**Universidade Nova de Lisboa**
Faculdade de Ciências e Tecnologia
*Departamento de Informática*

Dissertação de Mestrado

*Mestrado em Engenharia Informática*

# Automatic Musical Instrument Recognition for Multimedia Indexing

Frederico Alberto Santos de Carteado Malheiro (26449)

Lisboa
(Fevereiro 2011)

**Universidade Nova de Lisboa**
Faculdade de Ciências e Tecnologia
*Departamento de Informática*

Dissertação de Mestrado

# Automatic Musical Instrument Recognition for Multimedia Indexing

Frederico Alberto Santos de Carteado Malheiro (26449)

Orientadora: Prof. Doutora Sofia Cavaco

*Trabalho apresentado no âmbito do Mestrado em Engenharia Informática, como requisito parcial para obtenção do grau de Mestre em Engenharia Informática.*

Lisboa
(Fevereiro 2011)

*Aos meus pais*

# Acknowledgements

This work was integrated in the *VideoFlow* project, which is funded by a grant from the *Fundação para a Ciência e a Tecnologia*.

I would like to thank Prof. Sofia Cavaco for all her help and guidance throughout the completion of this work; Hugo Almeida and Luís Barreira for all the ideas, comments and suggestions; Hugo Guerreiro, researcher at the *Departamento de Engenharia Electrónica* at *Faculdade de Ciências e Tecnologia*, for lending the equipment necessary to record the musical instrument samples; Ricardo Andrade, musicologist and researcher at *Faculdade de Ciências Sociais e Humanas*, for the piano and his help in the recordings; and João Mateus, researcher at the *Departamento de Informática* at *Faculdade de Ciências e Tecnologia*, for the various sound clips that he obtained from the footage belonging to *Duvideo*'s archive.

To all my friends, Bernardo Toninho, Cátia Ferreira, Fabiana Seco, Francisco Silva, Hélio Dolores, Hugo Almeida, João Mateus, João Patacas, Joana Costa, Joana Pereira, Joana Matos, Luís Barreira, Luísa Lourenço, Manuel Pimenta, Mário Pires, Miguel Gomes, Pablo Vigarinho, Patrícia Beja, Pedro Simões, Ricardo Andrade, Ricardo Salvador, Rui Felizardo, Rui Rosendo, Rui Saldanha, Sílvia Alves, Sofia Penim, Sérgio Oliveira, Tiago Garcia and Tiago Seco, thanks for everything.

Finally, I would like to thank my parents for all the love and support they have always given me. Thanks mom and dad, you made everything possible.

# Abstract

The subject of automatic indexing of multimedia has been a target of numerous discussion and study. This interest is due to the exponential growth of multimedia content and the subsequent need to create methods that automatically catalogue this data. To fulfil this idea, several projects and areas of study have emerged. The most relevant of these are the MPEG-7 standard, which defines a standardized system for the representation and automatic extraction of information present in the content, and Music Information Retrieval (MIR), which gathers several paradigms and areas of study relating to music.

The main approach to this indexing problem relies on analysing data to obtain and identify descriptors that can help define what we intend to recognize (as, for instance, musical instruments, voice, facial expressions, and so on), this then provides us with information we can use to index the data.

This dissertation will focus on audio indexing in music, specifically regarding the recognition of musical instruments from recorded musical notes. Moreover, the developed system and techniques will also be tested for the recognition of ambient sounds (such as the sound of running water, cars driving by, and so on).

Our approach will use non-negative matrix factorization to extract features from various types of sounds, these will then be used to train a classification algorithm that will be then capable of identifying new sounds.

**Keywords:** audio indexing, non-negative matrix factorization, instrument recognition, ambient sound recognition, machine learning.

# Resumo

A indexação automática de multimédia é um assunto que tem sido alvo de consistente discussão e estudo. Este interesse deve-se ao crescimento exponencial de conteúdos multimédia e à consequente necessidade de criar meios que permitam gerir esta informação. Para ser possível concretizar esta ideia, surgiram vários projectos e áreas de estudo. Destes, os mais relevantes são a norma MPEG-7, que tem o intuito de definir uma forma sistemática de representação e recolha automática da informação relativa aos conteúdos multimédia, e *Music Information Retrieval* (MIR), que reúne vários paradigmas e áreas de estudo relacionados com a música.

A principal forma de abordar este problema de indexação baseia-se na análise dos dados, de modo a identificar e obter descritores que possam ajudar na caracterização do que é pretendido reconhecer (como instrumentos musicais, voz, expressões faciais, etc.), possibilitando a indexação dos dados consoante a informação obtida.

Este tema concentrar-se-á na indexação de áudio no contexto da música, propondo como principal problema o reconhecimento automático de instrumentos musicais a partir de notas musicais gravadas. Adicionalmente, o sistema será também adaptado para o reconhecimento de sons ambiente (por exemplo, sons de água, carros, etc).

A nossa abordagem irá usar factorização de matrizes não negativas para possibilitar a extracção de propriedades de som, estas serão depois usadas treinar um algoritmo de classificação que possibilitará a identificação de novos sons.

**Palavras-chave:** indexação áudio, factorização de matrizes não negativas, reconhecimento de instrumentos, reconhecimento de sons ambiente, aprendizagem automática.

# Contents

# List of Figures

# List of Tables

# 1 Introduction

For the last three decades we have seen a continuous growth of multimedia contents available to the general public. This trend is unlikely to diminish any time soon as new digital distribution methods continue to appear and grow. This development presents a problem that has yet to be conveniently solved: the indexing of these contents.

The principle of multimedia indexing is to gather knowledge from the data, be it basic information (such as its title, author, and so on), or more complex concepts (for instance, if a guitar is played on a piece of music or a movie has a scene on a beach). Its main purpose is to provide a user with an easier access to data contents. However, we still need to solve a variety of problems for it to be possible to transition the indexing task from a manual process to a completely automated process. Due to the vastness of the subjects, the study of these problems tends to be segmented through various areas. Music, for instance, is studied by a cross discipline known as Music Information Retrieval (MIR). It encompasses several other research topics, such as musical genre classification [NSM06], musical structure detection [PK06], and many others [Fin04]. Its main goal is the development of methods that allow the automatic extraction and organization of information from large collections of music (a task that is daunting to perform manually [TKSW07]). Our topic of interest, automatic recognition of musical instruments, pertains to this area, and is still far from being solved [HBPD03]. Unsurprisingly, it is, in itself, an immense topic, with many different concepts and approaches. Even so, there is a clear line that divides the problem into two classes, when considering the type of recordings from which the instruments will identified: monophonic recordings, when only single notes are present in the sound; and polyphonic recordings, when various notes are played simultaneously. This notion is crucial, as most techniques developed for musical instrument recognition in monophonic recordings can not be directly applicable to polyphonic recordings.

The monophonic problem has been widely researched through the optimization of the recognition process. This is accomplished by exploring more efficient and precise combinations of features [ALP03, Ero03]. Polyphonic recordings tend to be more realistic than their monophonic counterpart due to the notion that, in most cases, recordings will contain more complex sound constructions (e.g. chords) as opposed to simple single note compositions. Unfortunately, this notion of polyphony creates several difficulties for the recognition of specific instruments present in these recordings, such as the overlapping of features due to various instruments being played simultaneously.

There are two distinct approaches that attempt to solve this problem. The first focuses on developing methods of retrieving, directly from the mixed audio signal, information which will be used to identify the instruments present in the mix [JWR09, ZR07]. The second attempts to separate the instruments in the signal using statistical sound source separation algorithms, such as independent component analysis and non-negative matrix factorization (NMF). As a side effect, this latter approach also learns sound features that can characterize the instruments or notes in the signal [BKK06, LLT08, MBTL07, WP05]. From there the individual sounds can be treated in ways similar to monophonic approaches.

Our work can be compared to this last approach, as it proposes a system based on a NMF algorithm, however, the method we use to obtain the features needed to train the *k*-NN classifier is distinct, as we use values obtained from individual notes to create a reduced set of feature vectors. Not only does this allows us identify both the instrument and the note itself, but has also proven to yield very high recognition rates. Additionally, we adapted our system to be able to recognize ambient sounds (as explored in [MMS06]), by using several types of sound features, for which we obtained very positive results.

2

# Concepts Review

In this chapter we will approach several concepts such as sound representation, sound features, classification algorithms and blind source separation algorithms. These concepts are necessary to understand as they are intrinsic to the various stages of a musical instrument recognition system.

## 2.1 Sound Representation

Sound results from the disturbance of air particles and it can be defined as a wave that travels through the air. These disturbances are produced by sound sources (objects, people, animals) and may be due to different events (speaking or plucking a string, for instance). The way in which our auditory system handles these waves is what gives us the perception of sound.

Sound waves are characterized by the generic properties of waves – frequency, wavelength, period, amplitude, intensity, speed, and direction. Since our interest lies in musical sounds, we turn to a specific type of wave: the sinusoidal wave. As most musical instruments produce sounds that are nearly periodic, akin to a sine wave, we can approximate the overall vibration of a musical instrument as the sum of sinusoids of different frequencies using Fourier Analysis. These analyses of waveforms are computed using discrete Fourier transforms (or fast Fourier transforms for a more efficient computation), providing us with the spectrum and spectrogram of the sound produced by the instrument [Coo99].

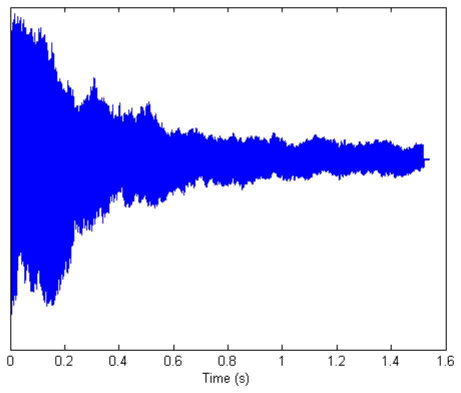

Figure 2.1: Piano note waveform

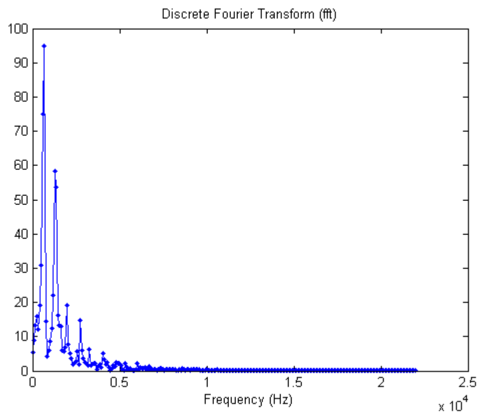

Figure 2.2: Piano note spectrum

To illustrate these concepts we will plot their graphical representations. For this, we used a sound clip of a piano note in wav format (obtained from [Pro]) that we converted from stereo to mono (by dividing the sum of the left/right audio components by two). After reading the file as an array, we plotted the waveform of the sound (figure 2.1, where the $x$ axis represents time and the $y$ axis represents amplitude), calculated and plotted its spectrum using a fast Fourier transform (figure 2.2, where the $x$ axis represents frequency and the $y$ axis represents amplitude), as well as its spectrogram (figure 2.3, where the $x$ axis represents time, the $y$ axis frequency and the colour intensity represents the amplitude).

Concerning musical notes and instruments, there are two commonly used concepts that need to be described, scientific pitch notation and musical instrument articulation.

The concept of scientific pitch notation is a form of uniquely identifying musical notes of the standard Western chromatic scale by attributing a letter do identify the note (from A to G, including accidentals) and a number identifying the octave, for instance, the note $A_4$ represents the middle A, which as a frequency of 440 Hz, $A\sharp_4$

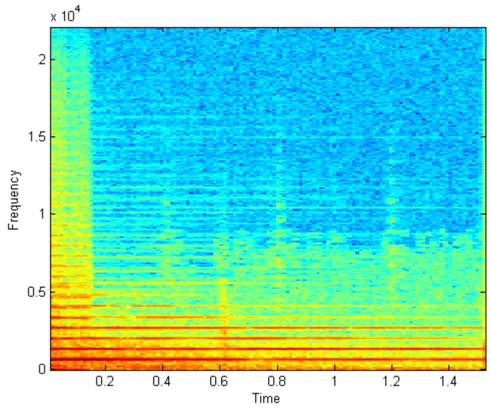

Figure 2.3: Piano note spectrogram

represents the next semitone in the scale, which has 466.16 Hz, and so on [You39].

The concept of articulation in the context of music is used to identify different techniques (such as "staccato", "legato" and so on) of transitioning between notes, however, our usage of the term is slightly more limited, as it is used to distinguish between different types of sounds that can be produced by a musical instrument. For instance, we can quickly hammer a piano key to obtain only the attack of the sound, a almost direct definition "staccato", or we can let a note ring by keeping the key pressed, which we consider "legato", even though the definition implies a transition of various notes.

### 2.1.1 Sound Features

Sound features are abstract or compressed numerical representations of a wide range of properties present in sound waveforms or spectrums. This multitude of information allows for the selection of any number of features to be used as feature vectors in data classifiers. These classifiers, as we will discuss later on, use these sound features to create distinct groups that are used to identify different sounds. In essence, sound features are the basic building blocks of a recognition system.

In the following paragraphs we will describe several methods of obtaining sound features and the properties that each one represents.

One of the most direct is to use the pure waveform and compute relevant data. For instance, by calculating the number of sign inversions it has, we obtain its *zero-crossing rate*.

Several types of spectral features can also be obtained from the sound's spectrum. Some of these are used in MPEG7 [MSS02], such as *audio spectrum flatness*, which signals the presence of tonal components if a high deviation from a flat spectral shape for a given frequency band exists; *audio spectrum centroid*, which computes the gravity centre of a log-frequency power spectrum; the *audio spectrum spread*, that gives the root mean square value of the deviation of the power spectrum in log frequency scale, with respect to the gravity centre in a frame; while others, even though being commonly used, do not belong to a specific standard, such as the spectrum's *roll off*, which calculates the averaged frequency, over all frames, below which an experimentally chosen percentage of the accumulated magnitudes of the spectrum is concentrated; the spectrum's *flux*, which gives the difference between the magnitude of the amplitude spectrum points in a given and successive frame, averaged through the entire sound; or the *energy of the spectrum*.

Temporal features, as the name implies, are based on time specific properties of the sound. Some examples that illustrate this concept are: the *temporal centroid*, the time instant where the energy of the sound is focused; or the *log attack time*, the decimal logarithm of the sound duration from the time instant when the signal starts, to the time when it reaches its maximum value, or to when it reaches its sustained part.

Finally, a commonly used feature is Mel-frequency cepstral coefficients (MFCC). While originating from speech recognition, as its design was to closely approximate the human auditory system by equally spacing the frequency bands on the mel scale, it was shown that it can be used to quite effectively in other contexts (namely, musical instrument recognition [Ero03] or even ambient sounds recognition [MMS06]).

There is continued research based on combining different numbers of these and many other features, as well as developing new kinds of features, with the purpose of improving recognition rates of sounds [HABS00, HBPD03, JWR09].

#### 2.1.1.1 MPEG-7

This standard was an important step to centralize and uniformize a great number of multimedia features. Unlike other standards developed by the Moving Picture Experts Group (MPEG), more focused on defining digital audiovisual compression standards, MPEG-7 was designed as a standard for the representation and automatic extraction of information (descriptors) present in audiovisual content. This notion is invaluable, as it defines a framework that allows us to continue a methodical approach to audio indexing.

The MPEG-7 standard aims to: provide a wide application base; support a wide array of data types; maintain a clear relation with the content; maintain media and format independence; be Object-based; allow different levels of abstraction; and permit extensibility. These principles are achieved by relying on a number of different tools: descriptors representing features (a feature is a distinctive characteristic of the data); description schemes that specify the structure and semantics of the relationships between its components; a description definition language (DDL) that allows the creation of new description schemes and descriptors; and several system tools needed for the binarization, synchronization, transport and storage of descriptors.

The creation of this standard took in consideration several aspects (such as visual, colour, texture, shape and motion descriptors, spoken content, as well as its applications for search, browsing and mobile environments) that, while very important to the applicable areas, will not be described here. For a more detailed review of the MPEG-7 standard, refer to [MSS02].

## 2.2 Classification Algorithms

Classification algorithms, being data driven, have the capability of being adapted to whatever data is in hand, giving us the possibility to fine tune class definition and the classification process. To explain these concepts, let us consider the subject of automatic musical instruments recognition.

Firstly, we need to understand what are classes. In general, classes are intended to define any kind of group, in this case, classes represent musical instruments. The algorithms learn how to classify the data into different classes through training samples, these features will serve as the comparison basis. The samples are represented through features that should be able to represent inherent characteristics of an instrument. These features are obtained by analysing various aspects of the sound wave - either directly, or through a transformation of the sound such as the sound's spectrum using a Fourier Transform, which allows to obtain other various properties.

Finally, we can input test samples to the trained algorithm so it can determine the class to which it belongs. The quality of the results, however, will depend both on the training data used and the algorithm itself.

In the following subsections, we will describe several classification algorithms that have been used in the area of musical instrument recognition.

### 2.2.1 *k*-Nearest Neighbours

Given a set of training samples and a test sample, this algorithm determines the $k$ nearest training samples in the set, assigning the test sample to the class with the most samples.

The "nearest neighbours" concept stems from the idea of having a multidimensional graph where the features are used as its variables. The training samples are then mapped as points in the graph, creating clusters that represent classes. When presented with a test sample, the algorithm calculates its distance between the existing samples using a function (generally, the Euclidean distance), allowing the closest samples to be determined. The sample is then considered to be the same class as the majority of the testing samples.

This algorithm has some drawbacks: all the training instances are required to be in memory; each new query may need significant computation time; and irrelevant features may have a disproportionate impact on class definition and/or classification. Nevertheless, its advantages are considerable: since no training exists, setting up the algorithm is just a matter of storing the data; and the results obtained are generally above average. See [Mit97] for a more detailed view of the matter.

It is widely used in musical instrument recognition (see [ALP03, EK00, JWR09, JZRC09]), providing consistent good results.

### 2.2.2 Hidden Markov Models

Hidden Markov Models (HMM) are represented as an automata, in which the transitions between states can be probabilistic. It can be used to model a wide variety of time related data. In addition to what defines an automata, HMM contain two more components: a set of hidden variables that cannot be observed directly from the data; and a Markov property that is usually related to some sort of dynamic behaviour of the hidden variables.

The paradigm is used to solve three main tasks: classification, segmentation and learning. Learning is the first problem that needs to be solved in order to use this model, unless the parameters of the model are externally specified. This means estimating the parameters of the models. The tasks of segmentation and classification are accomplished via forward-backward recursions, which propagate information across the Markov state transition graph. For instance, using several candidate HMM models that represent different acoustic sources (musical instruments in our case), the classification problem computes the probability that the observations came from these models. Whichever model gives the highest probability is chosen as the likely source of the observation. See [Blu04] for a more formal definition of HMMs.

An example of HMMs in instrument recognition is [Ero03]. Eronen used a HMM to model temporal features with good results.

### 2.2.3 Artificial Neural Networks

An Artificial Neural Network (ANN) is a learning method that provides a way of approximating functions. The concept is loosely motivated by biological learning systems, as it tries to mimic the architecture of the brain. This is accomplished with the grouping of simple units, called neurons, into layers (input, output, and hidden) that can be interconnected through different connectivity patterns. For instance, a neuron may have several inputs and produce an output that can be used as an input to another neuron, present in another layer. These connections between neurons also have weights, that are learned during the training of the algorithm. These changes may proceed either supervised or unsupervised. In the former case, a teaching instance is presented to the ANN, it is asked to generate an output which is then compared with an expected "correct" output. The weights are consequently changed in order to minimize future errors. In the latter case, the weights "settle" into a pattern that represents the collection of input stimulus.

The drawbacks of the ANN are as follows: the computation time for the learning phase is very long; determining the correct parameters (the architecture of the network, the number of hidden layers, etc) is a non-trivial task; and possibility of over-fitting (an excessive number of bad selected examples) can degrade their generalization capabilities. On the other hand, ANN have the advantage of being highly robust when trained correctly, as well as very fast at computing the classification of test samples.

Even though ANN are not widely used in the recognition of musical instruments, there are a few studies that explore its use, providing mixed results. The results obtained by Zhang using a type of unsupervised ANN (Kohonen's self-organizing map) were very good [Zha01]; and Jiang et al stated that, given the high number of attributes needed by the proposed approach, the algorithm would not perform well, even though no results were published [JWR09].

### 2.2.4 Support Vector Machines

Support Vector Machines (SVM) are a decision-based prediction algorithm, directed at data classification. SVM perform classification by constructing an $n$-dimensional hyperplane that optimally separates the data into two categories.

The concept of this algorithm is similar to other classification algorithms. Given a test sample along with set of training samples, whose the values are defined as belonging to one of two categories, the SVM training algorithm constructs a model that predicts the category of the test sample. This model represents the samples as points in space, mapped to create the largest possible gap separating the categories. When test samples are given, the algorithm will map them on the same space, allowing it to predict its category based on the side of the gap that it is located.

In the event that the target variable has more than two categories, two common approaches exist: “one against many”, where each category is split out and all of the other categories are merged; and “one against one”, where $k(k\text{-}1)/2$ models are constructed (with $k$ being the number of categories).

SVMs can provide high recognition rates when used for distinguishing musical instruments[ALP03].

### 2.2.5 Decision Trees

Decision trees (DT) are a method of approximating discrete-valued target functions. DT are constructed top-down, starting with the most informative feature. This is decided by evaluating each instance attribute using a statistical test that determines how well the single feature can classify test samples. Afterwards, branches are created from each one of the different values of this feature, as training samples are sorted to the appropriate descendant node. Finally, the entire process is repeated recursively using the training samples for each of the descendant nodes. Once the tree has been built, it can be pruned to avoid over-fitting and to remove any secondary features.

Some disadvantages of DT are: how to determine the depth allowed for the growth of the tree; the handling of continuous attributes; choosing appropriate attribute selection measures; and the improvement of computational efficiency. However, the generated tree is simple to interpret, the algorithm is robust (the training data may contain errors) and the target function has discrete output values.

Two examples of use of this classifier can be seen in [LW07, JWR09]. The results obtained in both works were mediocre, as it was consistently the worst performing classifier.

### 2.2.6 Naive Bayesian Classifiers

A Naive Bayes Classifier (NBC) is a probabilistic classifier based on the application of Bayes’ theorem using strong independence assumptions. Basically, this allows us to construct the posterior probability for a event among a set of possible outcomes or categories given a set of variables or predictors.

The assumption that the predictor variables are independent, while not always accurate, greatly simplifies the classification task, as it allows the class conditional densities to be calculated separately for each variable, reducing a multidimensional task to a number of one-dimensional ones.

The main advantage of NBC is that it requires only a small amount of training data to estimate the parameters necessary for classification. However, one difficulty of applying the method is the significant computational cost required to determine, in some cases, the Bayes optimal hypothesis.

Naive Bayesian classifiers were employed in [LW07] obtaining good results, similar to those of rough sets.

### 2.2.7 Rough Sets

Rough sets are used to evaluate the relevance of the features used for description and classification. While vague concepts cannot be characterized in terms of information about their elements, they may be replaced by two precise concepts: lower and upper approximation. The former consists of all the objects that are guaranteed to belong to the concept, while the latter contains all the objects that can possibly belong to the concept.

The assignment of an object to a set is made through a probabilistic membership function. Once information is organized into information tables this technique is used to assess the degree of vagueness of the concepts, the interdependency of attributes and therefore the alternatives for reducing complexity in the table without reducing the information it provides. Information tables regarding cases and features can be interpreted as conditional decision rules of the form *IF (feature x) is observed, THEN (isan Y object)*.

While not widely used, Lewis and Wieczorkowska obtained very good results with this technique, surpassing all the other classification algorithms they tested [LW07].

### 2.2.8 Discriminant Analysis

Discriminant analysis (DA) is a statistical technique that allows the study of two or more object classes while considering several variables. Considering a classification involving two target categories and two predictor variables, DA finds a transformation ("discriminant function") of the two predictors that yields a new set of transformed values that provides a more accurate discrimination than either predictor could by itself.

Two variants of this technique are used in [ALP03]: Quadratic Discriminant Analysis; and Canonical Discriminant Analysis. While the former gave positive results all around, the latter performed the worst in all the test conditions.

## 2.3 Audio Source Separation

The principle of audio source separation stems from the interest in emulating an ability common to most human beings: the capacity to distinguish and focus on particular sounds when various sounds sources are present. This so called "cocktail party effect" [Aro92] while easy to understand, is not as easy to reproduce. While several algorithms implement this idea, such as Principal Component Analysis (PCA), Independent Component Analysis (ICA) and Non-Negative Matrix Factorization (NMF), the problem is not yet solved.

The focus of our study was directed to NMF, as preliminary tests using both ICA and NMF indicated that better results could be obtained using the latter. In the following subsection we present a small introduction to the principles of NMF.

### 2.3.1 Non-Negative Matrix Factorization

NMF was developed to take advantage of the non-negativity constraint for matrix factorization [LS00]. NMF does this by taking a non-negative matrix **S** and decomposing it into two, also non-negative, matrices $\mathbf{\Theta}$ and **P**, such that

$$\mathbf{S} = \mathbf{\Theta} * \mathbf{P} \quad . \tag{2.1}$$

In the context of sounds, these matrices represent two tangible concepts: $\mathbf{\Theta}$, the mixing matrix, contains the spectra of the sounds; while **P**, the source matrix, contains the temporal envelopes of the sounds.

The algorithm produces these matrices by calculating the spectral basis functions, $\mathbf{\Theta}$, that describe the spectral regularities in the frames of the data set (that is, in $(\mathbf{S}_1, \ldots, \mathbf{S}_\mathrm{N})$, where N is the number of sound samples). This transposes the data to a new referential in which its axes are the spectral basis functions.

With this transformation, each frame is now represented by a new set of coefficients, one for each basis function. By taking, for each of the basis functions, all of the coefficients that represent the frames of one spectrogram, a vector is created. Each of these vectors represent source signals that range over time (temporal source signals). These vectors, associated to a spectrogram $\mathbf{S}_n$ (with $1 \leqslant n \leqslant \mathrm{N}$), can then be represented by a matrix $\mathbf{P}_n$ whose rows contain the source signals. Therefore, the spectrograms $(\mathbf{S}_1, \ldots, \mathbf{S}_\mathrm{N})$ can then be expressed as

$$(\mathbf{S}_1, \ldots, \mathbf{S}_\mathrm{N}) = \mathbf{\Theta}(\mathbf{P}_1, \ldots, \mathbf{P}_\mathrm{N}) \quad , \tag{2.2}$$

where $\mathbf{\Theta}$ is a matrix with one spectral basis function per column. The $i$th row of every matrix $\mathbf{P}_n$ is associated to the $i$th basis function in $\mathbf{\Theta}$, that is, its $i$th column.

These matrices calculated by NMF, in essence, isolate characteristics of specific sounds. This will allow us to create training sets for our classifier, as well as the data necessary to test new sounds.

3

# State of the Art

The recognition of musical instruments is a vast topic, with many different concepts and approaches. Even so, when we consider the type of recordings from which the instrument will identified, there is a clear line that divides the problem into two classes: monophonic recordings, when only one sound is played at any time (section 3.1); and polyphonic recordings, when various sounds can be played at once (section 3.2). This notion is crucial, as most techniques developed for musical instrument recognition in monophonic recordings are not directly applicable to polyphonic recordings.

There are two concepts, however, that are independent of these classes due to their applicability in both monophonic and polyphonic recordings, instrument families and musical instrument hierarchization. The notion of instrument families is used as a form of grouping similar sounding instruments in classes (for instance, violin and violoncello both belong to the "strings" family). Generally, identifying the family of an instrument is a much simpler task than identifying the specific instrument, due to a necessarily lax definition of the class in terms of features. On the other hand, musical instrument hierarchization is a much more formal concept, as it tries to systematize the classification of instruments. An example of this is the Hornbostel-Sachs system, widely used by ethnomusicologists and organologists. It divides musical instruments into five main categories (idiophones, membranophones, chordophones, aerophones and electrophones), which then contain numerous other subcategories, defined according to many physical characteristics of the instruments. Therefore, hierarchization is usually used as form of improving recognition rates.

In the following sections we describe several works that are relevant to the comprehension of this problem and the techniques devised to solve it.

## 3.1 Monophonic sounds

The recognition of musical instruments in monophonic recordings has been widely researched, having the main motivation of optimizing the recognition process. This is accomplished by exploring more efficient and precise features, or combinations of features, as we will see in the following works.

### 3.1.1 Musical Instrument Identification: A Pattern-Recognition Approach

Martin and Kim proposed a set of acoustic features to define physical properties of musical instruments [MK98]. The features proposed are: pitch; frequency modulation; spectral envelope; spectral centroid; intensity; amplitude envelope; amplitude modulation; onset asynchrony; and inharmonicity. The choice of these features is based on the research that the authors compiled about the subject. As this work was purely theoretical, no classification algorithms were used. They demonstrate manually the validity of the features by using the examples of the correlograms obtained from violin, trumpet, and flute tones, and analysing the values of the features. Several of these features are also used in MPEG-7, which is contemporary to this work.

### 3.1.2 Comparison of Features for Musical Instrument Recognition

Eronen presented an analysis of several sound features based on their success rates for classification of musical instruments [Ero01]. He explored cepstral, temporal and spectral sound features on families of musical instrument (wind, brass, strings) combined with a *k*-Nearest Neighbours (*k*-NN) algorithm.

The results for individual features showed that the mel-frequency cepstral coefficient (MFCC) provided the best accuracy for single and families of instruments (respectively, 30% and 60%). Furthermore, combining MFCC with features that describe the type of excitation, brightness, modulations, synchronicity and fundamental frequency of tones, as well as longer note sequences, improved the overall recognition of instruments in both cases.

### 3.1.3 Feature Selection for Automatic Classification of Musical Instrument Sounds

Liu and Wan presented an approach where a small set of features is used to classify musical instruments into the correct instruments families [LW01]. They agree that using a great number of features does not only lead to worst computational performance, as well as some features can lead to the degradation of the classification accuracy. The reduced feature set is selected by a sequential forward feature selection method. They used several classifiers: nearest neighbours (NN); *k*-NN; and gaussian mixture models (GMM). The results show that for a feature dimension of more than 20, the accuracy of GMM starts to decline, while for NN and *k*-NN an accuracy of 90% can be achieved with as few as 15.

### 3.1.4 Musical Instrument Timbres Classification with Spectral Features

Agostini *et al* focused on using a limited number of sound features for musical instrument recognition [ALP03]. This is achieved by focusing on spectral features and using eighteen descriptors for each tone, computed from the mean and standard deviation of nine features (zero-crossing rate, spectral centroid, spectral bandwidth, inharmonicity, harmonic energy skewness and the percentage of energy contained in each one of the first four partials). These are then used as training samples for the classification algorithms. To evaluate the recognition rates between algorithms, several are tested: discriminant analysis (DA); quadratic discriminant analysis (QDA); canonical discriminant analysis (CDA); *k*-NN and support vector machines (SVM).

The results are presented in a thorough manner, simplifying the analysis of the numerous data. They concluded that the best classifier for the recognition of individual instruments was the SVM (with a success rate of 69.7%, 78.6%, and 80.2% for, respectively, 27, 20, and 17 instruments). QDA achieved the second best score, with success rates similar to SVM, while overcoming all the other classifiers in family recognition and sustain/pizzicato (with a sucess rate of 81%). *k*-NN performed closely to QDA, obtaining best results with $k = 1$ (with a sucess rate of 74.5% for 20 instruments). CDA had the worst performance (71.2% with 17 instruments to 60.3% with 27 instruments).

### 3.1.5 A Categorization of Musical Instrument Sounds based on Numerical Parameters

Lewis and Wieczorkowska presented a hierarchical categorization methodology of musical instruments sounds that differs from the Hornbostel-Sachs system due to being based on the acoustic properties of the sound (temporal and spectral features), instead of the physical properties of the instrument [LW07].

The features used are based on MPEG-7 low level sound descriptors, which lead to a set of seven descriptors: LogAttackTime, the logarithm of the time duration between the point where the signal starts to the point it reaches its stable section; AudioHarmonicityType, describes the degree of harmonicity of an audio signal; Sustainability, is defined into 5 categories based on the degree of dampening or sustainability the instrument can maintain over a maximum period of 7 seconds; SpectralCentroid, measures the average frequency, weighted by amplitude, of a spectrum; TemporalCentroid, characterizes the signal envelope, representing the time at which the energy of a signal is focused (for example, it may distinguish a decaying piano note from a sustained organ note, when the lengths and the attacks of the two notes are identical); and Articulation, which represents the way how the sound is performed by a player.

To verify the applicability of this categorization system, several classification tests are conducted using rough sets, naive bayesian classifiers and a *c4.5* decision tree. The results obtained were very positive for the two first classifiers (with an average of 80% accuracy), while the third did not perform as well (with a 69% accuracy).

### 3.1.6 Application of Temporal Descriptors to Musical Instrument Sound Recognition

In this study, Wieczorkowska *et al* presented an approach that considers the temporal evolution of MPEG-7 descriptors, as well as their combinations, with the goal of optimizing sound representation in terms of musical instrument recognition purposes [WWSS03]. This goal is accomplished by using a small number of sound descriptors, as well as detecting "episodes". collections of events that are periodic in time, from the various descriptors. The descriptors used by the authors are: temporal descriptors (composed by signal length, relative length of the attack, quasi-steady and decay time, and moment when the maximal amplitude is reached); and spectral descriptors (composed by selected groups of harmonics in spectrum, tristimulus parameters, brightness, irregularity and fundamental frequency). To store this information, as well as all the training samples, they used a relational database. The classification algorithms they used were: *k*-NN; and RS-decision rules, an algorithm based on rough sets theory.

The results are presented as the performance obtained by the classifiers when using the various combinations of feature sets. Overall, the accuracy of the RS-decision rules classifier was quite low, underperforming *k*-NN in every instance, except for the combination of spectral descriptors with temporal patterns. The highest accuracy was obtained with *k*-NN using the following combination of descriptors: temporal and spectral (with 68.4%); and temporal, spectral and relational (with 65%). Unfortunately, as an in depth description of the results was not given, it is not possible to analyze the recognition rates for specific instruments or instrument families.

### 3.1.7 Musical Instrument Recognition using Cepstral Coefficients and Temporal Features

Eronen presented a musical instrument recognition system that is both pitch independent and makes an effective use of spectral and temporal features simultaneous [EK00]. The pitch independence is achieved by estimating the fundamental frequency of the test sample before classification, then comparing it to different instruments in similar pitch ranges. The feature set used was extensive, 44 in total, combining two types of features (cepstral coefficients and temporal features).

The classification method is based on a taxonomic representation of musical instruments, composed by three levels. In the first, instruments were divided into pizzicato and sustained, the second comprises instrument families, and the third, individual instruments. At each node either a Gaussian or a *k*-NN classifier were used to determine the best fitting sub-class for a test sample. The classification results were obtained through tests in three types of conditions, two different types of hierarchies and one without a hierarchy. The results were positive, as the classification accuracy for instrument families was over 93% and for individual instruments over 75%, but also showed that the use of hierarchies brought no significant advantage.

### 3.1.8 Musical Instrument Recognition using ICA-based Transform of Features and Discriminatively trained HMMs

Eronen proposed a system for the recognition of musical instruments from isolated notes in [Ero03]. He used a Hidden Markov Model (HMM) as a classifier to obtain a better representation of the spectral shape, due to its use in representing temporal evolution. As most musical instruments have several stages that are temporal by definition (a distinctive onset period, followed by a steady state, and finally a decay), the HMM was better suited for this type of modelling. The sound features used are mel-frequency cepstral coefficients (MFCC), since previous research has concluded it to be a well-performing feature set in musical instrument recognition, and delta cepstrum coefficients ($\Delta$ MFCC). To enhance the recognition accuracy, the features used are transformed using independent component analysis (ICA) (an optional technique defined in the MPEG-7 standard), as well as employing discriminative training. For the tests, the number of states and component densities per state for the HMM is modified using a heuristic segmentation scheme (each sound was segmented into as many adjacent segments as there were states in the model). The results show that using the ICA transform gives an almost consistent improvement in recognition accuracy across the set of model orders tested. Using discriminative training improves the accuracy mainly with models that have a low number of components in state densities.

## 3.2 Polyphonic Sounds

Polyphonic sound recordings are more realistic than monophonic recordings due to the notion that, in most cases, recordings will contain a mixture of various sound types. For example, while music exists in both types of recordings, a music piece can be for a solo piano or a full orchestra, it is, in general, more likely to contain instances of various musical instruments than just a single one. Unfortunately, this notion of polyphony creates several difficulties for the recognition of specific instruments present in these recordings, such as the overlapping of features due to various instruments being played simultaneously. These problems can not be directly resolved by the techniques developed for monophonic recordings.

There are two popular methods of accomplishing recognition of musical instruments in polyphonic recordings. The most covered develops various forms of retrieving data directly from the audio signal and then identifying the instruments from this data, some applications of this concept are presented in section 3.2.1. The other, less documented, method attempts to separate the instruments present in the audio, via blind source separation techniques, and apply classification algorithms to each of the generated tracks, some examples of this approach are described in section 3.2.2.

### 3.2.1 Audio analysis

#### 3.2.1.1 Instrument Classification in Polyphonic Music Based on Timbre Analysis

Zhang proposed a system that allows the recognition of musical instruments in both monophonic and polyphonic recordings by segmenting the music into notes [Zha01]. Its purpose is directed at the automatic categorization and retrieval of music pieces based on instruments. The features used are the ones that had been found in previous research to be important in determining the timbre of a note. These features were categorized in three groups: temporal, spectral and partial. Specifically, these include the temporal envelope and spectral power distribution of the note, locations and amplitudes of the partials, as well as their degree of inharmonicity and the presence of high frequencies at the moment of the attack.

The actual system is composed by several modules. First, the music piece is segmented into notes by detecting note onsets. Then, for each note, harmonic partials were detected, note features were computed, being normalized afterwards (to be independent of the volume, the length and the pitch of one note) and finally the feature vector is sent to the classifier. Having completed the process for all notes, the music piece can then be classified. The method used for the note classification is twofold. Initially, a Kohonen's self-organizing map (a type of unsupervised artificial neural network) is used to generate an optimal structure for the feature vector. The feature vectors of all the training and testing samples were then rearranged according to this structure. The training samples were used to train a multi-layer perceptron (MLP), which would be able to classify the notes (represented by the testing samples).

The results produced were promising, as an accuracy of 80% was achieved for the recognition of the main musical instruments (or 90%, when ignoring classification mistakes within the same instrument family). However, the music pieces used for the experiments were not thoroughly described, as well as no comparative analysis of the recognition accuracy between different instruments was shown. This leads to a difficult avaliation of the method, even if the basic ideas seem sound.

#### 3.2.1.2 Sound Isolation by Harmonic Peak Partition for Music Instrument Recognition

Zhang and Ra presented a system for the recognition of musical instruments in polyphonic sounds that uses harmonic peaks as a signature for instrument isolation [ZR07]. The sound isolation algorithm uses a clustering technique to separate the energy of the fast Fourier transform (FFT) points in the power spectrum by taking each harmonic peak as a cluster centroid. In other words, they implement sound separation by clustering the energy around each harmonic peak. They used time-frequency features based on MPEG-7 low-level descriptors, as well as several others, heavily based on harmonic spectra, for the transient state, quasi-steady state and the whole segment. Also, they used four different classifiers: bayesian networks (BN); logistic regression model (LRM); local weighted learning (LWL); and tree 48 (t48).

The system is composed by numerous modules. Initially, the audio signal is divided into a series of frames, with a short time Fourier transform (STFT) being applied to each. Afterwards, a harmonic peak estimator creates sets of data which is then passed on to an energy clustering engine that creates a coefficients matrix by proportionally calculating the energy of each FFT point between each pair of heterogeneous harmonic peaks. Next, an inverse FFT engine multiplies the data obtained from the STFT in the first step with the coefficients matrix transforming it back to the time domain in the form of separate audio files. Finally, the feature values are extracted from the separated sounds and are used as data for the classifiers.

A database containing musical sound recordings of seven instruments (violin, viola, cello, bass flute, flute, clarinet and alto flute) for two instrument families (string and woodwind) was used for the experiments. The training data was composed of all the database and the testing data was composed of twelve polyphonic sounds produced by mixing pairs of sounds from different instrument families. The results were measured by the the classification correctness of each set of feature vectors over different classification algorithms, showing that the BN, LWL and t48 classifiers had very similar overall performance (near 65%) while the LRM algorithm had a lower performance (54%).

There are two criticisms to this research. First, as the classification was only focused on determining instrument family, we do not know its performance when applied to individual instruments. Second, as no tests were done for polyphonic recordings with more than two instruments, we cannot conclude how well the system can scale or if this solution is contained to only this scope.

#### 3.2.1.3 Music Instrument Estimation in Polyphonic Sound Based on Short-Term Spectrum Match

Jiang *et al* propose a method that minimizes the information loss of non-dominant musical instruments due to the sound separation process by detecting sub-patterns directly from the short term power spectrum of the sound [JWR09]. This is an interesting approach, as sound features helpful to the identification of instruments can be obscured by their overlapping on polyphonic recordings.

The training is done by extracting the power spectrum from frames of single instrument sounds and saving these in a database. The frames have 8192 numeric samples and a duration of 0,04 seconds. Due to the high number of attributes, they use a *k*-NN classifier, as other algorithms (such as artificial neural networks) would not perform well.

The estimation process is possible by extracting the power spectrum from the frames of a test sample and using the *k*-NN classifier to determine the most similar frame in the database. However, contrary to traditional pattern matching or classification processes, when multiple matches are obtained all the candidates are recorded, with a confidence value attributed by the classifier to each one. This is due to the notion that it is possible for several different sub-patterns to exist in a spectrum. Therefore, to obtain a higher recognition rate, an algorithm (top-*n* winners) was developed where the instruments are determined by a vote. First, the instruments (candidates) detected for each frame are saved along with their confidence level in candidate pool. After evaluating all the frames of the test sample, the weights for all the candidates from the pool are calculated by adding up their confidences, and the final voting proceeds according to the weights of each instrument.

They performed a total of five experiments to evaluate the accuracy of the system. The testing samples were 52 polyphonic audio files created by mixing two of a set of 26 different instruments. The first two experiments were performed using a feature based classification strategy (that used a collection of five groups of both temporal and spectral features mainly originating from the MPEG-7 standard and a decision tree as a classifier), while the last three used the new system. The results showed that the recognition rate of the new system was far superior to the featured based classification strategy (in over 30%). Moreover, the highest accuracy was achieved by a combination of a higher number of neighbours (the parameter *k* in the *k*-NN algorithm), two top *n* candidates and disregarding percussion instruments. However, no mention was done of the system's performance on polyphonic recordings with more than two instruments, leading to the same criticism we noted in [ZR07].

### 3.2.2 Audio separation

Regarding these approaches, there are two common problems that severely encumbers their usage in real-life applications. First, it is necessary to have *a priori* knowledge of the number of sources (for instance, the number of musical instruments) present in the recording. Second, all the experiments used polyphonic recordings with only two instruments.

#### 3.2.2.1 Musical Audio Stream Separation by Non-Negative Matrix Factorization

Wang and Plumbley developed a methodology for separating musical audio into streams of individual sound sources [WP05]. They first decompose the input signal into a time-frequency domain by using the non-negative matrix factorization (NMF) algorithm. Afterwards, they generate time frequency masks by comparing the energies of decomposed bases and applying those masks to the spectrogram. Finally, the bases are grouped in the time domain to produce separated audio streams.

They performed two types of experiments. First, they tested the system on an artificial mixture of two instruments (piano and flute). Listening to the results, they found that separated flute sound had better quality, while the piano sound suffered some interference towards the end. The measured signal to noise ratio (SNR) of the experiment showed that the flute sound had an higher SNR than the piano sound. The second test recording is a real music clip played by flute and guitar, as the original individual sources are not available, only subjective analysis was given. Their opinion stated that the result sounded acceptable on the whole and the rhythm was extracted, but some guitar sound leaked to the flute channel. Overall, the results were positive.

#### 3.2.2.2 Estimation of Musical Sound Separation Algorithm Effectiveness Employing Neural Networks

Dziubinski *et al* presented a study that, while mainly focused on the separation of musical instruments using blind source separation techniques, employed a ANN classifier with purpose of evaluating the accuracy of the various separation algorithms and procedures that were used [DDK05]. They used four different algorithms for the separation of sounds, each using a different approach based on the harmonic signals of the sounds. The features were mostly based on MPEG-7 descriptors (audio spectrum envelope, audio spectrum centroid, audio spectrum spread, audio spectrum flatness, log attack time, harmonic spectral centroid, harmonic spectral deviation and harmonic spectral spread), however, harmonic content descriptors were also used (*KeyNum*, modified tristimulus parameters, and descriptors related to the behaviour of harmonic components in the time-frequency domain). For the classification task only an ANN classifier was used.

The experiments were conducted very thoroughly, as each of the four separation algorithms was combined with each of the four separation procedures, while the classifier was trained with six different feature sets (each using a different combination of features). The procedures focused on diversifying the separation process by using different orders in which the instruments were separated, based on their pitch or expected residual signals. The training samples used were based on the set of five musical instrument sounds of the same musical articulation, for a total of 361 sound samples. Each test sample contained from two up to four sounds, randomly chosen, each belonging to a different class. Only different pitched sounds were considered, that is there were no sounds of the same pitch within one mix, in all cases transients of the mixed sounds were overlapping.

In general, the results were very positive, achieving recognition rates of over 90%

in several cases. However, the authors give two recomendations: different descriptors might be necessary in more extensive tests (for mixes containing more than four instruments); and that the perfomance of these techniques should be analysed by using real polyphonic sounds (as opposed to the artificially mixed isolated signals that were used), with the the purpose of obtaining more complete results.

#### 3.2.2.3 Musical Instrument Category Discrimination Using Wavelet-Based Source Separation

Lampropoulou *et al* propose a system based on blind source separation (BSS) for the recognition of musical instrument categories [LLT08]. The system is composed by two phases. First, the sources (or instruments) are separated using a BSS technique called Convolutive Sparse Coding. Afterwards, a Gaussian Mixture Models is used to classify the musical category of each of the separated instruments. Three categories are used: wind; brass; and strings. They extract, from each sound signal, a set of 30 features that not only provide a low level representation of the statistical properties of the music signal but also include high level information, extracted by psychoacoustic algorithms in order to represent rhythmic content (rhythm, beat and tempo information) and pitch content describing melody and harmony of a music signal.

The sound samples used in the experiments were obtained from a musical instrument database that contains numerous instruments belonging to the referred categories. To create the audio signal for separation, sound samples of different instruments were selected and mixed in a program for the effect. The results were measured by analysing the predicted instrument category rate in relation to the actual instrument category. The most successful classification rate was of the wind category, also having the lowest mismatch rates. The overall accuracy was 70,66%, with an overall mismatch rate of 22%.

# 4 Implemented System

## 4.1 System overview

This section presents a simple and abstract view of the system, see figure 4.1, allowing to clearly distinguish its four main modules and how they are interconnected. As the architecture of our system follows previous works in this research area, its modules/stages are named in accordance to what is the commonly used: training, testing, classifier and data analysis. This system was implemented in MATLAB, using a NMF software package provided by Virtanen [Vir07].

In the following section we will describe in detail each of these modules by following their inner processes and explaining the important concepts behind each, as well as how they are relevant for the system in its entirety.

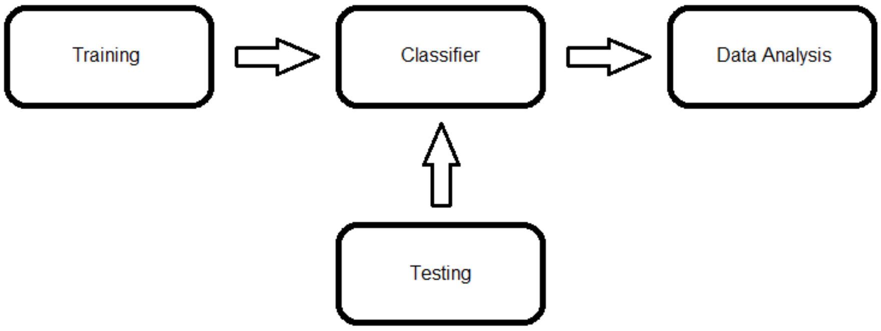

Figure 4.1: Overview of the recognition system

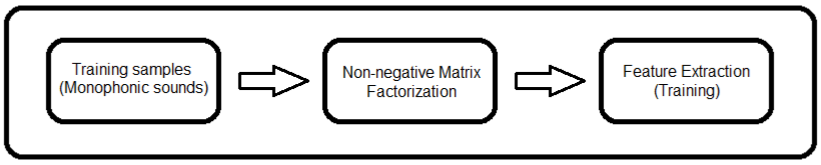
Training samples (Monophonic sounds)
Non-negative Matrix Factorization
Feature Extraction (Training)

Figure 4.2: Inner processes of the training module

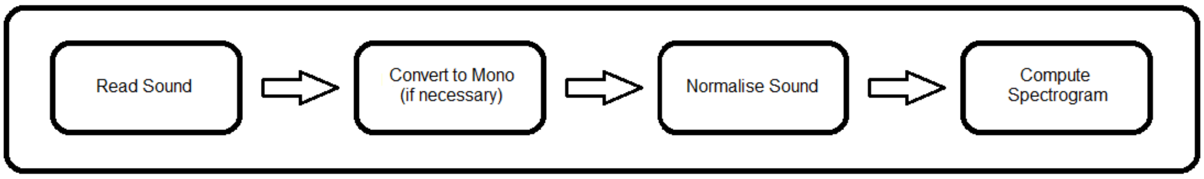

Figure 4.3: Sound processing

## 4.2 System details

### 4.2.1 Training Module

In this section we will cover the three main processes that, as depicted in figure 4.2, compose the training module of the system. It's important to note that this module is the basis of the system, as without a proper training feature set the results that can be obtained suffer greatly.

#### 4.2.1.1 Training Samples

An important step is the definition of the conceptual classes that we want to represent, that is, which types of sounds we want to identify. From there, we can start to construct a database of sounds that define these concepts, for instance, single notes from musical instruments or ambient sounds, such as birds, cars, water, and so on. Each sound needs to be processed, figure 4.3, for the system to be able to use the data.

Each sound is read to memory and converted to mono (one audio channel) if it was recorded in stereo (two audio channels), this conversion is done by calculating the sum of the values on both channels and dividing by two, an important step as we only need to handle one audio channel in terms of the calculations that will be done (from a practical viewpoint, this only transforms the spatialization of the sound, which may be distributed along the stereo panorama, to a centred hearing point). The sound is then normalized to uniformise the amplitude of the audio signal – this guarantees that all the sounds have a similar volume level, reducing the possibility of discrepancies in terms of spectral properties.

We then calculate the magnitude spectrogram of the normalized mono sound using the value of its *frequency rate* $* x$ for both the segment size, and the number of frequency points used to calculate the discrete Fourier transforms, the overlap between segments is given by $(\textit{frequency rate} * x) - (\textit{frequency rate} * x)/2$, where $x$ is 0.04. The value of $x$ is used to determine the segment sizes according to a percentage of the frequency rate, simplifying any adjustments that may be necessary when using very small sound files.

After we apply this process to all the sounds, we create a matrix **S** by horizontally concatenating each of the computed magnitude spectrograms. Additionally, we construct a matrix **C** that stores the time length of each spectrogram – an aid in the feature extraction process later on, as it basically gives us the indexes that identify each of the training sounds in the matrix **S**.

#### 4.2.1.2 Non-negative Matrix Factorization

As mentioned before, given a non-negative matrix **S**, NMF calculates two, also non-negative, matrices $\mathbf{\Theta}$ and **P**, as defined in equation 2.1. In this case, the matrix **S** is the magnitude spectrogram of one or more sound signals (be it the series of concatenated spectrograms, as calculated in section 4.2.1.1, or a single spectrogram, such as in section 4.2.2.1). The two matrices $\mathbf{\Theta}$ and **P** represent, respectively, the mixing matrix (containing the spectra of the sounds) and the source matrix (containing the temporal envelopes of the sounds). Each matrix has a specific use: **P** will be processed in the following feature extraction phase (section 4.2.1.3), and $\mathbf{\Theta}$ will later be crucial for the testing phase (section 4.2.2.2).

Regarding its application, the NMF algorithm requires the definition of two parameters: the cost function, for which we use a divergence function, and the update rule, for which we use a multiplicative update. Additionally, it takes as its input the number of sounds that were used to create these spectrograms.

#### 4.2.1.3 Training Feature Extraction

To obtain the feature's values, we use matrix **C**, calculated during the training sample processing, and matrix **P**, calculated in the previous step.

Essentially, each pair of indexes stored in matrix **C** defines the section of matrix **P** that represent a training sound. That is, throughout all the dimensions of **P**, in that interval, are contained the values of the spectral properties that define the sound.

So, to characterize each sound, we first determine the submatrix of matrix **P** that represents it. As said above, this subset can be found using the indexes in matrix **C**, which indicate the interval of columns in **P** where there values are present. In each of

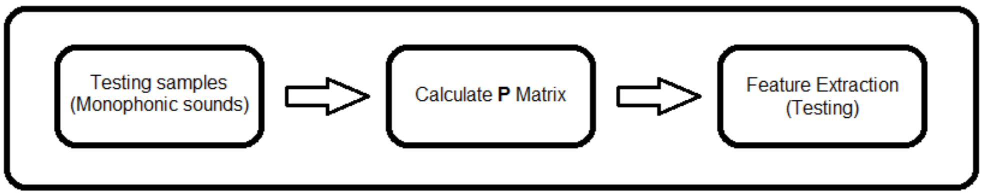
Testing samples (Monophonic sounds)
Calculate **P** Matrix
Feature Extraction (Testing)

Figure 4.4: Inner processes of the testing module

these segments we have the values of the spectral properties of a given sound throughout the dimensions of the matrix. We create the feature vector of that sound by obtaining the highest peak value from all its lines/dimensions, and then the values of the peaks found in each of the other dimensions that are in close proximity to this main peak. We define this interval by determining the columns within a lower and upper bound of 5% of the total number of columns in the subset matrix.

This process creates a training matrix composed of the feature vectors of the individual sounds which will later be used as the feature space of the classifier.

### 4.2.2 Testing Module

With the training complete, we can proceed to classify new test sounds. The testing module is, akin to the training module, composed by three processes, as can be seen in figure 4.4, which will be described in this section.

#### 4.2.2.1 Testing Samples

The samples that we want to classify can be either individual sounds or sequences of sounds. The audio processing of these sounds is equal to what we employ in the training of the system, as shown in figure 4.3, that is, the sound is read, if it is in stereo we convert it to mono, next it is normalized, its spectrogram is calculated (using the same parameters as those described in section 4.2.1.1) and finally we compute the modulus of the spectrogram.

The main difference to what we did in section 4.2.1.1 is that this process always produces a single spectrogram, be the testing sample a single sound or a sequence of sounds. The spectrogram is then passed to the next stage, where it will be used as input for the following process.

#### 4.2.2.2 Computing the test matrix $\mathbf{P}_T$

Logically, the testing features values are extracted from matrix $\mathbf{P}_T$ that we obtain from a test sample. However, differently from the training process, we do not use the NMF algorithm to produce this matrix. Instead, we rely on the mathematical manipulation of equation 2.1 to obtain

$$\mathbf{P}_T = \mathbf{\Theta}^{-1}\mathbf{S}_T \quad , \tag{4.1}$$

where $\mathbf{\Theta}^{-1}$ is the pseudoinverse of $\mathbf{\Theta}$.

This allows us to use the $\mathbf{\Theta}$ calculated in section 4.2.1.2 to project the spectral properties contained in $\mathbf{S}_T$, the magnitude spectrogram from the test sound, to the same functions that define the training data. By doing so, we create a test $\mathbf{P}_T$ matrix that is comparable to the one we obtained in the training process, allowing us to extract feature vectors that can be used to classify the sound via the feature space created in section 4.2.1.3.

#### 4.2.2.3 Testing Features Extraction

Given a test matrix $\mathbf{P}_T$, calculated in the previous step for the testing samples, we can proceed to obtain the features of the test sounds. While the concept is the same as described in section 4.2.1.3, that is, we want to obtain a feature vector for each of the sounds with values from every dimension of matrix $\mathbf{P}$ that can be used to define it.

The main difference in this case is that we do not have a way of knowing beforehand the timeframes of each sound (as we had with matrix $\mathbf{C}$ in section 4.2.1.3), since the test samples try to mirror a real world scenario. So, to determine when a sound is produced, we detect the highest peaks in each dimension of matrix $\mathbf{P}$, and create a feature vector (using the same process described in section 4.2.1.3) for each of the peaks detected in this manner. The peak detection algorithm is very simple, it considers a point as a maximum peak if it has the maximal value and was preceded (to the left) by a value lower than a given $\Delta$.

With the feature vectors calculated for all the sounds detected, we proceed to use the vector as input for the classifier.

### 4.2.3 Classifier

We decided to use a $k$-NN classifier with a Euclidean distance metric, as the studies we have seen show that it provides the all-around best results – even though it is one of the simplest and most direct of the many classification algorithms.

The usage of the classifier is totally standard. We use the training matrix obtained in section 4.2.1.3 as the feature space, which we then use to classify the features of the testing samples we processed in section 4.2.2.3. This produces a matrix with the $k$

number of nearest neighbours for each test sound, which we then use to classify the tests sounds by assigning them to the class of their most occurring neighbours.

### 4.2.4 Data Analysis

While not functionally relevant to the system in itself, this final module is important as it automatically compiles the results in a form that is much easier to interpret and analyse. The system creates a series of spreadsheets, one for each test set, that give information about each sound that was tested. This information consists of the neighbours calculated by the classifier for the sound, the percentage of these neighbours that are effectively correct, the class that was assigned to the sound and its correctness, as well as general percentage of the classification correctness for the given data set.

## 4.3 Example

In this section we provide a walk-through of the system by detailing a very simple use case. This use case will consist of using as training data a reduced dataset of three different notes from three different instruments. We then use the recognition system on another three notes from other recordings.

### 4.3.1 Data Input

The necessary input data for the system consists of two types of audio samples in wave format, training and test data. The following subsections will describe how they differ and how the system handles the data.

#### 4.3.1.1 Training Data

The raw data necessary for training must be composed of individually recorded musical notes played by any number of distinct instruments types. In a general sense, these recordings should be made with both various instruments of the same type (for instance, from different manufacturers) and a varied number of playing articulations. This heuristic is due to the knowledge that a wider range of data leads to a more accurate representation of the instruments properties, leading to better recognition results.

In this example, three single notes are used to train the system – $A_5$, $F_3$ and $C_4$ from flute, guitar and piano, respectively, played using different articulations.

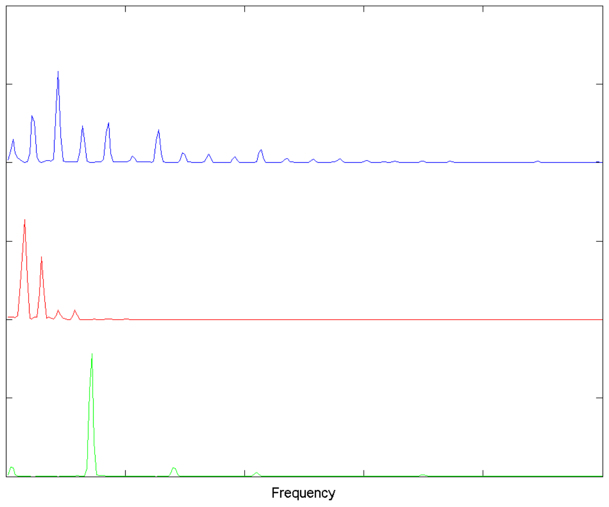

Figure 4.5: Example of a $\mathbf{\Theta}$ matrix computed with NMF of the spectrograms of three different notes (from top to bottom: piano – $C_4$, guitar – $F_3$ and flute – $A_5$)

#### 4.3.1.2 Test Data

Similarly to the training data, the test data is composed by audio samples than can contain both several or individual musical notes from different instruments.

The testing data for our example will consist of the same three notes used to train the system, with the distinction of being from different recordings and being presented as a single audio file.

### 4.3.2 Feature Extraction

The features used by the system are obtained through two steps: the data is first prepared for feature extraction and afterwards the actual features are calculated. The methods used to complete these steps are different for the two types of input data (training and test).

#### 4.3.2.1 For Training Data

After the sound is processed, as detailed in section 4.2.1.1, we apply the NMF algorithm to create the matrices $\mathbf{\Theta}$ and $\mathbf{P}$. The matrix $\mathbf{\Theta}$ contains the spectral basis functions learned by NMF of the spectrograms of the three different notes, figure 4.5, we call these spectra (or features) $\mathbf{\Theta}_{C_4,piano}$ (top row), $\mathbf{\Theta}_{F_3,guitar}$ (middle row), and $\mathbf{\Theta}_{A_5,flute}$ (bottom row). Using $\mathbf{P}$, we calculate the training features as explained in section 4.2.1.3.

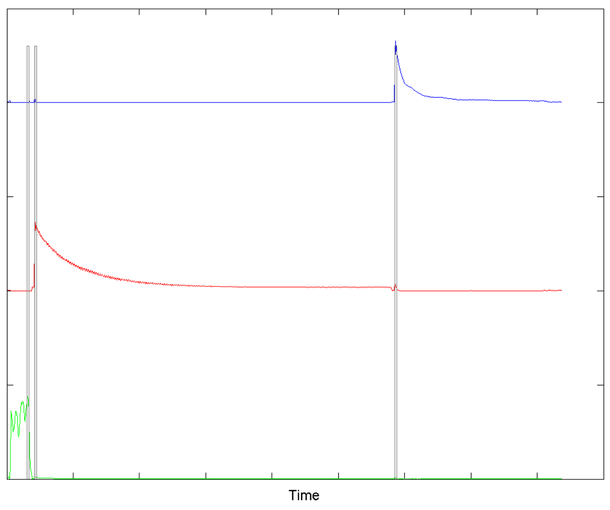

Figure 4.6: The temporal envelopes in **P** obtained by NMF of the spectrograms of three different notes (from top to bottom: piano – $C_4$, guitar – $F_3$ and flute – $A_5$)

| Data | | Features | | |
|---|---|---|---|---|
| Instrument | Note | $\mathbf{\Theta}_{A_5,flute}$ | $\mathbf{\Theta}_{F_3,guitar}$ | $\mathbf{\Theta}_{C_4,piano}$ |
| Piano | C4 | 0,0184 | 0,0000 | 4,3936 |
| Guitar | F3 | 0,3636 | 3,6361 | 0,1087 |
| Flute | A5 | 3,2725 | 0,4984 | 0,01562 |

Table 4.1: Training feature vectors for the training samples: piano – $C_4$, guitar – $F_3$ and flute – $A_5$

| Data | | Features | | |
|---|---|---|---|---|
| Instrument | Note | $\mathbf{\Theta}_{A5,flute}$ | $\mathbf{\Theta}_{F3,guitar}$ | $\mathbf{\Theta}_{C4,piano}$ |
| Piano | C4 | 0,0232 | 0,0121 | 4,8865 |
| Guitar | F3 | 0,2387 | 3,9605 | 0,0693 |
| Flute | A5 | 2,7487 | 0,3833 | 0,0618 |

Table 4.2: Test feature vectors for the test samples: piano – $C_4$, guitar – $F_3$ and flute – $A_5$

The idea behind this process can be observed in figure 4.6 – each line represents a dimension of the **P** matrix and each grey rectangle, determined by the largest peak of each dimension, represents the segment of the matrix from where the remaining peaks will be determined. Each feature vector consists of three coefficients, one for each of the features: $\mathbf{\Theta}_{A5,flute}$, $\mathbf{\Theta}_{F3,guitar}$, and $\mathbf{\Theta}_{C4,piano}$. This process produces the values presented in table 4.1, which will be used as the feature space for the classifier.

#### 4.3.2.2 For Test Data

Similarly to the training process, the sound is first processed, as explained in section 4.2.2.1. This allows us to obtain the test **P** matrix by the method covered in section 4.2.2.2. Next, the feature vectors of each note can be determined as detailed in section 4.2.2.3, This process produces the values shown in table 4.2 that are used to represent the test sound.

## 4.3.3 Classifier

For this example we used a $k$ of one, as it the maximum number of correct neighbours a test sample can have. By using the values of table 4.1 as the feature space for the $k$-NN classifier, we can test the values of table 4.2. Unsurprisingly, the classifier correctly classifies the three tests notes, a trivial task in this example, as it can easily be observed that the test features are extremely close to those of the training feature space.

# 5 Results

## 5.1 Sample description

Several types of samples were used to test the system. For our main focus, we completed several recordings of three different instruments. Additionally, we adapted our system for the recognition of ambient sounds by making minor adjustments to the feature extraction process, for which we used real world audio extracts to test its efficiency.

### 5.1.1 Musical Instrument Samples

We recorded individual notes spanning over several octaves using three different instruments and various recording conditions: guitar (a Fender Stratocaster Electric Guitar), piano (a Yamaha Grand Piano) and flute (a Hohner Recorder/English Flute).

Two recording methods were employed to obtain these samples. Piano and flute were recorded via a microphone (AKG C1000S) connected to an USB Audio Interface (Edirol UA-25). The guitar was initially recorded by connecting it directly to the audio input of a PC (this option, while extremely simple, provided the necessary data to successfully complete initial viability tests, even though most samples have a very low volume and a considerable amount of static noise, due to the absence of a pre-amplifier and a specialized analog-to-digital converter in the PC's sound card) and afterwards by connecting it to the Audio Interface (which, of course, provided a much more reliable sound recordings). In all cases, Audacity was used to capture and edit the audio. The

| Group | Flute | Guitar | Piano |
|---|---|---|---|
| **1** | $\text{Long}_1$ | $\text{Bridge}_1$ | $\text{Normal}_1$ |
| **2** | $\text{Short}_1$ | Bridge + Middle | $\text{Attack}_1$ |
| **3** | $\text{Long}_2$ | Middle | $\text{Soft}_1$ |
| **4** | $\text{Short}_2$ | Middle + Neck | $\text{Normal}_2$ |
| **5** | $\text{Long}_3$ | Neck | $\text{Attack}_2$ |
| **6** | $\text{Short}_3$ | $\text{Bridge}_2$ | $\text{Soft}_2$ |

Table 5.1: Composition of the test sound groups.

editing of the audio consisted solely in removing any detected audio silence at the beginning and/or end of the recordings.

In all, sixty six notes (spanning four octaves) were recorded, of which thirty ($E_2$ to $A_4$ for the first recording and $C_3$ to $B_3$ were rerecorded in the second) are guitar, twenty four ($F\sharp_3$ to $F_5$) are piano and twelve ($C_5$ to $B_5$) are flute. Different recordings sets were made for each note: eleven for guitar (one for each guitar pickup and an additional recording for the bridge pickup with different guitar strings), six for flute (three sets of two different playing techniques - sustain and staccato) and piano (two sets of three different playing techniques - strong sustain, soft sustain and staccato).

### 5.1.2 Ambient Sound Samples

These sounds consist of several recordings that represent four different concepts, "water", "car", "train" and "people". A total of twenty recordings were extracted, seven for "water" for a total of one minute and ten seconds, six for "car" for a total of one minute and forty two seconds, five for "train" for a total of eight minutes and seventeen seconds and two for "people" for a total of two minutes and twenty six seconds. These clips where extracted from select intervals of audio obtained from video footage by matching the visual concepts with sounds that can unambiguously identify this concepts. Unfortunately, we could not determine the recording equipment that was used, nor the exact recording conditions, for any of the sound clips.

## 5.2 Musical Instrument Recognition

The main focus of our tests lies in this section, with the recognition of musical instruments and musical notes.

We use six different data sets for our tests. This data sets were created by using a leave-one-out cross-validation scheme. In other words, we rotate through several sound groups to create our training set and our testing set. Table 5.1 presents the

| | **Tests** | | | | | |
|---|---|---|---|---|---|---|
| **Note** | **Set 1** | **Set 2** | **Set 3** | **Set 4** | **Set 5** | **Set 6** |
| $C_5$ | Correct | Correct | Correct | Correct | Correct | Correct |
| $C\sharp_5$ | Correct | Correct | Correct | Correct | Correct | Correct |
| $D_5$ | Correct | Correct | Correct | Correct | Correct | Correct |
| $D\sharp_5$ | Correct | Correct | Correct | **$E_5$** | Correct | **$E_5$** |
| $E_5$ | Correct | Correct | Correct | Correct | Correct | Correct |
| $F_5$ | Correct | Correct | Correct | Correct | Correct | Correct |
| $F\sharp_5$ | Correct | Correct | Correct | Correct | Correct | Correct |
| $G_5$ | Correct | Correct | **$F\sharp_5$** | Correct | Correct | Correct |
| $G\sharp_5$ | Correct | Correct | Correct | Correct | Correct | Correct |
| $A_5$ | Correct | Correct | Correct | Correct | Correct | Correct |
| $A\sharp_5$ | Correct | Correct | Correct | Correct | Correct | Correct |
| $B_5$ | Correct | Correct | Correct | Correct | Correct | Correct |
| **Totals** | 100% | 100% | 92% | 92% | 100% | 92% |

Table 5.2: Flute notes recognition results

groups of sounds that we used with their different articulations and recordings (different subscripts indicate different recording takes). The training sets are always composed of five different groups, leaving the remaining group as the testing set, for a total of six different scenarios to gather results. This allowed us to thoroughly test our system with distinct types of sounds, giving the possibility of obtaining results in scenarios that try to mirror real world conditions.

As the maximum number of correct neighbours a test sound can have is five, due to our definition of five groups for the training set, we use this value as the *k* used in the *k*-NN classifier.

Finally, for each type of tests, the results are presented in table form with a series of commentaries on any significant details. When applicable, the tables will detail which notes were correctly identified and present which notes were incorrectly identified, and which note the system obtained.

### 5.2.1 Individual Instrument Recognition

Each of these test results were compiled using only one type of musical instrument sounds to verify the efficiency of the recognition of individual musical notes.

The tables for this section present the notes correctly and incorrectly recognized for each of the test sets. Incorrect notes are displayed with the (erroneous) note that was recognized by the system.

For flute, a total of twelve notes were tested, spanning over one octave, from $C_5$ to $B_5$. Of the six test sets, three had a success rate of 100%, while other three had 92%.

| | **Tests** | | | | | |
|---|---|---|---|---|---|---|
| **Note** | **Test 1** | **Test 2** | **Test 3** | **Test 4** | **Test 5** | **Test 6** |
| $E_2$ | Correct | Correct | Correct | Correct | Correct | Correct |
| $F_2$ | Correct | Correct | Correct | Correct | Correct | Correct |
| $F\sharp_2$ | Correct | Correct | Correct | Correct | Correct | Correct |
| $G_2$ | Correct | Correct | Correct | Correct | Correct | Correct |
| $G\sharp_2$ | Correct | Correct | Correct | Correct | Correct | Correct |
| $A_2$ | Correct | Correct | Correct | Correct | Correct | Correct |
| $A\sharp_2$ | Correct | Correct | Correct | Correct | Correct | Correct |
| $B_2$ | Correct | Correct | Correct | Correct | Correct | Correct |
| $C_3$ | Correct | Correct | **$C\sharp_3$** | Correct | Correct | Correct |
| $C\sharp_3$ | Correct | Correct | Correct | Correct | Correct | Correct |
| $D_3$ | Correct | Correct | Correct | Correct | Correct | Correct |
| $D\sharp_3$ | Correct | Correct | Correct | Correct | Correct | Correct |
| $E_3$ | Correct | Correct | Correct | Correct | Correct | Correct |
| $F_3$ | Correct | Correct | Correct | Correct | Correct | Correct |
| $F\sharp_3$ | Correct | Correct | Correct | Correct | Correct | Correct |
| $G_3$ | Correct | Correct | Correct | Correct | Correct | Correct |
| $G\sharp_3$ | Correct | Correct | Correct | Correct | Correct | Correct |
| $A_3$ | Correct | Correct | Correct | Correct | Correct | Correct |
| $A\sharp_3$ | Correct | Correct | Correct | Correct | Correct | Correct |
| $B_3$ | Correct | Correct | Correct | Correct | Correct | Correct |
| $C_4$ | Correct | Correct | Correct | Correct | Correct | Correct |
| $C\sharp_4$ | Correct | Correct | Correct | Correct | Correct | Correct |
| $D_4$ | Correct | Correct | Correct | Correct | Correct | Correct |
| $D\sharp_4$ | Correct | Correct | Correct | Correct | Correct | Correct |
| $E_4$ | Correct | Correct | Correct | Correct | Correct | Correct |
| $F_4$ | Correct | Correct | Correct | Correct | Correct | Correct |
| $F\sharp_4$ | Correct | Correct | Correct | Correct | Correct | Correct |
| $G_4$ | Correct | Correct | Correct | Correct | Correct | Correct |
| $G\sharp_4$ | Correct | Correct | Correct | Correct | Correct | Correct |
| $A_4$ | Correct | Correct | Correct | Correct | Correct | Correct |
| **Totals** | 100% | 100% | 97% | 100% | 100% | 100% |

Table 5.3: Guitar notes recognition results

| | **Tests** | | | | | |
|---|---|---|---|---|---|---|
| **Note** | **Set 1** | **Set 2** | **Set 3** | **Set 4** | **Set 5** | **Set 6** |
| F♯$_3$ | Correct | Correct | Correct | Correct | Correct | Correct |
| G$_3$ | Correct | Correct | Correct | Correct | Correct | Correct |
| G♯$_3$ | Correct | Correct | Correct | Correct | Correct | Correct |
| A$_3$ | Correct | Correct | Correct | Correct | Correct | Correct |
| A♯$_3$ | Correct | Correct | Correct | Correct | Correct | Correct |
| B$_3$ | Correct | Correct | Correct | Correct | Correct | Correct |
| C$_4$ | Correct | Correct | Correct | Correct | Correct | Correct |
| C♯$_4$ | Correct | Correct | Correct | Correct | Correct | Correct |
| D$_4$ | Correct | Correct | Correct | **D$_5$** | Correct | Correct |
| D♯$_4$ | Correct | Correct | Correct | Correct | Correct | Correct |
| E$_4$ | Correct | Correct | Correct | Correct | Correct | Correct |
| F$_4$ | Correct | Correct | Correct | Correct | Correct | Correct |
| F♯$_4$ | Correct | Correct | Correct | Correct | Correct | Correct |
| G$_4$ | Correct | Correct | Correct | Correct | Correct | Correct |
| G♯$_4$ | Correct | Correct | **A♯$_4$** | Correct | Correct | Correct |
| A$_4$ | Correct | Correct | Correct | Correct | Correct | Correct |
| A♯$_4$ | Correct | Correct | Correct | Correct | Correct | Correct |
| B$_4$ | Correct | Correct | Correct | Correct | Correct | Correct |
| C$_5$ | Correct | Correct | Correct | Correct | Correct | Correct |
| C♯$_5$ | Correct | Correct | Correct | Correct | **C♯$_4$** | Correct |
| D$_5$ | **D$_4$** | Correct | Correct | **D$_4$** | **D$_4$** | **D$_4$** |
| D♯$_5$ | Correct | Correct | Correct | Correct | Correct | Correct |
| E$_5$ | Correct | Correct | Correct | Correct | Correct | Correct |
| F$_5$ | Correct | Correct | Correct | Correct | Correct | **F$_4$** |
| Totals | 96% | 100% | 96% | 92% | 88% | 92% |

Table 5.4: Piano notes recognition results

| | **Tests** | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **Set 1** | | **Set 2** | | **Set 3** | | **Set 4** | | **Set 5** | | **Set 6** | |
| **Inst.** | Class | Note | Class | Note | Class | Note | Class | Note | Class | Note | Class | Note |
| Flute | 100% | 75% | 100% | 83% | 92% | 83% | 100% | 92% | 92% | 92% | 100% | 92% |
| Guitar | 100% | 100% | 100% | 100% | 100% | 100% | 100% | 100% | 100% | 100% | 100% | 100% |
| Piano | 96% | 100% | 100% | 100% | 92% | 100% | 96% | 100% | 100% | 100% | 88% | 96% |

Table 5.5: Recognition results for all the instruments

| | **Flute** | **Guitar** | **Piano** | **Total** |
|---|---|---|---|---|
| **Flute** | **70** | *0* | *2* | 72 |
| **Guitar** | *0* | **72** | *0* | 72 |
| **Piano** | *6* | *1* | **137** | 144 |

Table 5.6: Confusion matrix for the instrument recognition

In total, only three notes in seventy two where identified incorrectly, each one in a different test (table 5.2).

For guitar, a total of thirty notes were tested, spanning over three octaves, from $E_2$ to $A_4$. Of the six test sets, five had a success rate of 100%, while the other had 97%. Only one note in one hundred and fifty was identified incorrectly (table 5.3).

For piano, a total of twenty four notes were tested, spanning over three octaves, from $F\sharp_3$ to $F_5$. Of the six test sets, only one had a success rate of 100%, while the others ranged from 88% to 96%. Eight notes in one hundred and forty four were identified incorrectly, however, seven of these were only wrong in terms the octave, the fundamental note was correctly identified (table 5.4).

The results we obtained for each individual instrument were nearly flawless, providing a strong basis for the testing scenario with multiple instruments.

## 5.2.2 Multiple Instrument Recognition

The results presented in this section are for tests conducted by training the system with all three instruments.

#### 5.2.2.1 Individual Instrument Tests

A total of forty eight notes spanning over three octaves were tested. Twelve for flute, another twelve for guitar and twenty four for piano. The detailed results are presented in table 5.5, the recognition rates for both classes and notes for all test sets, and table 5.6, a confusion matrix presenting the number of sounds that were misclassified.

From the three instruments, "flute" had the overall lowest recognition rate in terms of identifying notes, while "piano "had the highest misclassification rate in terms of instrument class identification. "Guitar" had a 100% recognition rate on both accounts.

| | **Tests** | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **1** | | **2** | | **3** | | **4** | | **5** | | **6** | |
| **Inst.** | Class | Note | Class | Note | Class | Note | Class | Note | Class | Note | Class | Note |
| Flute | 5/6 | 6/6 | 4/5 | 5/5 | 4/5 | 5/5 | 2/3 | 3/3 | 5/5 | 5/5 | 3/3 | 3/3 |
| Guitar | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 3/3 | 3/3 | 4/4 | 4/4 |
| Piano | 9/9 | 9/9 | 10/10 | 10/10 | 10/10 | 10/10 | 12/12 | 12/12 | 12/12 | 12/12 | 12/13 | 12/13 |

Table 5.7: Recognition results for the random sound clips

The comparatively low values for the recognition of "flute" notes is due to the flute sounds being almost purely sinusoidal (as can be seen by analysing their spectra, for instance, the bottom line of Figure 4.5). This indicates that our use of peaks obtained from the temporal characteristics of the sound are not optimal. Nevertheless, the recognition rates we obtained were very high and consistent, showing the validity of this approach.

#### 5.2.2.2 Randomly Generated Sound Clips

The sounds clips were automatically generated by randomly choosing twenty notes from the testing set and merging them in a single continuous file with some seconds of silence between notes. The purpose of this test is twofold – firstly, it allowed us to study the behaviour of the peak detection algorithm, and secondly, to observe how NMF would handle a single spectrogram containing various notes (instead of a single spectrogram per note, as was tested so far).

In table 5.7 we present, for each test, the fraction of notes and classes that were correctly classified. We favoured this approach over presenting the percentages, due to notion that the varying number of notes selected from each instrument would be obscured, giving a skewed vision of the results.

By analysing the results, we can observe that the recognition rate is on par with what was achieved in previous tests. From a total of 120 notes, only one note was misclassified (piano in the last test group) and only four notes were misclassified in terms of their instrument class.

## 5.3 Ambient Sounds Recognition

As previously mentioned, we required some adjustments to adapt the system for ambient sound recognition. Two of these changes affect both the training and testing processes, while the third alters the way in which we determine the value of $k$ used for the $k$-NN classifier. All the other processes in the system were unchanged from what was described in section 4.

The first change modified the spectrogram creation; instead of calculating one for each sound and concatenating all the spectrograms obtained in this manner, we cut each sound in twenty segments of 0.2 seconds, taken uniformly throughout its length (if the sound is shorter than 4 seconds, the segments will overlap) and concatenate all these spectrograms. Our choice of 0.2 seconds for the sound segments is mainly due to the limited number of sound samples we had available, however, we found that it also produced better results than when using longer segments.

The second modification was to the feature extraction. We create a feature vector for each of the 0.2 second segments by calculating three distinct features for each of the matrix' dimensions in that segment: the average value and median of the spectrum, as well as the spectral energy (the sum of the values in a dimension).

Finally, the value of $k$ used for the $k$-NN classifier is given by $ceiling(\sqrt{mean(sounds)})$, where $sounds$ is a vector with the number of training samples for each of the concepts used. So, for instance, if we use three concepts where two of these have 15 samples and the remaining one has 20 samples, we would have $ceiling(\sqrt{mean([15, 15, 20])})$, which would give us a $k$ of 8. This calculation stems from the necessity of a dynamic value for $k$, due to the inconstant number of training samples for each group. We found, empirically, that this formula worked well for a variable number of samples, as it consistently generated adequate values of $k$.

The following tests were completed using two data sets. The first set is composed of sounds representing three different concepts, "water", which consists of sound of flowing water (rivers, fountains, etc.), "car", sounds of cars driving through streets, and "train", sounds recorded inside a moving train. The training samples consist of four sounds for "water", for a total of 51.910 seconds, four sounds for "car", for a total of 1 minute and 17.380 seconds and four sounds for "train", for a total of 3 minutes and 37.484 seconds, and the testing samples of three sounds for "water", for a total of 18.903 seconds and 96 segments, two sounds for "car", for a total of 24.975 seconds and 125 segments, and one sound for "train", for a total of 4 minutes and 40.204 seconds and 1400 segments. The second set uses the same sounds as the previous one, but adds the concept "people", which consists of sounds of large gatherings of people (a soccer stadium and a large meeting of people). We used only one sound with the length of 23.472 seconds for training, and another for testing, with 2 minutes and 2.984 seconds and a total of 615 segments.

### 5.3.1 "Water", "car" and "train" recognition

The results for this data set are presented in table 5.8, which shows the recognition rates in terms of the whole audio samples, and table 5.9, that shows the recognition

| | Water | Car | Train | Total |
|---|---|---|---|---|
| **Water** | **2** | 1 | 0 | 67% |
| **Car** | 0 | **2** | 0 | 100% |
| **Train** | 0 | 0 | **1** | 100% |

Table 5.8: Confusion matrix for the complete samples of data set 1

| | Water | Car | Train | Total |
|---|---|---|---|---|
| **Water** | **73** | 23 | 0 | 76% |
| **Car** | 8 | **111** | 6 | 89% |
| **Train** | 0 | 317 | **1083** | 77% |

Table 5.9: Confusion matrix for the sound segments of data set 1

rates in terms of the total number of segments obtained from the audio files.

For this data set, only one of the complete samples belonging to the "water" class was misclassified, lowering the recognition rate of the class to 67%. By observing the results for each of the 0.2 second segments, it's visible that most were correctly classified for this class. While the rates for the other classes were lower, the accuracy was substantially high with over 75% for all the classes.

### 5.3.2 "Water", "car", "train" and "people" recognition

The recognition rates are presented in table 5.10 and table 5.11 in similar fashion to the results of the first data set.

Even with the addition of a new class "people", the system continued to produce overall good results. Two complete samples were misclassified, one of the "car" class and the same sample from the "water" class that misclassified in the first data set. However, by analysing the recognition rates for the sound segments, we notice that the "car" class was the only one that had a decrease in accuracy, due to a number of segments being misclassified as "people" (an unexpected result, as the sounds do not seem at all similar), both "water" and "train" maintained similar values and the new "people" class attained a very high value (94%).

| | Water | Car | Train | People | Total |
|---|---|---|---|---|---|
| **Water** | **2** | 1 | 0 | 0 | 67% |
| **Car** | 0 | **1** | 0 | 1 | 50% |
| **Train** | 0 | 0 | **1** | 0 | 100% |
| **People** | 0 | 0 | 0 | **1** | 100% |

Table 5.10: Confusion matrix for the complete samples of data set 2

| | **Water** | **Car** | **Train** | **People** | **Total** |
|---|---|---|---|---|---|
| **Water** | **75** | 20 | 0 | 1 | 78% |
| **Car** | 7 | **61** | 6 | 51 | 49% |
| **Train** | 0 | 342 | **1053** | 5 | 75% |
| **People** | 0 | 35 | 0 | **580** | 94% |

Table 5.11: Confusion matrix for the sound segments of data set 2

# 6 Discussion and Future Work

Our approach allowed for a successful implementation of a system that obtains consistently high recognition rates for a variety of scopes, such as musical instruments, musical notes and ambient sounds. More so, the dual focus of this work, musical instrument recognition and partial transcription of musical notes, was shown to be an uncommon approach to the problems.

The developed system has several advantages, namely the possibility of using a reduced and easily computable feature set that is shown to provide extremely solid results. Compared to other monophonic works, these results are always on par, but generally better (as can be seen from the descriptions of the results in section 3). Additionally, our tests with ambient sounds also proved that the system not only can easily be adapted to tackle new problems, but can do so by using very reduced sets of training data (as we use a maximum of four seconds from each sound). This can be easily exemplified by the 94% accuracy obtained for the "people" concept in section 5.3: with only one training sound, the system was able to correctly identify five hundred and eighty segments of six hundred and fifteen.

The system, however, has some aspects that are disadvantageous. The computation time to train the system, essentially the calculation needed by the non-negative matrix factorization, is exponential to the data. While this is a problem inherent to most types of machine learning systems, it can severely encumber a rapid training of new sounds. Furthermore, the system does not handle correctly sounds that belong to concepts that were not trained in system. In other words, if we tried to classify notes played by violin, using as training data the sounds we used to test our system, the probability of the system correctly classifying the notes would be high, however, it would always assign the sounds to one of the tree classes of instruments, when, optimally, it should detect that it does not "know" that type of instrument.

As future work there are some ideas that should be considered. For one, improving the peak detection algorithm to some other, more efficient, onset detection algorithm. This would increase greatly the type of test sounds that could be used. The system could be adapted to use ICA or PCA, allowing a study, in greater detail, of the results that can be obtained with these blind source separation algorithms. Finally, we completed preliminary tests on polyphonic sounds that seem, from an empirical standpoint, highly promising. This was unsurprising, considering what other works have achieved (for example, in [SB03]), as well as the characteristics of non-negative matrix factorization. Thoroughly testing in this conditions would be very interesting, as polyphony is what better represents real world scenarios in audio analysis.

# Bibliography

[ALP03] G. Agostini, M. Longari, and E. Pollastri. Musical instrument timbres classification with spectral features. *EURASIP J. Appl. Signal Process.*, 2003:5–14, 2003.

[Aro92] B. Arons. A review of the cocktail party effect. *Journal of the American Voice I/O society*, 12:35–50, 1992.

[BKK06] E. Benetos, M. Kotti, and C. Kotropoulos. Musical Instrument Classification Using Non-Negative Matrix Factorization Algorithms and Subset Feature Selection. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 5, pages 221–224, Toulouse, France, May 2006.

[Blu04] P. Blunsom. Hidden markov models. Lecture notes, August 2004.

[Coo99] P. R. Cook, editor. *Music, Cognition, and Computerized Sound: An Introduction to Psychoacoustics*. MIT Press, Cambridge, MA, USA, 1999.

[DDK05] M. Dziubinski, P. Dalka, and B. Kostek. Estimation of musical sound separation algorithm effectiveness employing neural networks. *J. Intell. Inf. Syst.*, 24(2):133–157, 2005.

[EK00] A. Eronen and A. Klapuri. Musical instrument recognition using cepstral coefficients and temporal features. In *ICASSP '00: Proceedings of the Acoustics, Speech, and Signal Processing, 2000. on IEEE International Conference*, pages II753–II756, Washington, DC, USA, 2000. IEEE Computer Society.

[Ero01] A. Eronen. Comparison of features for musical instrument recognition. In *In Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 19–22, 2001.

[Ero03] A. Eronen. Musical instrument recognition using ica-based transform of features and discriminatively trained hmms. In *7th International Symposium on Signal Processing and its Applications*, pages 133–136, 2003.

[Fin04] M. Fingerhut. Music information retrieval, or how to search for (and maybe find) music and do away with incipits. In *IAML-IASA 2004 Congress*, Oslo, Norway, August 2004.

[HABS00] P. Herrera, X. Amatriain, E. Batlle, and X. Serra. Towards instrument segmentation for music content description: A critical review of instrument classification techniques. In *International Symposium on Music Information Retrieval*, 2000.

[HBPD03] P. Herrera-Boyer, G. Peeters, and S. Dubnov. Automatic classification of musical instrument sounds. *Journal of New Music Research*, 32:3–21(19), March 2003.

[JWR09] W. Jiang, A. Wieczorkowska, and Z. W. Ras. Music instrument estimation in polyphonic sound based on short-term spectrum match. In *A.-E. Hassanien, A. Abraham, F. Herrera (Eds.), "Foundations of Computational Intelligence. Volume 2. Approximate Reasoning". Studies in Computational Intelligence Vol. 202*, pages 259–273. Springer Berlin Heidelberg, 2009.

[JZRC09] W. Jiang, X. Zhang, Z. W. Ras, and A. Cohen. Multiple classifiers for different features in timbre estimation. In *Advances in Intelligent Information Systems (Eds. Ras, Z.W. and Tsay, L.), Studies in Computational Intelligence*. Springer, 2009.

[LLT08] P. S. Lampropoulou, A. S. Lampropoulos, and G. A. Tsihrintzis. Musical instrument category discrimination using wavelet-based source separation. In *New Directions in Intelligent Interactive Multimedia*, pages 127–136. Springer, 2008.

[LS00] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *NIPS*, pages 556–562, 2000.

[LW01] M. Liu and C. Wan. Feature selection for automatic classification of musical instrument sounds. In *JCDL '01: Proceedings of the 1st ACM/IEEE-CS joint conference on Digital libraries*, pages 247–248, New York, NY, USA, 2001. ACM.

[LW07] R. A. Lewis and A. Wieczorkowska. A categorization of musical instrument sounds based on numerical parameters. In *in Proceedings of RSEISP, LNAI*, pages 784–792. Springer, 2007.

[MBTL07] L.G. Martins, J.J. Burred, G. Tzanetakis, and M. Lagrange. Polyphonic instrument recognition using spectral clustering. *Proc. International Conference on Music Information Retrieval (ISMIR)*, 2007.

[Mit97] T. Mitchell. *Machine Learning (Mcgraw-Hill International Edit)*. McGraw-Hill Education (ISE Editions), 1st edition, October 1997.

[MK98] K. D. Martin and Y. E. Kim. Musical instrument identification: A pattern-recognition approach. *The Journal of the Acoustical Society of America, Volume 104, Issue 3*, page 1768, 1998.

[MMS06] L. Ma, B. Milner, and D. Smith. Acoustic environment classification. *ACM Trans. Speech Lang. Process.*, 3(2):1–22, 2006.

[MSS02] B.S. Manjunath, P. Salembier, and T. Sikora, editors. *Introduction to MPEG-7: Multimedia Content Description Interface*. John Wiley & Sons, Inc., New York, NY, USA, 2002.

[NSM06] G. Zoia N. Scaringella and D. Mlynek. Automatic genre classification of music content: a survey. *Signal Processing Magazine, IEEE*, 23(2):133–141, 2006.

[PK06] J. Paulus and A. Klapuri. Music structure analysis by finding repeated parts. In *AMCMM '06: Proceedings of the 1st ACM Workshop on Audio and Music Computing Multimedia*, pages 59–68, New York, NY, USA, 2006. ACM.

[Pro] The Freesound Project. `http://www.freesound.org`.

[SB03] P. Smaragdis and J. C. Brown. Non-negative matrix factorization for polyphonic music transcription. In *In IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 177–180, 2003.

[TKSW07] G. Tzanetakis, A. Kapur, W. A. Schloss, and M. Wright. Computational ethnomusicology. *Journal of Interdisciplinary Music Studies*, 1(2):1–24, 2007.

[Vir07] T. Virtanen. Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria. *IEEE Transactions on Audio, Speech, and Language Processing*, 15, 2007.

[WP05] B. Wang and M. D. Plumbley. Musical audio stream separation by non-negative matrix factorization. In *Proceedings of DMRN Summer Conference*, July 2005.

[WWSS03] A. A. Wieczorkowska, J. Wróblewski, P. Synak, and D. Ślęzak. Application of temporal descriptors to musical instrument sound recognition. *J. Intell. Inf. Syst.*, 21(1):71–93, 2003.

[You39] R. W. Young. Terminology for logarithmic frequency units. In *J. Acoust. Soc. Am.*, volume 11, pages 134–139, 1939.

[Zha01] T. Zhang. Instrument classification in polyphonic music based on timbre analysis. In *Proceedings of SPIE–The International Society for Optical Engineering, Vol. 4519*, pages 136–147, 2001.

[ZR07] X. Zhang and Z. W. Ras. Sound isolation by harmonic peak partition for music instrument recognition. *Fundam. Inf.*, 78(4):613–628, 2007.