruct, InoHalfs)$

5.15 Whole Notes Extraction

At a first glance whole notes are similar to half note heads. Instead, they're much more compact and contain more pixels. Hence, they're much closer to full heads. For this reason, whole notes are seldom broken and often intact. So, full template matching is applied as previously and the convolution result is thresholded to an empirical value that discriminate such notes. In pseudocode 5.15.1 it is described

(a) Parts of a flat that are disconnected.

(b) After iterative closing the discontinuities are reconnected.

Figure 5.31: Closing process on a flat.

the algorithm for whole note extraction.

Algorithm 5.15.1: WHOLENOTEEXTRACTION($InoHalfs, linestep$)

comment: Whole note extraction

load whole head note template
Resize it
convolve with InoHalfs image
find maxima after thresholding
for each centroid of previous maxima
  do save position of each whole note
erase from the image
return ($WholeStruct, InoWhole$)

5.16 Accidentals Extraction

All accidentals are treated in the same way except flats. In fact flats, as half notes, present discontinuities in the shape due to a non-perfect cleaning of the staves. This accidental is recognized through convolution and then it undergoes subsequent morphological closings to ensure the reconnection of disconnected regions of the same object (identical process used in 5.14). Instead, other accidentals are processed and recognized with the common algorithm of full template matching. In Fig. 5.31 is shown a detail of the closing process upon a flat in the score. Finally, the algorithm is presented in pseudocode5.16.1.

Algorithm 5.16.1: ACCIDENTALEXTRACTION($InoWhole, linestep$)

comment: Accidentals extraction

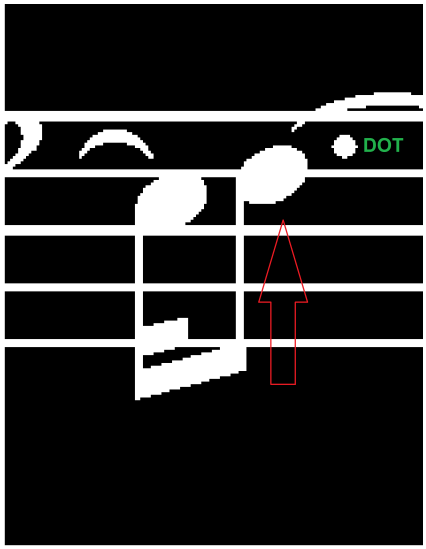
for each *accidental*
do
- load accidental template
- Resize it
- convolve with InoWhole image
- find maxima after thresholding
- **for each** centroid of previous maxima **do**
- **if** $accidental == flat$ **then**
  - extract a frame sorrounding the centroid
  - compute connected regions of the frame
  - morphological closing until
  - number of connected region equal to 1
- save position of each accidental
- erase from the image

return ($AccidentalStruct, InoAccidental$)

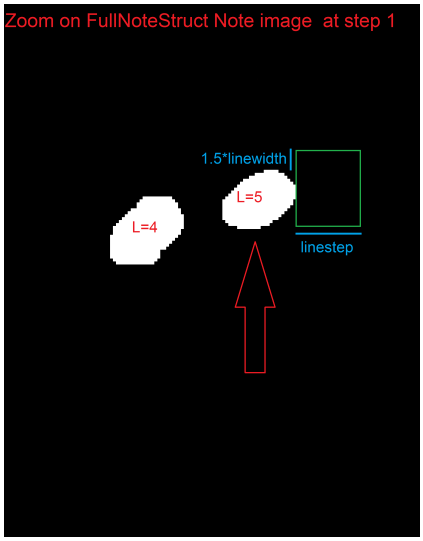
5.17 Dot Value Extraction

This function will find dots next to each head of each step in *FullNoteStruct*. Although dots can also be next to half, whole notes and rests, the function will find just those dots related to full head notes. However, the process adopted for full heads is also fully compatible with void notes and rests. It consists of localizing each head, extracting a portion next to it[18] and convolving such frame with a dot template. If after thresholding there exists at least a maximum, the full note head value is dotted, hence its original value is augmented of half of it. In Fig. 5.32 is shown an example of the process while the algorithm is presented in pseudocode 5.17.1.

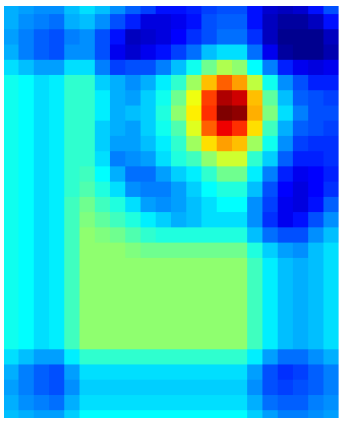
[18] Find the right-upper and right-lower extrema and move to the right of linestep and move up of $1.5 \cdot linewidth$



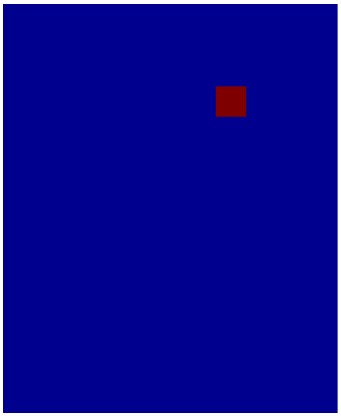
(a) Zoom on the original image.



(b) Zoom on FullNoteStruct Note image.



(c) Convolution on the frame of InoAccidental.



(d) One maximum equals to a dot.

Figure 5.32: Process for detecting dots next to full note heads.

Algorithm 5.17.1: DOTEXTRACTION($FullNStruct, InoAcc, lwidth, lstep$)

```
comment: Dots extraction
load dot template
resize it
for each step in FullNoteStruct
do { for each connected region
     do { Extract a frame next to the region in InoAccidental
          convolve with the dot template
          Thresholding on the convolution
          if atleastamaximumexists
            then That note has a dot
          update FullNoteStruct values
return (AccidentalStruct, InoAccidental)
```

5.18 Key Signature Detection

Key signature is a series of sharps and flats at the very beginning of the staff that indicates the tonality of the piece. This algorithm allows the recognition of such accidentals (if they exist) and will guess the possible main tonality of such music page. So, each first measure of each staff is searched in its first half (in Fig. 5.33 is shown such region) and the system looks for flats and sharps. If no accidentals are found it means that the tonalities are *C Major* or *A Minor*[19]. If some accidental is found, the distance between them is computed and evaluated with respect to *linestep*. In fact, accidentals in the key signature are one *linestep* far from each other as shown in Fig. 5.33. The resulting number of accidentals that satisfies the constraints are then associated to a specific tonality. However, to improve the guess on tonality it is necessary the extraction of note pitches and then the harmony evaluation but, unfortunately, this functions haven't yet been implemented[20]. Finally, after guessing the tonality, the system will allow the user to listen to a brief recording in the detected tonality. Pseudocode 5.18.1 recalls the adopted algorithm.

[19] A major and a minor tonality are associated for each key signature.
[20] Harmony detection is still an open issue.

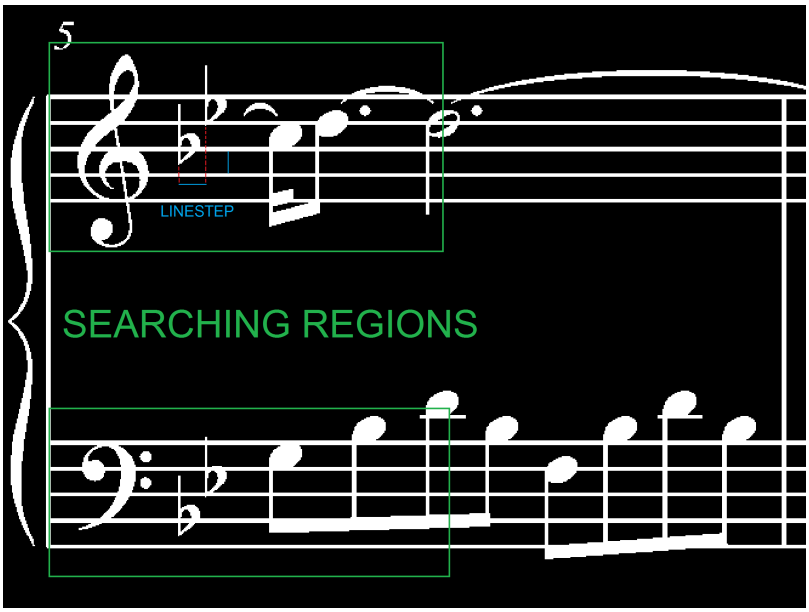


Figure 5.33: The region in which the key signature is searched.

Algorithm 5.18.1: GUESSTONALITY($doublestaff, FlatStruct, SharpStruct$)

---

comment: Tonality extraction

extract the first measure of each staff
take the first half
look for accidentals
if no accidentals
  then Tonality = C major or A minor
  else if 1 accidental
  then Tonality = F major-D minor or G major-E minor
  else if accidentals distance $<$ *linestep*
Tonality is detected
listen to an example

Chapter 6

Experimental Results

The evaluation of an OMR system is quite difficult as there are lots of variables involved. The approach proposed to test *OMRJX* is that of subdividing the symbol recognition by classes in order to have localized results that allow a precise understanding and correction of errors. Hence, the test results will be divided as follows:

1. **Structure**: all structural elements such as accolades, title, staves, bars, clefs and time signatures .
2. **Full note heads**: recognition of all full heads (single or touching heads).
3. **Void heads**: recognition of half and whole notes.
4. **Note values**: detection of values for full head notes such as quarter, eighth, and so on.
5. **Rests**: recognition of all types of rests.
6. **Accidentals**: detection of flats, sharps and naturals (double flats and double sharps have not been tested).
7. **Tonality check**: final guess about the tonality of the piece.

The tests have been conducted on 10 music sheets as output of notation softwares[1]. The quality is set at 300 DPI and there are no physical degradations. Scores contain a variety of rhythms and symbols in such a way to test most of the symbols that *OMRJX* can recognize. In addition, among all scores there are some with different fonts. Tests were carried out without changing the system parameters in order to have much more fair evaluation of *OMRJX*. Finally, there's a challenge test in which the system faces the commercial products *OMeR* and *Smart Score* (see Sec.3.4) on a score at 150 DPI and another one at 300 DPI.

Each of the following sections will present difficulties and strength points encountered for each analyzed score and the test results are gathered in Tab. 6.1 reminding the validation formulas:

$$Precision = \frac{tp}{tp + fp} \tag{6.1}$$

[1]All scores have been downloaded from the web and the source notation software is unknown.

$$Recall = \frac{tp}{tp + fn} \tag{6.2}$$

where tp, fp and fn stand for true positive, false positive and false negative.

6.1 Test 1

This sheet is not crowded with too many symbols and the most of them are eighth notes. The structural recognition is perfect with a 100% accuracy. Full note values are perfectly recognized too. Some difficulties have been encountered with half note heads detecting just 15/26 of them and giving 3 false positives. There are no rests but no false positives have been detected. All sharps have been recognized but 2 false positives came out from flats detection. Finally, thanks to the good sharp recognition, the tonality has been correctly detected.

Half heads are well recognized when found alone on a stem while are badly detected when belonging to a chord as shown in Fig. 6.1. Instead, flat's false positives come from a small vertical segment which is probably an unclean part of a previously erased object.

6.2 Test 2

This score contains much more objects with respect to test 1 music sheet. There are lots of time signature changes in the first six measures and the number of sharps is increased with a factor 4. There are still no rests but the number of full heads is doubled. The structural part is successfully completed with a 100% accuracy. Then, full notes are fully detected in each step but there are lots of false positives in the half notes extraction (see Fig. 6.2). Full notes values are detected with maximum accuracy and as far as accidentals are concerned, small segments are still mistaken for flats. Although all sharps are detected, the tonality is misunderstood because of the flat's false positives.

6.3 Test 3

Test 3 page introduces one of the untreated problems (future work) of full note heads detection. In fact, full heads of a chord very near in pitch are placed one next to the other and not one above the other as shown in Fig. 6.3. Those note heads won't be removed thus further processing will be corrupted and for example, note values will be misunderstood. Instead, single note heads are discovered with a 100% accuracy. In addition, this score has octaves indications which confuse the whole notes extraction (see Fig. 6.4). Then, rests are properly found but the last flat of every key signature is missed (corrupting the tonality detection).

6.4 Test 4

In this music page the process is good overall except numerical time signature detection. In fact, the template matching didn't work well due to the use of a slightly different font for number 2. Hence, 30% of time signatures was not detected and

removed creating ambiguity with eighth rests (see Fig. 6.5) and whole notes. This time flats are completely recognized and the tonality is well detected.

6.5 Test 5

As in test 4, test 5 fails in the time signature recognition and precisely most of numbers 2 are undetected. This will corrupt rest detections as in the previous test in which the base of the number 4 is mistaken for an eighth rest. Another important mistake is shown in Fig. 6.6 where a natural under a beam invalidate the projection method adopted in *fullnotevalueextraction* (see Sec.5.11). To conclude, the system misses one sharp basically because such sharp has been deleted in a previous step.

6.6 Test 6

This score is full of half notes and this time all of them are well recognized; the good recognition is obtained thanks to the heads of the same chord which are not so close. Then, this score contains the only mistake on clefs among all tests. An F-clef has been wrongly recognized letting on the score a G-clef. If a G-clef remains on the image until the end of the process, its centroid spiral will be detected as a whole note.

6.7 Test 7

Structural elements are well found but whole notes are heavily mistaken. In fact, there are lots of false positives found in a dynamic notation (see Fig. 6.7). In addition, the skeletonization process for the value recognition works well also for sixteenth notes. There are no accidentals on the page.

6.8 Test 8

Also this page gives great results when detecting the structure of the page and note heads. Half notes are perfectly recognized but whole notes still give some false positives. Then three eighth rests are missed and this is obviously a problem of convolution threshold setting.

6.9 Test 9

This page shows the system weakness on whole notes recognition. Indeed, the template threshold value adopted is not enough high for the whole notes detection, causing a 70% of error rate on this page. On the other side, half notes are well recognized with a 100% accuracy. As far as accidentals are concerned, some naturals are not well recognized as they are really close to each other as shown in Fig. 6.8(typographic issue).

6.10 Test 10

This last score at 300 DPI tests the recognition of notes very far from the staff and with several ledgers. The detection of such notes is perfect with maximum accuracy. In addition, this page introduces a new symbol – Pedal signature. Such object, like all words on a sheet, deceives the system when recognizing half and whole notes as shown in Fig. 6.9. Although there exists a small amount of false positives, this score gives a total detection of everything.

6.11 Test: OMRJX VS OMeR and Smart Score

This test compares the performances of *OMRJX* against those of the commercial softwares *OMeR* (see Fig. 3.12) and *Smart Score*. Furthermore, it is a challenge between *OMRJX* and itself because one of the test pages is the same of test 1 but at half of the original resolution (150 DPI).

When *OMeR* starts and a new page is loaded, the system will output some information about the integrity conditions of such page. BMP, PICT and TIFF are the only picture formats that it can support. On the other side, *OMRJX* doesn't give any information about the page quality but most of the picture formats are supported by the system. The structural part is the first clear difference between the first two systems. *OMRJX* detects and removes the accolades, the title, the staves, the bars, the clefs and both types of time signature. *OMeR*'s approach is much more superficial and the bar lines are the only objects it detects and plots. Though, some bar lines are wrongly recognized and confused with stems and braces. This step of the processing is one of the *OMRJX* major strength points. In fact the accuracy, also if the quality of the page has been halved, remains still at 100%. *Smart Score* can detect as well all structural elements and in addition it outputs a midi representation of the processed score (the user can immediately listen to it).

The difference between the three softwares is sensible. Indeed, *Smart Score*, in both tests, detects all note values while *OMRJX* has more or less an 80% accuracy and *OMeR* reaches a 70%. As far as rests are concerned, *OMRJX* is the leading system. In conclusion, the number of false positives of each system[2] can be compared emphasizing that when the quality of the page is halved this number tends to increase. As a final remark, notice that *OMeR* and *Smart Score* are commercial softwares and are on the market respectively at 20$ and 399$. In Tab. 6.2 the results of the two test pages for each of the three softwares are collected.

6.12 Conclusions

In this chapter, the results of *OMRJX* have been presented and analyzed. First, the system has been tested with 10 scores, outputs of an unknown scorewriter, at 300 DPI and then compared to two commercial softwares. Results emphasize some good and negative points. The system can rely on a real accurate detection of accolades, staffs, bar lines and titles. Sometimes, the time signature recognition fails because of the use of a different number font from the convolution template.

[2] *Smart Score* has a number of false positives near to zero.

The method proposed in Sec.5.10 is really valid and can recognize, most of the time, all full note heads. Such method is one of the few methods (see [17]) found in literature that can deal with touching heads (in piano scores). The filtering conditions of the method allow the system to maintain a very high accuracy also when the music page has low resolution. Unfortunately, the possible situation in which two near touching heads are arranged horizontally (Fig. 6.3) has not been considered thus occasionally invalidating the processing.

The other core part of the framework is the full note values detection. The skeletonization and the projection method turned out to be really accurate. The mistakes are often due to a parameter which is not set properly in the projection method. This parameter corresponds to the maximum range of observation on the projection neighbourhood of each head. Instead, skeletonization is accurate and never fails with high quality music pages.

Template matching works well when the system can force the convolution to certain regions of the image. I.e. in clef detection, the system looks just at the beginning of the first measure of each staff and in time signature recognition *OMRJX* narrows down the search to the staves limits. On the contrary, when template matching has too many degrees of freedom, the system is heavily slowed down and a correct threshold is not always tunable. Indeed, an adaptive thresholding method should be used in order to at least reduce the amount of false positives.

Furthermore, intermediate steps should be performed to remove those symbols that are not yet covered by *OMRJX* (e.g Fig. 6.4 and Fig. 6.9). Such symbols are most of the time the cause of a high number of false positives. Infact, the main idea is that for each symbol class, there are some constraints to respect. These constraints are the precedence conditions in the removal process that is, for example, if a G-clef is not removed before the whole note, there will certainly occur many misunderstandings.

In conclusion, the last lack of the framework is the pitch detection that has not been implemented. However, it has been thought a first simple solution which considers the distance of heads from the nearest staff lines. Once pitch detection will be completed, *OMRJX* could improve its performances creating an output MIDI file or XML representation of what it detected.

Score	Structure	Full heads	Void heads	Note Values	Rests	Accid.	Tonality	f_p	Precision	Recall
Test 1	78/78	143/143	15/26	146/146	0/0	7/7	✓	5	96.05%	97.25%
Test 2	102/102	257/257	3/5	289/289	0/0	30/30	✗	22	96.60%	99.71%
Test 3	89/89	248/272	16/33	196/211	14/14	21/31	✗	25	86.52%	89.85%
Test 4	107/113	335/335	1/1	266/274	0/0	33/33	✓	42	92.98%	98.15%
Test 5	95/101	331/331	0/0	223/224	0/0	38/39	✓	8	97.72%	98.85%
Test 6	121/122	388/388	30/30	372/387	7/10	2/2	✓	7	97.25%	97.98%
Test 7	120/120	546/546	7/9	229/231	4/8	0/0	✓	11	97.95%	99.12%
Test 8	119/119	644/644	7/7	197/208	4/7	1/1	✓	4	98.18%	98.58%
Test 9	119/119	122/122	34/84	84/87	27/35	23/27	✓	8	84.85%	86.29%
Test 10	71/71	198/198	2/2	192/192	27/27	28/28	✓	6	98.85%	100%

Table 6.1: Tests reuslts.

Score	Structure	Void heads	Note Values	Rests	Accid.	f_p	Precision	Recall
OMeR (150 DPI)	19/44	0/20	50/140	0/0	0/7	74	24.91%	33.99%
OMRJX (150 DPI)	44/44	14/20	83/140	0/0	4/7	39	58.00%	68.72%
Smart Score (150 DPI)	44/44	16/20	140/140	0/0	6/7	1	97.17%	97.63%
OMeR (300 DPI)	44/51	25/30	337/387	6/10	2/2	35	44.56%	46.31%
OMRJX (300 DPI)	50/51	30/30	351/387	7/10	2/2	24	87.30%	91.67%
Smart Score (300 DPI)	51/51	29/30	387/387	3/10	2/2	0	98.33%	98.33%

Table 6.2: OMRJX VS OMeR and Smart Score.

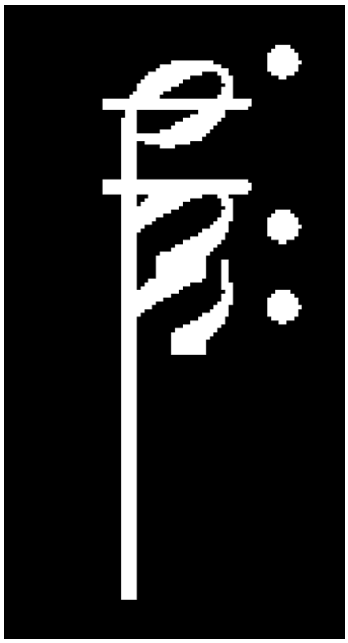


Figure 6.1: An undetected half chord.

Figure 6.2: Two examples of half heads false positives.

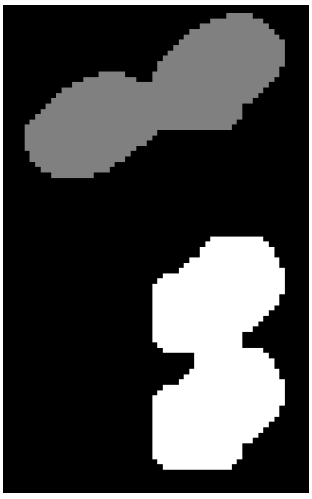


Figure 6.3: In grey two notes very near in pitch that won't be detected and removed.

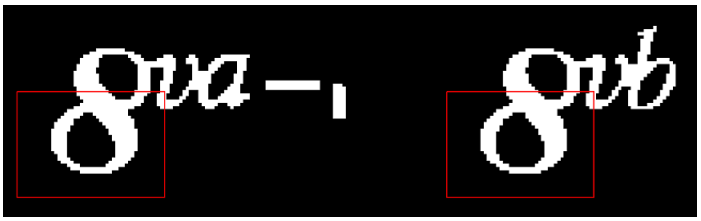


Figure 6.4: Whole notes are misunderstood with octave indications.

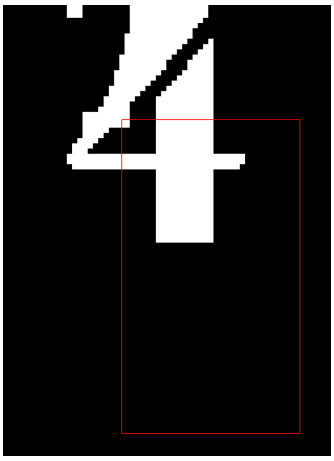


Figure 6.5: Eighth rest confuesd with the base of TS 4.

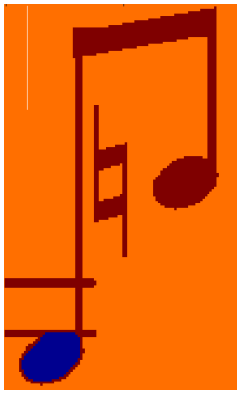


Figure 6.6: The natural pixels will be summed up to the beam's one and the blue head is then confused with a thirty-second note.

Figure 6.7: Words on the image deceive *OMRJX* while recognizing whole notes.

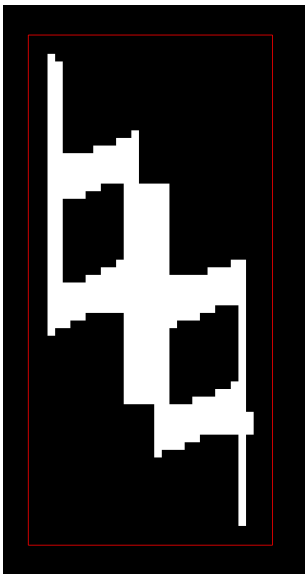


Figure 6.8: Two naturals too near from each other.

Figure 6.9: Pedal signature confused for a half note.

Chapter 7

Future Work

Despite *OMRJX* reached good results in the detection of most of the musical symbols on piano scores, there still remain a set of open issues that will be addressed in the future.

Thresholding The choice of a good threshold is always really hard. In this master thesis this issue has been addressed all the time and there are still some parts (especially template matching) of the framework that are affected by this problem. A threshold value may depend on the quality of the image or its resolution hence, in order to improve the system performances a good adaptive thresholding method must be implemented.

Chords In Fig.6.3 the only flaw in chord detection is shown. Full heads placed horizontally are due to notes whose pitch is really close (distance of a semitone). The non-recognition of such heads affects lots of further detections and gives rise to a conspicuous number of false positives. Therefore, for such issue the full note heads extraction algorithm should be reinforced adding a particular condition on note heads too much close in pitch.

Ties and slurs Ties and slurs are curved objects that are confused most of the time with half and whole notes. In addition, due to a typographic mistake, ties may touch the note heads they refer to, becoming a sort of *pixel bridge* between two heads. If ties and slurs would have been considered since the beginning, the system performances would be significantly increased.

Text and Dynamics Metronome notations and dynamics are symbols that the system can't recognize yet.

Pitch *OMRJX* can compute the rhythmic values of notes but can't still compute the pitches. Therefore, a possible idea would be to compute the distance of each centroid head to the nearest staff line.

Output In order to have a much more direct impression of the framework result, all internal information of the system could be transformed in a common MUSIC XML or MIDI notation.

Contemporary classical music Music evolves. Contemporary music is absolutely different from romantic or modern music. A good starting point would be

a proper analysis of contemporary scores and then the implementation of algorithms that can recognize such new *strange* symbols.

Author recognition Author recognition was the initial aim of this thesis. Thus, in order to discriminate classical authors it was mandatory to have a certain kind of information that a common OMR software was not able to provide. *OMRJX* reached at least the 80% of such goal information.

The proposed method is to automatically search the Petrucci library[1] for scores of certain authors and use them as input to *OMRJX*. The obtained information would be gathered and used as features for an ad-hoc neural network.

[1] `http://imslp.org/`

Chapter 8

Conclusion

The thesis has addressed the optical music recognition (OMR) issue which is the process of converting scores in such a way to be understandable by a computer. Hence, after clarifying the music notation and the main topology of the problem, a complete OMR system has been developed and explained into details. The implementation is written in MATLAB® and each function has been created from scratch.

The very first system step is that of analyzing the score looking for the main basis of it. Hence, staves and bars are recognized (removed) through the Hough transform. In order to have the geometry of the score, the staff line width and the distance between two of them has been computed. Next step is to detect and remove objects that are in a specific region of the score such as a title, clefs and time signatures. For this step a K-NN algorithm has been used for the title detection while localized template matching has been adopted for clefs and TS recognition. Next, a sophisticated algorithm has been thought for full note heads detection. This part is a sort of filtering box that rejects objects which are different from note heads and maintain heads themselves. In addition, it can deal with touching heads which is a non-trivial issue. Once the heads have been saved, a value must be assigned to each of them. For such purpose an innovative algorithm has been used, involving skeletonization and the projection method. It follows rest, accidentals and dot extraction which have been implemented through a reinforced template matching. In conclusion, the system can recognize the tonality of the score.

The results in Chap.6 showed that the structure recognition and the full note heads (with their value) detection perform highly well. The system was also tested against other commercial softwares giving interesting and good results. In fact, *OMRJX* response to low resolution scores gives about 70% accuracy.

Bibliography

[1] *Dealing with superimposed objects in optical music recognition. Trinity College, Dublin, Ireland: Proc. Sixth International Conference on Image Processing and its Applications, volume 11, 'July 1997'. 756-760.*, 1997.

[2] *Inexpensive optical character recognition of music notation: a new alternative for publishers*, April 1988.

[3] David Bainbridge and Nicholas Carter. *Handbook of character recognition and document image analysis.* 1997.

[4] Nicholas P. Carter and Richard A. Bacon. *Structured Document Image Analysis.* Sringer-Verlag, 1992.

[5] C. Dalitz, M. Droettboom, B. Pranzas, and I. Fujinaga. A comparative study of staff removal algorithms. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(5):753 –766, May 2008.

[6] Lee Sau Dan. Automatic optical music recognition. Technical report, 1998.

[7] Ichiro Fujinaga. Staff detection and removal. In Susan George, editor, *Visual Perception of Music Notation: On-Line and Off-Line Recognition*, pages 1–39. Idea Group Inc., 2004.

[8] W. Homenda and M. Luckner. Automatic knowledge acquisition: Recognizing music notation with methods of centroids and classifications trees. In *Neural Networks, 2006. IJCNN '06. International Joint Conference on*, pages 3382 –3388, 0-0 2006.

[9] Hirokazu Kato and Seiji Inokuchi. *Structured Document Image Analysis.* Springer-Verlag, 1991.

[10] S. Ohteru Matsushima, T. and S. Hashimoto. An integrated music information processing system. *International Computer Music Conference*, 1989.

[11] H. Miyao and Y. Nakano. Head and stem extraction from printed music scores using a neural network approach. In *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, volume 2, pages 1074 –1079 vol.2, August 1995.

[12] T. Ejima M. Miyahara Miyao, H. T. and K. Kotani. Symbol recognition for printed piano scores based on the musical knowledge. *Transactions of*

the Institute of Electronics, Information and Communication Engineers D-II J75D-II (11): 1848-55., 1992.

[13] D. S. Prerau. *Computer pattern recognition of standard engraved music notation.* Ph.d. dissertation, Department of Computer Science and Engineering, MIT, 1970.

[14] Laurent Pugin. Optical music recognition of early typographic prints using hidden markov models. 2006.

[15] C. Fluhr F. Pepin R. Randriamahefa, J.P. Cocquerez and S. Philipp. Printed music recognition. 1993.

[16] Florence Rossant and Isabelle Bloch. Robust and adaptive omr system including fuzzy modeling, fusion of musical rules, and possible error detection. *EURASIP J. Appl. Signal Process.*, 2007:160–160, January 2007.

[17] F. Toyama, K. Shoji, and J. Miyamichi. Symbol recognition of printed piano scores with touching symbols. In *Proc. 18th Int. Conf. Pattern Recognition ICPR 2006*, volume 2, pages 480–483, 2006.

Appendix: Matlab sourcecode list

This appendix lists all functions developed in Matlab® R2010a to build *OMRJX*. None of them relies on external functions.

1. `arrangematrix.m`
   Switch the x-th row/coloumn to y-th row/coloumn in a matrix.

2. `barlineextractor.m`
   Extract bar lines and save their coordinates.

3. `binarizeimagefolder.m`
   Binarize all the images in a selected folder.

4. `binarycontourmatch.m`
   Compare 2 binary image contours through FFT.

5. `braceextractor.m`
   Extract braces and save their coordinates.

6. `CHECKNOTEVALUE.m`
   Support function to evaluate the correctness of the value extractor algorithm.

7. `checkothersinglests.m`
   Check what's the same TS number in the lower or upper single staff.

8. `checkothersinglestsC.m`
   Check what's the same TS C in the lower or upper single staff.

9. `checkpairedts.m`
   Check paired symbol convolution in the lower or upper halfstaves.

10. `clefextractor.m`
    Extract initia clefs and save their coordinates.

11. `collapsematrix.m`
    Support function for createlabelcell.m.

12. `convcoord2imagecoord.m`
    Move from the convolution domain to the image domain.

13. `convcoord2imagecoordC.m`
    Move from the convolution domain to the image domain.

14. `createlabelcell.m`
    Associate labels of LInoTSC and FullNoteStruct.

15. `cropbinaryimage.m`
    Crop binary images.

16. `croppng.m`
    Crop binary PNG images in a folder.

17. `dotfullnoteextraction.m`
    Extract and save full note dots.

18. `doublesharpcompleteremoval.m`
    Support function for doublesharpextractor.m.

19. `doublesharpextractor.m`
    Extract double sharps and save their coordinates.

20. `FFTfilter2D.m`
    Reimplementation of conv2 matlab function for faster computations.

21. `flatcompleteremoval.m`
    Support function for flatextractor.m.

22. `flatextractor.m`
    Extract flats and save their coordinates.

23. `fullnoteheadextractor.m`
    Extract full note heads.

24. `fullnoteheadfeature.m`
    Filter full note heads before their final extraction.

25. `fullnoteinitialize.m`
    Initialize full note extrction.

26. `fullnoteremover.m`
    Erase from the image all full notes.

27. `fullnotevalueextractor.m`
    Extract the value for each full note.

28. `guesstonlity.m`
    Guess score tonality.

29. `GUI.m`
    The main *OMRJX* graphical user interface.

30. `halfompleteremoval.m`
    Support function for halfextractor.m.

31. `halfextractor.m`
    Extract and save half notes.

32. `imagecoord2convcoord.m`
    Move from the image domain to the convolution domain.

33. `imagecoord2convcoordC.m`
    Move from the image domain to the convolution domain.

34. `main.m`
    Main function.

35. `morphologicalrecognizer.m`
    Compute the best match on contour matching.

36. `naturalcompleteremoval.m`
    Support function for naturalextractor.m.

37. `naturalextractor.m`
    Extract and save naturals.

38. `noteconvolvercontrol.m`
    Filtering control for full note head extraction.

39. `numberofbraces.m`
    Compute the number of braces on the score.

40. `objectfft.m`
    Compute and save the F-transform of an image contour.

41. `restextractor.m`
    Extract and save all types of rests.

42. `sharpcompleteremoval.m`
    Support function for sharpextractor.m.

43. `sharpextractor.m`
    Extract and save all sharps.

44. `staffextractor.m`
    Extract and save the coordinates of staves.

45. `stafflinestep.m`
    Compute the distance between two stafflines.

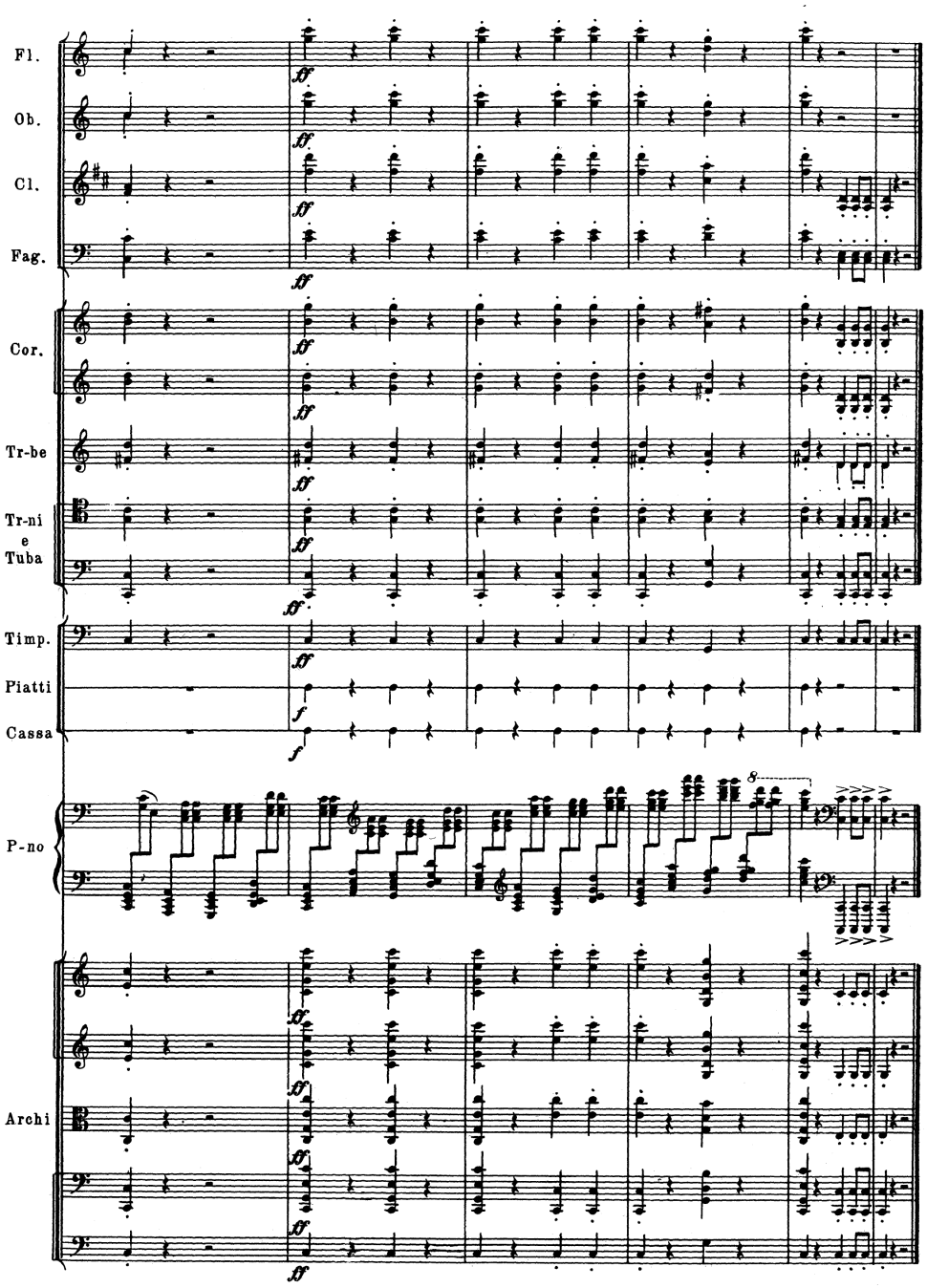
46. `timesignatureCextractor.m`
    Extract C time signatures.

47. `timesignaturenumberextractor.m`
    Extract number time signatures.

48. `titleextractor.m`
    Extract the title of the score.

49. `wholeextractor.m`
Extract and save whole notes.

50. `wholeorhalfrest.m`
Discriminate between whole and half rests.


Fl.
Ob.
Cl.
Fag.
Cor.
Tr-be
Tr-ni
e
Tuba
Timp.
Piatti
Cassa
P-no
Archi