Technological Educational Institute of Crete
School of Applied Science (Rethymnon)
Department of Music Technology & Acoustics Engineering

Audio to Score Alignment using Hidden Markov Models

By Michalis Morakeas
(Student ID 604)

Supervisors:
Chrisoula Alexandraki
Panagiotis Zervas

April 2014

Abstract

The objective of this project is the development of an Audio to Score Alignment (ASA) system, which is a computer program that, given a score of a music piece and a recording of that same piece, it can detect the point in the signal that corresponds to each musical event of the score. In other words, it can 'follow' the score by 'listening' to the recording, similarly to human listener. There are several applications sought by ASA, such as assisting digital audio editing and post-processing that often requires knowledge of the location of a particular note or phrase in the score, allowing automatic annotation in music libraries hence permitting efficient search and retrieval, assisting musical education, or more generally providing automatic audio segmentations, a task that is a prerequisite to most applications exploring musical content.

ASA is one of the several tasks targeted by Music Information Retrieval research, an interdisciplinary scientific field aiming at retrieving semantic information from digital music representations. The first chapter provides an introduction to basic music concepts and a description of several Music Information Retrieval tasks. The second chapter provides a review of relevant research initiatives on ASA and showcases some representative software applications. The third chapter is an introduction to the basic pattern recognition and machine learning techniques used by the system under investigation, emphasizing on the use of Hidden Markov Models (HMM).

Following chapter 4 presents the overall methodology and the implementation of the ASA system developed in the context of this work. Chapter 5 presents the evaluation of the implemented software and the final chapter discusses conclusions, shortcomings and future work.

Περίληψη

Ο σκοπός αυτής της πτυχιακής εργασίας είναι η ανάπτυξη ενός συστήματος Στοίχισης Ήχου και Παρτιτούρας (ΣΗΠ), το οποίο είναι ένα πρόγραμμα για υπολογιστή που, δεδομένης μιας παρτιτούρας ενός μουσικού κομματιού και μιας ηχογράφησής του, μπορεί να ανιχνεύσει το σημείο του ηχητικού σήματος που αντιστοιχεί σε κάθε μουσικό γεγονός της παρτιτούρας. Με άλλα λόγια, μπορεί να "ακολουθήσει", την παρτιτούρα, "ακούγοντας" την ηχογράφηση, ακριβώς όπως θα έκανε, ένας μουσικά εκπαιδευμένος, ακροατής. Υπάρχουν, διάφορες εφαρμογές της ΣΗΠ, όπως η διευκόλυνση της επεξεργασίας ψηφιακού ήχου που συχνά απαιτεί την γνώση της ακριβούς θέσης μιας συγκεκριμένης νότας ή φράσης της παρτιτούρας πάνω σε ένα ηχογραφημένο σήμα, η αυτόματη επισημείωση σε μουσικές βιβλιοθήκες με στόχο την εστιασμένη πρόσβαση σε μουσικό περιεχόμενο, η βοήθεια στην μουσική εκπαίδευση, ή γενικότερα η αυτόματη τεμαχιοποίηση ηχητικών σημάτων, μια εργασία απαραίτητη στις περισσότερες εφαρμογές που εξερευνούν μουσικό περιεχόμενο.

Στο γραπτό μέρος της εργασίας, το πρώτο κεφάλαιο παρουσιάζει μια εισαγωγή σε βασικές μουσικές έννοιες, καθώς και μια περιγραφή των διάφορων υπολογιστικών διεργασιών που μελετώνται στο ερευνητικό πεδίο της Ανάκτησης Μουσικής Πληροφορίας, στις οποίες ανήκει και η ΣΗΠ. Το δεύτερο κεφάλαιο παρέχει μια εποπτεία συναφών ερευνητικών πρωτοβουλιών στη ΣΗΠ και παρουσιάζει κάποιες ενδεικτικές εφαρμογές λογισμικού. Το τρίτο κεφάλαιο είναι μια εισαγωγή στις βασικές μεθόδους αναγνώρισης προτύπων και μηχανικής μάθησης, που χρησιμοποιούνται από το υπό υλοποίηση σύστημα, δίνοντας έμφαση στη χρήση των Κρυμμένων Μοντέλων Markov (Hidden Markov Models, HMMs).

Εν συνεχεία, το κεφάλαιο 4 παρουσιάζει τη μεθοδολογία που ακολουθήθηκε για την υλοποίηση ενός συστήματος ΣΗΠ. Το κεφάλαιο 5 περιγράφει τα πειράματα που διεξήχθησαν για την αξιολόγηση της αλγοριθμικής απόδοσης του συστήματος αυτού και τέλος στο κεφάλαιο 6 συζητιόνται συμπεράσματα, προτεινόμενες βελτιώσεις στο σύστημα που έχει αναπτυχθεί και μελλοντικές προοπτικές έρευνας στον τομέα αυτό.

Table of Contents

1. Music and Music Information Retrieval

1.1. Introduction

Ever wondered if it is possible to play a song along with your computer? Interpreting a musical piece on the piano and having your computer "listen" to your interpretation and follow your playing, providing accompaniment? Or did you ever imagine you could hum to your cellphone, a song stuck into your mind, and it would reply the song title, the artist as well as other songs and artists that share the same music genre or mood with your song? What about having a recorded piece of music and having your computer recognize all the instruments playing and print you the score of it?

Recent technological advances and the vast digitization of music have led to the development of a new, interdisciplinary research field known as Music Information Retrieval (MIR). Questions such as the above are addressed by MIR research by means of musicology and signal processing in combination with information theory and pattern recognition techniques. MIR research has presented major technological achievements in a short period of time, many impressive results and promising perspectives. The work of this field, and more specifically one of its most basic tasks, Audio to Score Alignment, is the main topic of this thesis.

1.2. Music Fundamentals

1.2.1. Basic Concepts

Music consists mainly of what we call **musical sounds** or musical tones. Musical tones are sounds produced by musical instruments with an almost steady periodic vibration and

usually it is a sum of a fundamental frequency and a number of integer multiples of it, called harmonics.

All musical tones have four basic features: pitch, intensity, timbre and duration [1].

Pitch is the feature associated with the perception of the fundamental frequency of the sound. It depicts the "height" of the sound, ordering the sounds into a range of "low" and "high" or "deep" and "acute". Although it can be expressed as a frequency, its perception is not straightforward and objective, since the human auditory system has a non linear behavior as far as pitch recognition is concerned.

Intensity is the feature associated with the perception of the amplitude of the vibration and can express how "soft" and "quiet" or how "loud" and "intense" a sound is. Loudness, which is another word for intensity, is also a subjective measure, since it is related with the non linearity of the human ear, as well as other psychoacoustic phenomena, but the basis for its approximation is the sound energy.

The physical characteristic of a sound that is associated with **timbre** is mainly that of its frequency spectrum and secondary that of its envelope. Its perception is the most difficult to be defined, but we could say that it is the feature that enables the listener to distinguish two sounds of the same pitch and intensity but different sound source. This could be a piano and a violin or the human voice of two different persons. Other information a listener can draw from pitch include the playing technique of the instrumentalist (a plucked or bow playing of a string instrument), the surrounding acoustics of the sound source (a small room or a cathedral) and the equipment used for recording or transmitting of a sound (cassette, telephone, etc.).

Duration is the amount of time the certain sound lasts. The beginning point is called onset time, while the ending point, offset time.

However, music may also be generated by **percussive sounds,** not necessarily associated with a clear perception of pitch [1]. The spectrum of percussive sounds does not have a harmonic structure and is characterized by dominating noise components.

Musical sounds can be divided into two broad categories, according to their musical content: monophonic and polyphonic sounds [1].

Monophonic sounds are those for which there is a single pitch value (musical tone) at any time. These pieces consist of one and only melody.

Polyphonic sounds are those that include more than one parallel lines of independent melody, usually performed by different musical instruments, or instruments that are capable of producing chords. Chords are formed by the simultaneous playing of two or more musical tones and they express a different quality, depending on the distance between the included pitches. They consist of a core element in most musical genres.

There are also four more concepts that are used to characterize a musical piece: tempo, tonality, time signature and key signature [1].

Tempo is the speed of the playing of a piece. It plays a significant role in music and it is measured in Beats Per Minute (BPM.).

Tonality feature is used to describe the "tonal center" of a musical work, which is defined by the note and chord that play a central role and is related to the role played by each chord and the relations between them.

Time signature defines the number of the beats included in each measure and is usually in the form of a fraction. It is related to the note accent and divides the various beats into "weak" and "strong" (arsis and thesis).

Key signature is in the form of note alterations (sharp and flat symbols) and informs the performer of the notes to be played constantly altered. It is a sign of the tonality of the musical piece.

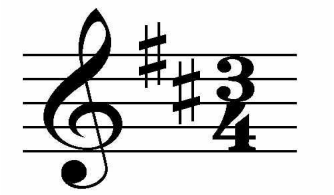

Figure 1: Example of a key and time signature

1.2.2. Music Forms

Music is an art, involving composition as well as performance. A person can be interested in any of these two procedures, when he is coming to contact with music, while their combination is of the most usual interest. Thus, communication in music, is carried out in two different levels [1], [2].

The first level is the **symbolic level**. At this level the composer, communicates to the musicians all the musical gestures that are required for the piece to be performed. The symbolic representation of all these gestures is called the score. There are numerous symbolic representation systems proposed through the centuries, but the one associated with western musical tradition and the most popular one is the one introduced by Guido d'Arezzo on the 11th century and was established in its current form on the 14th century. A formal definition of what a score is, follows [1]:

> *A musical score is a structured organization of symbols, which correspond to acoustic events and describe the gestures needed for their production.*

The information contained in a musical score can be divided into two sets of parameters: the general parameters and the local parameters.

The general parameters concern the universal properties of the musical piece that never or not often change. These include main tonality, modulations, time signatures, tempi, musical form, number of voices and instruments and repetitions. These parameters are usually not distinctive of the musical piece, since there is a wide number of pieces having the same or similar general parameters such as tonality or time signature.

The local parameters contain the sequence of the notes that have to be played by each musical instrument, along with their correct time positions and durations and maybe

intensity indications. These are the parameters that enable us to extract information about the melody, the rhythm and the harmony of the piece. The intensity indicators are presented in a subjective way, ranging from “very soft” to “very loud” while information about the timbre are only partially suggested through instrumentation and playing technique indications, when present, and are not clearly carried out.

Therefore, although the musical score works as a representation of the ideal performance, it is clearly only an approximation of the musical work, since it is impossible to represent all the parameters required.

The second level is the **performance level**. At this level, the musicians interpret the symbols of the musical score and perform the piece, communicating their interpretation to the audience. This is done by producing a personal realization of the gestures proposed within the score, that results in what is called the performance. It is the conversion of symbols into sounds, through the musicians, and a formal definition is the following [1]:

> *A musical performance is made of a sequence of gestures performed by musicians on their musical instruments; the result is a continuous flow of acoustic waves, which correspond to the vibration induced on musical instruments or produced by the human voice.*

A music score is the means used by composers to communicate their intentions to music performers. However, it is nowadays considered common sense that precise interpretation of a music score sounds mundane and machine-like. Expressive music performance necessitates deviating from the score of a music piece in a number of ways including, tempo deviations, intensity and occasionally melodic and harmonic alterations.

A recording of a performance is often the only means of reference to a specific musical work, due to the lack of a score. This happens a lot in jazz and folk music, as well as other genres with an emphasis on improvisation.

A performance can generate a more universal interest than a musical score, since it does not require the knowledge of the symbolic language, in order to be understood. But a

performance can generate special interest to specific categories of people. Such may be the musicologists, interested in studying the characteristics of the interpretation, or audio engineers, interested in studying the acoustics or the recording techniques.

On the other side, from the musician's point of view, it is far more difficult to learn how to play or study the structure of a musical piece, only by listening to a performance of it, without having access to its score.

1.2.3. Facets of Music

According to [3], music can be viewed in seven different facets: pitch, temporal, harmonic, timbral, editorial, textual and bibliographic facet. Each one of them plays various roles in the concept of understanding music. They are often correlated between them and there is a number of combinations of them, that can result in a meaningful descriptor of the musical work. These are useful for the tasks described in 1.4.2.

Pitch, as mentioned above, is associated with the fundamental frequency of a note. There is a variety of methods for depicting pitch, with note names (A, B, C, …), scale degrees (I, II, III, …) or solfege (do, re, mi, …) being among the most popular. What is worth mentioning, is the highest significance of the intervals between the pitches in a melody, over the actual pitches. This means that the distances between each note within a note sequence is what makes it distinguishable, and not the actual notes themselves. These intervals are measured in semitone distances and each interval has a specific name, such as “Major third”, “Minor second”, “Diminished fourth”, etc.

All information of a musical piece, related to the time axis falls into the **temporal** facet and is related to the horizontal dimension of music. This information includes tempo, meter, pitch duration, harmonic duration and accents and they shape the rhythmic components of the piece. Some information access problems arise in this facet, due to the various notations of tempo (absolute, general, relative).

Harmony exists when two or more sounds are played simultaneously (chords). Harmony facet is linked with polyphonic musical pieces and is related to the vertical dimension of music (the simultaneous sounds are aligned vertically on the time axis since they are played on overlapping times). A big branch of music theory is devoted to codifying specific combinations of notes, according to their intervals and their relation to the key of the musical work, as well as their sequencing (chord progression). Again, there is a number of methods for depicting chords and chord progressions, just like pitches (F major, F+, IV, etc.). In this facet, access problems arise because of the non explicit notation of chords, and the investigation of them is not always an easy task.

The **timbral** facet involves all information having a relation with tone color. One of the most important of them is orchestration, which is the enumeration of all the musical instruments that participate into the musical piece and the part each one plays. However, this information is often attributed to the bibliographic facet, which will be examined below. Other information, related to the timbral facet is the style of playing the instrument (pizzicato, muted, etc.), but again, this is related to the editorial facet too.

All information given to the performer as instructions are contained in the **editorial** facet. Such information include fingerprints, ornamentation, dynamic instructions, slurs, articulations, staccati, bowings, etc. There are several problems associated with the access to the editorial information of a musical piece. Firstly, the information can be given in either textual or iconic form. Secondly, some of the editorial information are references to the music to be played itself (e.g. basso continuo). Finally, the lack of editorial information, assuming the performer can determine them himself, only adds to the difficulties.

Textual facet is mostly referring to the lyrical content of the songs and musical works, as well as the libretti (the text of operas and musicals). What may not seem obvious is the non strict association of a lyrical snatch to a specific melody. This occurs due to the various translations or other lyrical variations of songs, as well as the different musical settings of the same lyrics.

Bibliographic facet contains all the information that cannot be derived from the content of the musical piece, and instead are about the piece. Such information include the title of

the piece, the composer, the lyric author, the publisher, the edition, the catalog number, the publication date, the performer, etc. The difficulties associated with this facet are the same with the difficulties associated with traditional bibliography.

The first four of these seven facets are related to another way to examine musical content, proposed in [1]. This is by dividing it into seven dimensions, each of which falls into a category of short-term, mid-term or long-term information. Briefly, these are:

Short-term:

- Timbre, quality of the produced sound.
- Orchestration, sources of sound production.
- Acoustics, quality of the recorded sound.

Mid-term:

- Rhythm, patterns of sounds onsets.
- Melody, sequences of notes.
- Harmony, sequences of chords.

Long-term:

- Structure, form and organization of the musical work.

1.3. Digital music

When it comes to the digital domain, there is a wide variety of formats for representing music. All these representations fall into two main categories, with regard to the levels of communication described above. The first category is the symbolic formats, and is obviously related to the symbolic level of communication. The second one is the audio or sampled formats, and is the one that captures the performance level.

1.3.1. Symbolic formats

The range of the formats for representing digital music on the symbolic level is wide, and is similar to the variety of music editing software. The formats can be categorized according to the use of a graphical representation system and the use of a markup language.

Finale and Sibelius are among the most popular choices of music notation processing software. Software of this kind, usually try to imitate the format of printed music through the graphical representation of the musical sheet.

The sequencers of Digital Audio Workstations (DAW), such as Cubase, Logic and Ableton, while usually offering the choice of representing music in a similar to the printed sheet form, offer alternative representation forms too. One of the most popular of these is the one called **piano roll** (Figure 2). The term originates from the storage mediums that were used to control automated pianos. Music is depicted in a two-axis graphical system, in perfect match to the horizontal and vertical dimensions of music, described above. The horizontal axis stands for time, while the vertical axis stands for pitch. Notes are indicated as horizontal lines of the correct height, according to their pitch and of the correct size, according to their duration. Some software, offer the ability of representing one more dimension, usually the note intensity, through the variation of the line color.

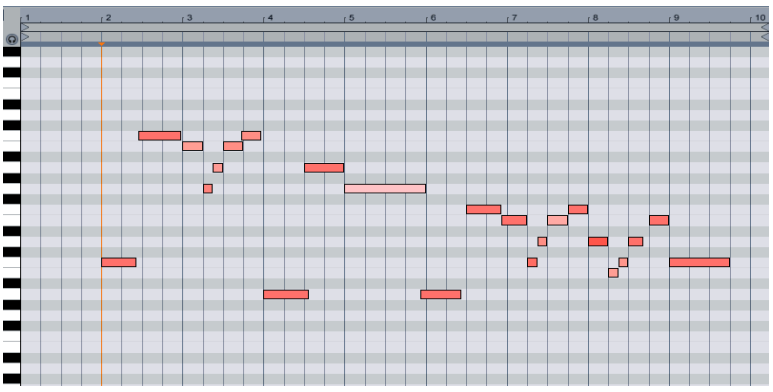

Figure 2: Example of a piano roll of a DAW (Ableton)

As one can guess, problems arise when it comes to the compatibility and the portability

between the various formats. This has been partially covered by a format called Enigma [4]. It has publicly available specifications and a number of tools for converting between that and other formats have been developed.

Markup languages is the latest advance in digital music notation. Its rising popularity is attributed mainly to the public digital music libraries. Examples of such languages are GUIDO [5], whose basic idea is representation adequacy, or the free and open-source GNU LilyPond [6]. Another attempt has been MuseData [7], but the markup language that seems to be of the most interest, is the one built upon it, MusicXML [8]. It is an extended version of XML with the aim of representing musical notation, and it is publicly and fully documented and available for use by anyone.

Finally, **Musical Instruments Digital Interface** (MIDI), whilst more than a notation system, is one of the most popular choices for saving and editing digital music in the symbolic form. It was firstly introduced in 1982, as a means of communicating between electronic musical instruments such as keyboards and sequencers and the information exchanged, were among others, which key was played, its intensity and the channel (voice/instrument). These, along with other information stored in meta-data, such as the author, title, etc. formulate the Standard Midi File (SMF) which can be used as a medium of storage, editing and reproduction of all the musical instructions for a specific musical work. Mainly, due to the great amount of available MIDI documents, this format is still one of the most popular choices for both users and musicians.

Additional information on the symbolic formats of music may be found in [1] and [2].

1.3.2. Audio formats

The goal of these formats is to digitally represent the sound of a music piece. This is achieved by the procedure called **digitization** of the audio signal and it includes sampling the signal at a specific sampling rate, quantizing it according to a specific bit rate and encoding its value. The sampling rate is the number of samples taken in an amount of time

and bit rate is the resolution (number of bits) available for encoding the amplitude value. According to the Nyquist-Shannon sampling theorem, the sampling rate should be at least two times greater than the highest frequency being sampled. Since the highest pitch, the human ear can perceive is about 20 kHz, the most popular choice of a sampling rate is 44.1 kHz. Although, the bit rate which has the greater signal-to-noise ratio is 24 bit, the most popular choice is still 16 bit. The standard method for representing a digitized audio signal is the Pulse Code Modulation (PCM). This is the one used in computers, Compact Discs and digital telephony. This is the method also used by the most popular digital audio file formats, such as WAVE, AIFF and AU.

The main disadvantage of these file formats is the great amount of information stored in them and therefore, their size. The sampling properties and the duration of the file are the only factors on that, while its content, whether it is silence or a loud noise, do not affect the size of the file. This creates the need for **compression methods**, which exploit psychoacoustic phenomena in order to reduce the amount of data within the file, removing information that seem to not be noticeable by the human auditory system. The most common file format of compressed audio is MPEG-1 Audio Layer III (MP3) [9], while an open source alternative is the OGG Vorbis format. Despite all the sophisticated compression algorithms used by these formats, there is a certain amount of loss in quality of audio, the human ear can perceive, depending on how skilled the listener is. Lossless compression formats such as Free Lossless Audio Codec (F.L.A.C.) [10] and Monkey's Audio APE [11] give solution to this problem, although the ratios of the compressed to the uncompressed file sizes are much bigger. Finally, there is another category of audio file formats that achieve a great compression rate. They are those that describe the synthesis instructions for the sound to be produced instead of the actual sampled sound. This can be perceived as the equivalent of the vector graphics files in the audio domain. Such formats have been included in the MPEG-4 and MPEG-7 standards [9].

Sampled formats are covered in [1] and [2].

1.4. Music Information Retrieval

The objective of Music Information Retrieval research is to provide efficient mechanisms for analyzing musical material (represented either on the symbolic or the performance level), so as to derive meaningful information that is useful for a number of user applications relating to music access, music performance or music composition.

The increasing availability of affordable information storage, computational processing and network technologies has largely permitted experimentation with MIR research. This translates to increased amounts of locally stored data, faster and more powerful methods of processing them, as well as increased availability and access of musical data remotely. [12]

1.4.1. Context vs Content

First of all, the various queries set for MIR, as well as the various approaches for giving answers to them are divided into two main categories: the **context-based** approaches and the **content-based** approaches.

In the category of context, the query is based on textual information describing the object. These can be the meta-data information that comes with the audio object. There are many services whose purpose is to provide the correct meta-data for music tracks. Typical examples of them are services such as Gracenote, Musicbrainz and freedb, which try to recognize a music album that the user has encoded in audio files from a compact disc or other source, relying on the number of tracks and their duration, and provide him with the accompanying information such as artist name, title, etc. These are objectively true information about the album and the music tracks, and are called **factual meta-data**.

In addition, there are other subjective information that can describe music, such as mood, emotion, style. They are called **cultural meta-data** and they implicate a personal perception of the user. In any case, the clarity, comprehensibility and accuracy of the

attached meta-data are essential for an MIR system. Web 2.0 and the rise of on-line communities have contributed considerably to outreach such problems throughout a democratic process. Users, as members of the community, attach tags with either factual or cultural information to music, which provides at least a partial validity, within the limits of the community. These information are often used by services, such as Last.fm, for categorizing and recommendation purposes. However in user provided meta-data, there is often the problem of mismatching spellings or mistaken orderings of names between users, to be surpassed. Community meta-data can also be time-aware, depicting for example the change of an artists' style or his listeners' perception.

The category of content is based on information gathered directly from the object. This means that meaningful descriptors are derived by the analysis of the various audio features. These features are extracted from the audio information using digital signal processing techniques. These can be **high-level descriptors**, referring to music elements easily perceived by humans such as musicians or even casual music listeners, or **low-level descriptors** referring to particular physical measurements of the signal.

There is a number of MIR tasks associated with the various music elements, that can result in a high-level descriptor. These include instrument recognition for timbre, melody extraction for melody, onset detection and tempo tracking for rhythm, fundamental frequency estimation for pitch, chord label extraction for harmony, modulation tracking for key, verse/chorus extraction for structure, lyrics identification and singing detection for lyrics, etc. Each of these tasks is based either on the symbolic or the sampled form of music and they will be described further on 1.4.2.

Low-level descriptors are either measurements made upon time frames of the signal (with fixed interval or aligned to beats) or statistical values of the features. Many of them are based on the spectral analysis of a windowed Fast Fourier Transform (FFT) of the signal:

$$X_k = \sum_{n=0}^{N-1} x_n e^{[-i2\pi k \frac{n}{N}]} k = 0, \ldots, N-1 \quad (1)$$

where X_k is the k -indexed sample of the signal in the frequency domain and x_n is the n -indexed sample of the signal in the time domain. Some of the most often used low-level features include Short-time magnitude spectrum, constant-Q / mel spectrum, chromagram, onset detection, Mel/log Frequency Cepstral Coefficients (MFCC), spectral flux, log power and beat tracking.

Further information on the content-based and context-based approaches of MIR can be found in [2] and [13].

1.4.2. Techniques / Tasks

As mentioned above, the first stage of an MIR system involves the extraction of the various descriptors applying the several MIR tasks either on the symbolic or the sampled form of music. In the early years of MIR the focus on the symbolic format was stronger, due to the easier concentration and transferring of MIDI files. However, the vast advances in file storage and transfer speeds, as well as the development of sophisticated algorithms have resulted in a shift towards focusing on the sampled form too. Here, we will examine how both of these forms are related to some of the main music dimensions described in 1.2.3.

The descriptors that can be easily extracted from the symbolic form of music are related to the dimensions of melody and harmony.

Melody can be derived easily when the musical piece is monophonic, since the whole symbolic sequence is a direct depiction of the melody. Problems arise together with polyphony. The easier case is that of the multiple parallel melodies, as in pieces with the element of counterpoint. These pieces are handled in a similar manner, as containing many

independent monophonic melodies. However, there are more difficult cases that require the automatic recognition of the main voice of the piece. These cases include most of the pop songs, where the main melody is usually sung by a human voice, while each of the other instruments plays an accompanying line. One of the proposed approaches is the use of statistical means associated with pitch, range, difference of subsequent notes and the relation of the voice length and the total length, based on a training set of pieces with predefined melodies. The problem becomes more difficult when voices are mixed together and there is simultaneous playing of chords and single notes. In this case, the use of listeners perception and composers organization rules have to be taken into consideration.

Although not necessary to all MIR applications that use the melody descriptor, another important task is the **segmentation** of the extracted melody. This involves sectioning the melody into basic sequences to be used as meaningful indexes, similar to sectioning a text into lexical units. However, this is not an easy process, since single notes cannot be considered as useful segments and pauses do not play the role of limits indications, as spaces do in a text. One approach is to segment all the notes sub-sequences of a specific length or to segment all the sub-sequences that are repeated a number of times. Other approaches include again the knowledge of human listeners perception rules and the attempt to emulate them.

Harmony can be extracted from the symbolic form of a musical piece in the sense of chord sequences. This is a difficult and fatiguing process even for humans, but more so, the design of a system for the automatic recognition of chord progressions involves some complicated computations. The fact that the duration of a chord is usually equivalent to the duration of a sequence of single notes makes obvious why recognition and segmentation are usually applied together. The task of chord recognition involves again statistical means and the use of training sets.

The sampled form of a musical piece offers the ability to apply tasks such as the extraction of features related to the dimensions of timbre, orchestration, rhythm, melody and harmony.

As aforementioned, **timbre** is maybe the most ambiguous dimension of music. The relatively loose timbre related instructions to instrument performers, written on musical pieces by composers, is indicative. That makes the tasks related to the extraction of meaning from it, challenging. The graph of timbre over time is called spectrogram and is derived by the FFT of the audio signal. Spectrogram itself cannot be considered as an ideal descriptor, since it contains large amounts of data and it provides little meaning. However, many low-level descriptors, such as spectral flux, spectral centroid and MFCC's are exploiting the extraction of the spectrogram. The creation of other meaningful descriptors that are computed using the spectrogram is possible and is done in accordance with the particularities of each specific MIR application.

The recognition of **orchestration**, which is the recognition of the musical instruments participating in a musical piece is one of the most difficult tasks of MIR The results are more satisfactory when the piece is monophonic, or when it is polyphonic but all the instruments are of the same type. There are also some instrument types, such as the flute, that are more easily recognizable, while others, such as the cello, are more difficult. The approaches usually involve training, based on a set of recordings of the various timbres and analysis of the MFCC features.

Although **rhythm** is considered to be the most easily recognized music dimension, in fact, this only applies to western classical music and its contemporary derivatives, due to their main focus on melody and harmony. In the eastern and African music cultures, rhythm plays a significant role and its recognition is often a much more complicated task. Tempo tracking, which is the identification of the BPM of the musical piece is the easiest task, related to rhythm, but it is significantly important in many MIR applications,. More complicated approaches that also take into consideration periodicity histograms can result in a classification of musical pieces or parts of them, according to its rhythm patterns.

The extraction of the main **melody** is a task that can be performed on the sampled domain too. The difficulty rises when the musical piece is not monophonic and the various musical instruments interfere, exactly as it does in the symbolic domain. The task of tracking the fundamental frequency of a musical piece is called f0-estimation. However,

there are often several problems encountered, that cause mistaken results. Such are the recognition of the wrong octave of a note (one higher or one lower), the addition of extra notes when vibrato or glissando is present and the mistaken recognition of a single note as a sequence of repeating the same note or vice versa. Other factors that, at least in pop music genres, contribute in the task of the main melody extraction is the assumption that the main melody is of greater amplitude than other music parts and that it is usually centrally panned between the left and the right audio channels.

The extraction of descriptors from the **harmony** dimension is another task that is performed on both the symbolic and the sampled domain. On the sampled domain, it is a difficult task, and the harmonic content descriptors that are first computed, can then be used to perform the chord progression recognition task. This is done by mapping the several spectral components to the twelve steps of a chromatic scale and comparing to some predefined chord templates.

The extraction of features related to the two remaining dimensions of music, structure and acoustics, is not so often but it consists of one of the challenges of forthcoming MIR applications.

The alignment of the symbolic and the sampled form of a musical piece is often one of the first tasks that have to be performed on an MIR system and is necessary for the realization of many other tasks. This task is the topic of this thesis and it will be fully described in the following chapters.

The techniques and tasks of MIR, as well as their relation to the dimensions of music are fully covered in [1].

1.4.3. Applications

The various MIR applications can be classified into three categories, according to their approach of managing information. These categories are music searching, music filtering and music browsing, classification and visualization, as proposed in [1].

Searching for music is the most basic application of MIR This is done by giving an example of any of the dimensions of the musical piece to look for, such as timbre, structure or harmony. However the dimension that most of the MIR work has been done, regarding music searching, is that of melody. The three approaches of a melody retrieval system are the use of index terms, the sequence matching and the geometric methods.

Searching based on index terms, offers the advantage that computations are not performed at the time of the query and instead, the query is being matched with some predefined indexes assigned to the files of the data base. This method exploits the lexical units methods described above, which can be also applied in audio information, in combination with other methods, also originating from the text information retrieval field.

The sequence matching approach consists of matching an excerpt of a piece provided by the user to the correct piece from the data base. The advantage of this approach is that it can cover the possible mismatches between the two, which is the main problem of the based on index terms approach. Such mismatches can be the insertion, deletion or modification of a note in the melody sequence. Several techniques for music searching through sequence matching have been proposed, including Dynamic Time Wrap and Hidden Markov Models (described in 2.3.2 and 3.2).

Geometric methods consider the melody as a graph on a two-dimensional space with the horizontal axis being that of time and the vertical axis being that of pitch. Each note of the melody is depicted as a dot on this space. This approach is suitable for polyphonic music, because it does not require the extraction of the main melody.

Music **filtering** is an application of recommender systems, such as on-line music stores, discography data bases and music communities. These services have the ability to suggest music to the user that he is supposed to be interested in, according to some criteria. The system can result in such a suggestion, applying a technique called collaborative filtering on gathered information concerning the user profile.

Content-based approaches are also used by filtering systems. They look for similarities

between the items that have been rated in some way by the user, and the items of the data base. The similarity can be examined in either low or high level features. Some services take advantage of the timbre features such as MFCC's, while others use descriptors such as rhythm and harmony.

A particular MIR application, associated with music filtering is the one of automatic play-list generation. This can be applied to music collections owned by users or by on-line services, like Internet radios. Each song is linked to the songs that appears to have a similarity with, in one of its dimensions, and the play-list is created, listing songs in a way that each subsequent pair of songs have such a similarity.

Apart from searching, a music collection can be accessed in terms of **browsing** its items, **classifying** them in certain categories and using **visual** cues to access them. These approaches offer the user alternatives in case, for example, a query-by-example is not sufficiently effective.

Browsing through music is carried again in terms of similarity between the various items, either they are in the symbolic or the sampled form. It is a very useful approach in cases that the user has not the ability to satisfactorily describe the item he is looking for. The navigation through the various categories of music can be organized in terms of content, such as low-level descriptors.

The most basic approach of audio classification is the separation of the audio signal into the parts where speech, music, or environmental sounds exist. What latest MIR work tries to add, is the separation of music itself, into several other classes. The classification can be applied, based on any of the dimensions of the items such as their orchestration. However, the most usual classification task is the genre classification. Pieces are separated into the various genres and sub-genres, which is the way that most users tend to understand and classify music. The size of the audio data that are necessary to have good classification results is the main concern, regarding this task.

Visualization of music can be applied on either single documents, or on whole collections of music. In the first case, the visual depiction of the musical piece works for

the user in a similar way with a thumbnail of a picture. In the case of a collection, it provides the user with a memorable picture of its collection, making it easier for him to access documents.

1.4.4. Research / Challenges

Music Information Retrieval is an ongoing research field. The number of applications and services that show up steadily is great, and so is the number of related researches and publications. Although the advances are rapid and the solutions to problems comes one after another, there is a great amount of challenges for the future.

Two major constitutions in MIR today are the International Society for Music Information Retrieval (ISMIR) and the Music Information Retrieval Evaluation eXchange (MIREX). ISMIR is an international forum on MIR research topics, which holds an annual conference since 2000. Its purpose is the gathering of researchers, educators, students and professionals and the presentation and exchange of the latest news, ideas and results on MIR. MIREX is the annual contest that is taking place along with ISMIR, where researchers propose their approaches to MIR tasks such as the following [14]:

- Audio Train/Test Tasks
    - Audio Artist Identification
    - Audio Genre Classification
    - Audio Music Mood Classification
    - Audio Classical Composer Identification
- Symbolic Genre Classification
- Audio Onset Detection
- Audio Key Detection

- Symbolic Key Detection
- Audio Tag Classification
- Audio Cover Song Identification
- Real-time Audio to Score Alignment (a.k.a Score Following)
- Query by Singing/Humming
- Multiple Fundamental Frequency Estimation & Tracking
- Audio Chord Estimation
- Audio Melody Extraction
- Query by Tapping
- Audio Beat Tracking
- Audio Music Similarity and Retrieval
- Symbolic Melodic Similarity
- Structural Segmentation
- Audio Drum Detection
- Audio Tempo Extraction

Although, the list of challenges for MIR is never ending, an attempt to enumerate the current and ongoing research direction, according to [13] and [3], follows:

- The magnification of data bases for content-based searches from thousands of pieces to millions of pieces.
- The integration of the already existing tools and frameworks.
- The separation of the sources in polyphonic music.
- The bridging of the remaining gap between the objective measurements of acoustical properties and the semantic listeners' perception.
- A stronger focus on the user side and the subjective user preferences.
- The identification of all the relevant to music kinds of information, other than audio.

- A unification of data formats and protocols and an agreement on the required quality requirements.
- The maintenance of large, open and legal data bases of accurate music data for users and researchers.

2. Audio to Score Alignment

2.1. Definition

The goal of the Audio to Score Alignment (ASA) task is the matching of each musical event of a given musical score to a point in a given recording of it, thus aligning, or synchronizing the audio with the score (Figure 3). It can be considered as a linking between the symbolic level and the performance level, with regard to the alterations between those two, as they were defined and described in 1.2.2.

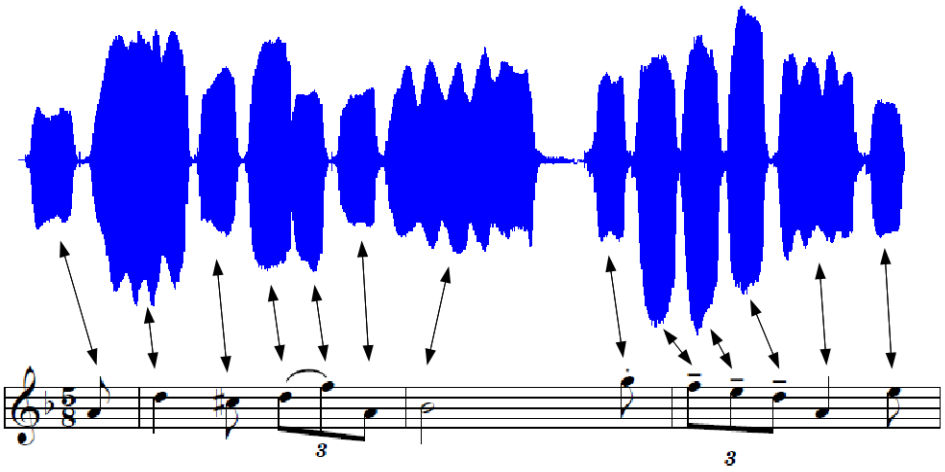

Figure 3: Assignment of each note of the score to a specific part of the performance waveform

In simpler words, it can give the computer the ability to "listen" to a performance and track the performer's position in the score, exactly as a human being could do. The real-time version of ASA, where the synchronization is taking place live, along the musical piece being performed, is called **Score Following**.

The biggest challenge of the creators of such a system is the invention of techniques

and approaches of coping with the different kinds of errors, omissions or interpretation variations of the performer.

2.2. Applications

There is a wide number of different activities, ASA can apply to. Each of them, concerns different fields, related to music and sound, and may be useful to a professional specialist or a casual user.

First of all, ASA can be considered as a beginning stage of an **integrated MIR system**. In this case, it can provide indexes and note labels for the sections of a large continuous audio recording, making the content-based retrieval for each section possible. It can also provide a distance measure between couples of notes in the sequence, thus contributing in a matching task.

An alignment between the score and the performance of a musical piece is also very useful from a **musicological** point of view. Researchers of this field, are able to study performances at time positions of their interest easily, as well as compare the unique characteristics of different performances of a piece. Music education has a lot to gain from ASA too. For example a musical instrument student can track his position in the score, while the system can offer him error locating, page turning etc.

Sound analysis is another field, to which ASA has a lot to contribute. Score following, the on-line version of audio to score synchronization, can be used in musical pieces, composed for both a computer and a human performer, giving the performer the possibility to differentiate his playing in some ways, and have the computer follow him. In this occasion, the computer system works as a virtual accompaniment performer, simulating a human musician. This can also apply in music education, since an accompanying performer for a musical instrument student will be always available. Another application related to sound analysis, is the contribution in a fundamental frequency estimation, by offering a limited number of possible note values, which has resulted from the alignment to the score.

The sectioning of a performance in separate notes, can also help in the source separation task, although this works in reverse too.

Finally, segmentation and indexing of audio performances, through the means of aligning it to its score, has many applications in **music creation** and composing. The re-transcription of a score, in accordance to a specific performance of it, is one of them. Furthermore, the musical units that can be derived from a segmentation of a performance, can be used to re-synthesize a musical piece in terms of concatenative synthesis. In similar manners the musical units can be used to create a virtual musical instrument.

Further discussion on ASA applications can be found in [15] and [16].

2.3. History / Approaches

The problem of audio to score alignment, has a long history of research of almost 30 years. Throughout all these years, although perfection has not been achieved, many different solutions have been proposed, most of which with satisfactory results. The real-time factor has been a major concern, since the beginnings of the related researches. What follows is an overview, as well as a time-line, of the milestone projects, divided into categories, according to their technical approach. For a full review of ASA history, one can address to [17] and [18].

2.3.1. String matching and pitch detection

The first attempts took place in the early years of the decade of the 1980's. Researcher Barry Vercoe was one of the first to define the problem and try to approach it. The computer model he proposed in 1984, was called “Synthetic Composer” [19] and was supposed to be able to replace a human performer in a group of musicians, without the others understanding the difference. His approach was taking into consideration the positions of the fingers of a performer on a flute, in order to understand the pitch being

played and be able to follow his performance. This system was also the first to include a “learning to improve” method, based on rehearsals. According to Vercoe, the three main stages of the system were: Listen, Perform, Learn [20].

Roger Dannenberg also made use of the performer pitch to propose a solution to the real-time accompaniment problem [21]. His algorithm considered the musical events as string sequences and used dynamic programming techniques for finding the longest common sub-sequence. This system was divided in these three tasks: recognizing the playing of the performer, matching it with a score, and providing an accompaniment. The performance mistakes it took into consideration are omitted and extra notes.

The following years, Dannenberg and his students made several extensions of the string matching algorithms first used [22], [23]. They introduced new matching algorithms, with the use of multiple matchers at the same time that attempted to deal with chords, trills, grace notes and glissandi. They also added a temporal delay before any matching decision is made, to prevent false matchings.

In the early 1990's, the development of Score Following at IRCAM was introduced, starting with EXPLODE, a system proposed by Miller Puckette, firstly in 1990 [24] and further described in 1992, together with Cort Lippe [25]. This system was designed, taking into consideration specific musical pieces, specially composed for being performed with Score Following. Such pieces include Philippe Manoury's “Pluton” and Pierre Boulez' “...Explosante-fixe...”. The algorithm of EXPLODE was based on pitch detection, taking advantage of MIDI symbols as performance input and providing pointers to the current and not-matched notes. It was tempo-independent and made no predictions for the forthcoming notes.

While EXPLODE had relatively good results when used with the compositions mentioned above, a new piece called “An echo” by Phillippe Manoury, placed new challenges for Miller Puckette. The piece was composed for a soprano and a computer and gave rise to a new system proposal in 1995 [26]. While it was still based on pitch detection, it had to use the real performance audio data to do that, so it was the first time a real audio to score alignment system was introduced. The audio analysis was done, dividing the signal

into frames, and trying to estimate the fundamental frequency of the singing voice, through the accelerated constant Q transform, surpassing, in this manner the difficulties such as the vibrato of the voice. The system also gave a strong emphasis on the elimination of the delay between the musician's performance and the response of the computer.

At about the same time, another system was proposed by Baird, Blevins and Zahler [27]. This system was based on Dannenberg's and Vercoe's publications. The novelty of it was that, instead of focusing on single music events, such as notes, it performed the matching, on segments of predefined length.

2.3.2. Dynamic Time Warping

One of the matching approaches, introduced for audio to score synchronization in the early 2000's was Dynamic Time Warping (DTW). It. is a popular and highly efficient algorithm for finding the optimal alignment between two time-series, taking into consideration the possible time shifting, deformation and variations in speed. The sequences are warped in time and determine a similarity measure. It has been used as a sequence alignment method in various data mining and information retrieval applications, such as in video and graphics, while it has played a major role in audio applications, especially those concerning speech, such as speech recognition.

From a more technical aspect, DTW computes the best path by:

$$\begin{aligned} p(m,n) &= min\{p(m-1,n-1)+2\mathrm{d}(m,n)\} \\ p(m,n) &= min\{p(m-1,n)+d(m,n)\} \\ p(m,n) &= min\{p(m-1,n)+d(m,n)\} \end{aligned} \tag{2}$$

where, $p(m,n)$ is the cost for a path up to (m,n), d is the local distance and initial cost $p(1,1)=d(1,1)$ [28].

DTW was first applied in audio to score alignment by IRCAM researchers, and specifically Nicola Orio [28]. The system he proposed, introduced a new low-level audio

feature called Peak Structure Distance (PSD), which was used as the local distance in the DTW algorithm. This feature was used, in place of the pitch measurements of the previous alignment systems. Orio considered that pitch is an error-prone feature and is not suitable to polyphonic pieces. PSD instead, analyzes the structure of the peaks of the expected harmonics of the notes, providing a reliable and versatile way to recognize single notes as well as chords. The audio feature of Peak Structure Distance and Peak Structure Match will be further described in 4.2.2.

This system, had the advantages of having a good memory resources management as well as achieving good results on pieces containing complex musical elements such as polyphony, multiple instruments, trials, vibrato and fast sequences. However, its requirement of having access to the knowledge of both the whole sequences before it was able to do the alignment computations, gave it the important drawback of working only off-line.

An attempt to overcome this obstacle, was done by Simon Dixon in 2005 [29], who proposed an on-line version of the DTW algorithm. This was achieved by using linear time and space costs, computing only the minimal path to the current position and performing calculations only to the latest fixed-length window of the performance.

2.3.3. Stochastic Approaches

During the second half of the 1990's, a new probabilistic approach emerged in the audio to score alignment researches. This approach attempted to cope with the amount of uncertainty, provoked by the variation and the mistakes that take place during a performance.

The first to provide a stochastic proposal for a solution to the audio to score alignment problem was Roger Dannenberg, this time with Lorin Grubb, in 1997 [30], [31]. In this system, the position in the score, is represented as a PDF, with the probabilities referring to the variation of the audio features having been formally set or empirically derived. The

position of the performer is represented as a continuous density function over the position in the score.

Bryan Pardo and William Birmingham, in 2001 [32] and 2002 [33], provided some publications on a simpler stochastic approach, extending some of the older work of Dannenberg and Puckette. They defined a match score that was being trained off-line from a data base of sounds, as well as a skip penalty. However, in this case, they were modeled according to a probability distribution.

A major milestone in the development of audio to score alignment, was the introduction of Hidden Markov Models. This was done by Chrostopher Raphael through a series of publications, starting with that of 1999 [34]. Much of the later work done by various researchers on this field was based on this publication, including the work on the IRCAM Score Follower, on which this thesis is based. Another proposal based on HMM's was done by Pedro Cano, Alex Loscos and Jordi Bonada in 1999 [35]. Anna Jordanous and Alan Smaill also implemented a system based on HMM's in 2008 [36], although it used MIDI data as the performance input, instead of analyzing audio data. The technical specificities of how HMM's and their algorithms can apply to the problem of audio to score synchronization will be fully described in 3.2 and 4.2.

Other developments on the probabilistic approaches domain is a combination of HMM's with Bayesian belief networks, proposed by Raphael in 2001 [37], providing a more reliable way of handling time issues, based on training from a specific performance.

2.3.4. Anticipatory approaches

The latest advances in the audio to score alignment researches include the use of anticipatory modeling. Artificially intelligent agents are used to make predictions about the future behavior of the real-time music data. Specifically, Arshia Cont, starting in 2008 [38], has further developed the IRCAM Score Follower in this direction.

The focus is on better temporal modeling and automatic tempo decoding. For this to be achieved, an occupancy distribution for the system states is defined. Thus, the use of

Hidden semi-Markov Models is imposed. The states of this type of models are dependent, not only on the previous states, but also on the occupancy distributions. However, the usual algorithms applied in HMM's, such as Forward-Backward and Viterbi (described in 3.2.3) have to be adapted to fit to this new type of models and the complexity of the new algorithms has important effects on the real-time performance of the system. The solution to this problem is the creation of hybrid Markov/semi-Markov models, since the actual need for a semi-Markov model is relatively rare.

This approach is believed by its creators to be a better extension of the previous systems proposed, because instead of borrowing techniques that were applied in speech processing, it tries to adapt to the real needs and specificities of music. The main intended application of the system is the use for live music performances, and in this domain, it has been used many times and with satisfactory results.

Further information on anticipatory approaches of ASA, can be found in [39] and [40].

2.3.5. Other approaches

Finally, there are some approaches to the audio to score alignment task, that cannot be included in any of the previous categories.

One of them was conducted by Jason Vantomme in 1990. This system was using temporal patterns as the main cue for recognition. Although the creator reports that the system is robust and able to cope with performance errors, there is a number of limitations this technical approach is bound to. The most important of them is that the focus on the performer's rhythmic patterns limits the diversity of the musical pieces the system can work with, into those with no complex rhythm patterns.

Another proposed system, is ComParser, developed by Schreck Ensemble and Pieter Suurmond in 2001 [41]. Although strongly related to the audio to score alignment issue, it is described by its authors as a pseudo-scorefollowing system, while their intention was to create a system for use in the ensemble's performances. Technically, it makes no use of

symbolic score information. Instead, the user manually sets some cue information on audio data, which is used as an input. The neural network technology which is applied, requires supervised training.

2.4. Available software

Although not explicitly stated, there are a number of commercial applications implementing some ASA algorithm so as to achieve their intended functionality. The following section presents some application examples having ASA at the core of their functionality.

2.4.1. Tonara

“Music that listens to you” is the main marketing motto of the Tonara software company. It was launched in September of 2011 as an application for the iPad mobile computer. It falls into the category of intelligent sheet music readers and its functionality can be basically described as a digital sheet music book, similar to an electronic book, with the special feature of being able to follow your playing and behave interactively. A cursor is showing your progress on the score and when you reach at the end of the page, that turns automatically. While there is other similar software that do that, usually that is achieved, relying on a fixed pre-defined tempo, and not by actually “listening” to your playing, coping with your mistakes etc. It is supposed to work well with polyphonic music, even with more than one instrument, while it is able to ignore environmental sounds. It also shows the tempo changes you do, while you are playing. Additional features, include recording and sharing of your music playing as well as making notes on the score on different layers. Tonara, together with a basic set of music scores are free, while you can purchase a music score you are interested in, in the appropriate format from the Tonara library.

The software has gained a good coverage by mainstream media. It has been used in a

number of concerts, while it was selected as “App of the week” by Apple in October 2011 in four countries.
[42]

2.4.2. Miso Music: Plectrum

Another similar application, compatible with the iPhone, iPod and iPad is Plectrum, developed by Miso Media Inc. The difference in this application is that it works with tablatures, instead of sheet music. It is intended to work as an educational software, offering error locating and giving instant feedback to the instrument player. While considered a guitar-learning application, the company claims that their software also support other instruments, including ukulele, bass, banjo, tenor banjo and mandolin.

The credits of the software include the TechCrunch Disrupt SF People’s Choice Award for 2010, as well as a nomination for App of the Year at the Siemer Silicon Beach Summit in 2011.
[43]

2.4.3. antescofo~

The long research on Score Following by the people at IRCAM have resulted in the modular system called “antescofo~”. It is distributed through the IRCAM forum since November 2009. The research & development team includes: Arshia Cont, Jose Echeveste, Jean-Lous Giavitto, Florent Jacquemard and Thomas Coffy, while its development was achieved in cooperation with composer Marco Stroppa. It is published as an external module for the Max/MSP ad PureData graphical audio programming environments.

It makes use of the latest technological advances described above, such as anticipation (from which its name derives) and includes of a synchronous programming language for

musical composition. It is able to automatically recognize the score position as well as the tempo, while it is supposed to work as a plug and play module, meaning that it requires no off-line training.

While the antescofo~ creators' intentions were to create a tool, useful for the live performance of electro-acoustic musical pieces, synchronizing a computer with a musical instrument performer, they do not end to that. IRCAM mentions that the further evolution of the software should be expected, focusing on interaction in computer music, for both composition and performance.

There are tens of musical pieces that are written or adapted for being performed with the use of antescofo~ and presentations of the software through concerts are often. The work of IRCAM has also been awarded by the French Ministry of Industry.

Antescofo~ has finally been a part of MuSET IMuSE, a system that, apart from audio, it takes advantage of live performance gestures data from standard controllers, move trackers, video trackers, etc.
[40]

2.4.4. PROBADO

PROBADO is a framework, that offers digital library services for non-textual data. Its focus is mainly on 3D graphics and music, while it allows browsing and accessing of the various documents.

As far as music is concerned, the system offers the ability to browse and playback the documents in an audio viewer or a score viewer. The user can switch between these views at any time, as well as move through the score, by choosing specific parts of it and listen to the corresponding parts of the performance immediately.
[44]

2.4.5. SampleSumo Music Following

SampleSumo is a “music technology and interactive multimedia” company. Their aim is to develop products and technological packages to cover the specific need of end-users as well as companies, taking advantage of some of the latest advances in MIR. The technologies they offer, other than melody transcription, percussive sound recognition etc., also include music following.

According to the creators' description of their technology, their software offers all the basic features of a real-time audio to score synchronization system, such as following the position of a performer in the score with robustness, filtering environmental noise, while it is supposed to work with multiple musical instruments.

Music notation software MuseScore and NeoScores have integrated audio to score alignment features, based upon SampleSumo. The system has also been showcased with live performances in various concerts, conventions and television shows.
[45]

3. Pattern Recognition & Machine Learning

3.1. Introduction

Pattern recognition is an important field of computer science and artificial intelligence, whose goal is the automatic identification of observed objects.

Pattern recognition has made its first successful achievements long ago, such as the discovery of the regulations of the planetary motions, by Johannes Kepler in the 17th century, or the work related to the atomic spectra, at the beginnings of the 20th century. The related theoretical research in the field of statistics has a long history too. However, it was only after the 1960's and the explosion of the computer use, that the number of applications of pattern recognition, as well as the demand for additional theoretical development have been raised enormously. [46]

A general recognition system, usually, receives some input data, which is then given a label out of a given set of classes, according to a scheme, applied on **feature** values that have been extracted from the data. The objects to be identified, could be anything, such as an audio recording of a voice, images of fingerprints, or any other collection of measurements and there is a wide variety of pattern recognition example tasks, including machine vision, optical character recognition, speech recognition, musical genre classification and many more.

Figure 4: Pattern recognition basic stages

The stages of the development of a pattern recognition system could be pointed out as

the following:

1. The feature generation and selection stage, at which a set of meaningful descriptors and their values are extracted from the input data and the most useful out of them are chosen to be taken into consideration for the classification task.
2. The classifier design stage, at which the relationships between the features and the system estimations of the objects identities are set out.
3. The system evaluation stage, at which the classification errors are enumerated and the system performance is evaluated.

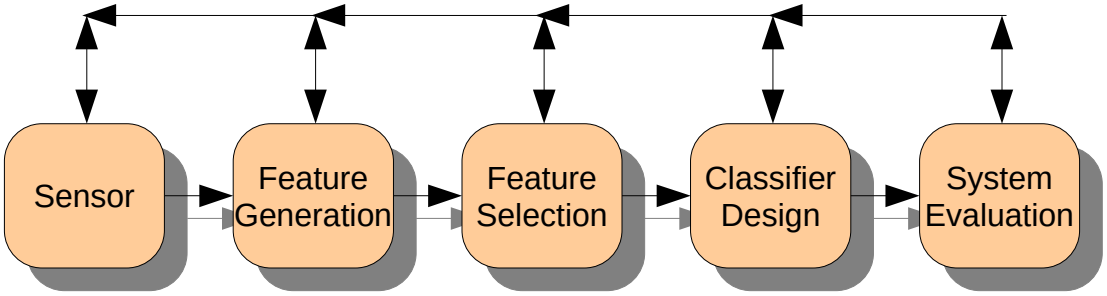

Figure 5: The development stages of a recognition system

The key point, in the development of pattern recognition systems is, instead of providing the system with all the rules needed to be able to recognize each object, giving it a set of real examples of objects and making it able to "learn" the rules from them. The branch of artificial intelligence, related with these tasks is called **Machine Learning** and a definition of it, is the following [47]:

> *A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E*

Here, we will try to describe the task of learning in terms of some functions. Consider

f is the function our system is trying to learn. The function we use to approximate f, will be h, so that, when it accepts vector-valued input $X=(x_1,x_2,\dots,x_N)$, it returns vector-valued output $h(X)$. We select function h, from a class of functions H, of which f could be also a member, and we do this choice, based on a training set $\Xi=(x_1,x_2,\dots,x_M)$.

More introductory information on pattern recognition and machine learning can be accessed in [47], [46], [48] and [49].

3.1.1. Supervised and Unsupervised Learning

The types of learning are divided in two main categories. At first, we will examine the setting we use to learn a function, called **supervised** learning. In this type of learning, the output values of f for the input values of the training set Ξ, are known. This allows us to assume that if we find a function h that, given the same input Ξ, returns output values close to those of f, then it is a good guess of f. Also, the greater the size of our training set, the better the guess of the function will be.

In more practical terms, having supervised learning, means that we give our program a collection of data that have been already labeled with their identity, and we expect the program to find out the rules of identification, so that it can use it on other, unlabeled, data.

There are many ways of finding out such a function. Fitting a polynomial curve onto

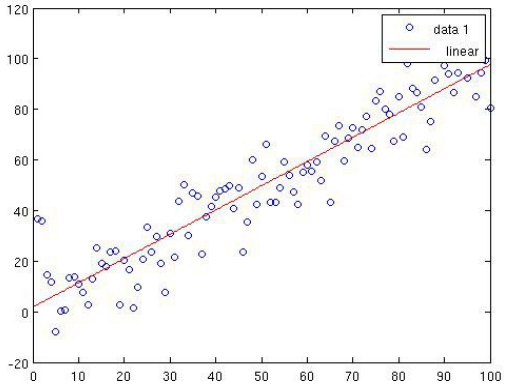

Figure 7: Linear curve fitting

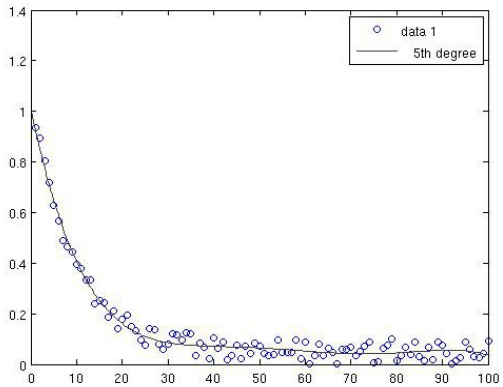

Figure 6: Exponential curve fitting

the given points, is the simplest example (Figure 7, Figure 6).

The other type is the one called **unsupervised** learning. In this type, the training set Ξ is provided without the function values of f. The problem in this case, is to find an appropriate way to divide the training set Ξ into some sub-sets $\Xi_1, \Xi_2, \ldots, \Xi_R$, and the function we search for it, the one that returns the sub-set that the input vector belongs to.

Again, in more practical terms, having unsupervised learning, means that we give our program a collection of data that are not characterized in any way, and we expect the program to divide them in groups of similar, in some way, content.

Supervised and unsupervised learning are fully described in [47], [46], [48] and [49].

3.1.2. Probability Theory

The most important branch of mathematics that pattern recognition makes use of and one of its central foundations, is the one concerned with uncertainty, random phenomena and probability of events.

The marginal probability of an event X is denoted as $p(X)$. The joint probability of X and Y is denoted as $p(X,Y)$ and the conditional probability of Y given X is denoted as $p(Y|X)$.

Given that, we can deliver the two fundamental rules of probability theory.

The **sum rule**:

$$p(X)=\sum_{Y} p(X,Y) \tag{3}$$

And the **product rule**:

$$p(X,Y)=p(Y|X)\,p(X) \tag{4}$$

The **Bayes theorem** derives from the product rule and the symmetry property $p(X,Y)=p(Y,X)$ and plays a central role in pattern recognition and machine learning:

$$p(X|Y)=\frac{p(Y|X)p(X)}{p(Y)} \tag{5}$$

The Bayesian interpretation of the theorem links the probability of an event X before and after the observation of event Y. $p(X)$ is called the prior and it is the initial probability before the observation, $p(X|Y)$ is called the posterior and it is the probability after the observation of Y, the conditional probability $p(Y|X)$ is called likelihood and $p(Y)$ is the normalization factor.

Furthermore, two probabilities $p(X)$ and $p(Y)$ can be characterized as **independent**, when the probability of event X does not affect the probability of event Y, and

$$p(X,Y)=p(X)p(Y) \tag{6}$$

The conditional independence of the probabilities of two events X and Y, given a third event Z, exists when given the knowledge of the occurrence of Z, X and Y are independent, while if given that Z does not occur, X and Y are not independent:

$$p(X,Y|Z)=p(X|Z)p(Y|Z) \tag{7}$$

All the basic probability theory is covered in [47], [46], [48] and [49].

3.1.3. Statistical models

The observable outputs that are produced by real-world processes can be characterized

as signals of values over time. These real-world signals can be described by signal models, thus making the realization of recognition systems for those processes able.

There are two types of signal models. The deterministic models which are specified by taking advantage of the known properties of the signal, and the statistical models, in which we consider the real-world process a stochastic process, thus it can be described only by statistical properties.

The idea behind a statistical model is uniting the elegance and generality of mathematics with the statistical developments emerging from data. Building a statistical model of a phenomenon, involves collecting all available data from the real-world phenomena, and drawing all possible conclusions, interpreting the data in a senseful way. If the statistical model is a precise representation of the real-world process, it will enable us to make useful predictions about how the process will behave in the real-world, under certain circumstances we are interested in. Such a model is called a good fit, while the opposite is called a poor fit.

The building blocks of which a statistical model is made, are the probability distributions. These distributions represent both the random and the systematic variance, and the success of modeling lies in finding the balance that makes the answering to the problems occurred, possible. The same datasets can be used for building different models, depending on the problems and questions of interest.

In mathematical terms, a statistical model is a collection of probability distribution functions. If each distribution is indexed by a unique finite-dimensional parameter, the model is called parametric. All parameters are collected together and form a k -dimensional parameter vector $\theta=(\theta_1,\ldots,\theta_k)$. If the model has infinite dimensional parameters, it is called non-parametric. Finally, if the model is made of both parametric and non-parametric components, it is called semi-parametric.

Statistical models are further covered in [50] and [51].

3.1.4. Probability densities, the Gaussian distribution and Gaussian Mixture Models

The concept of probabilities for discrete events, examined in 3.1.2, can be extended, taking into account probabilities of continuous variables. A **Probability Density Function** (PDF) of a variable is the function that gives us the probability of the variable to take on a given value. Its integral over a region gives us the probability of the variable to fall into this region.

The probability density can be expressed as the derivative of a **Cumulative Distribution Function** (CDF), which is the probability $P(z)$ of the variable to be found at a value less than or equal to z, that is to say to lie in the interval $[-\infty, z]$:

$$P(z)=\int_{-\infty}^{z} p(x)dx \tag{8}$$

The **Gaussian**, or Normal distribution is one of the most widely used models for representing the distributions of the continuous variables. It is defined on the entire real-line and its Probability Density Function is given by the Gaussian function:

$$f(x|\mu,\sigma^2)=\frac{1}{\sigma\sqrt{2\pi}}e^{-(\frac{1}{2})(\frac{x-\mu}{\sigma})^2} \tag{9}$$

μ is the mean, σ^2 is the variance, σ is the standard deviation and $\frac{1}{\sigma^2}$ is the precision.

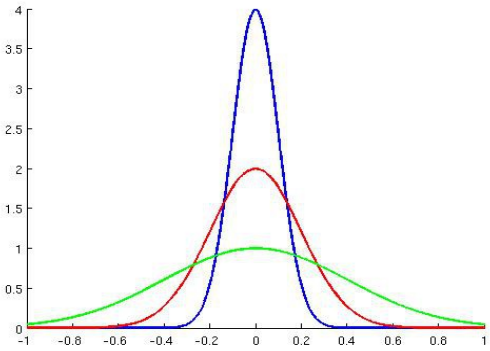

Figure 9: Gaussian PDF's of different variances

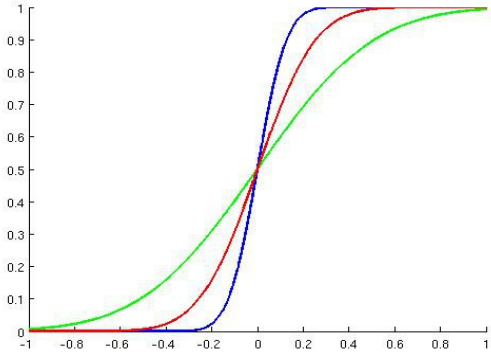

Figure 8: Gaussian CDF's of different variances

However, simple Gaussian functions are not always able to sufficiently model complex data distributions. This limitation is often surpassed by using a Mixture Model, which is the linear combination of a number of sub-populations. When each of these populations correspond to the Gaussian distribution, then the mixture is called a **Gaussian Mixture Model** (GMM) and is given by the following equation:

$$p(x)=\sum_{k=1}^{K} \pi_k f(x|\mu_k, \sigma_k^2) \tag{10}$$

Where k is the component index, K is the number of Gaussian components and π_k is called the mixing coefficient, for which:

$$\sum_{k=1}^{K} \pi_k = 1 \tag{11}$$

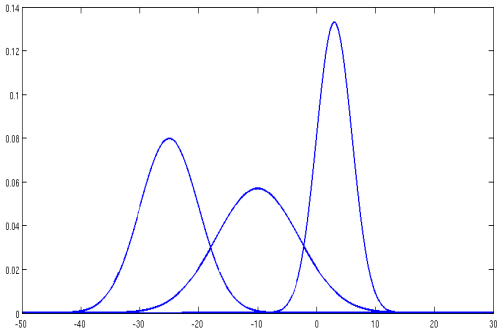

Figure 11: Three gaussian distributions with different means and variances

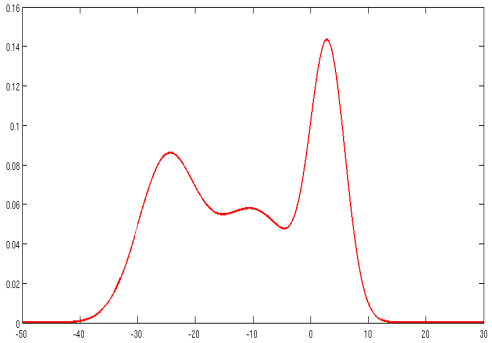

Figure 10: A GMM consisting of three components

In Figure 11 and Figure 10, we can see an example of how a GMM is constructed by superposing three different Gaussian functions.

Further information on probability densities, the Gaussian distribution and GMM's can be found in [47], [46], [48] and [49].

3.1.5. The Expectation-Maximization algorithm for GMM's

The **Expectation-Maximization** (EM) algorithm is an iterative process which can estimate the parameters of a statistical model, such as the means and variances of each component of a GMM. It has many applications in machine learning.

The Baum-Welch algorithm, often used for training Hidden Markov Models, is also using the EM algorithm and will be covered in 3.2.3.

The steps of the process, when applied to GMM's, are the following:

1. **Initialization** of the parameters (μ , σ^2 , π) and log likelihood.
2. **E step**: Evaluation of the probabilities

$$\gamma(z_{nk}) = \frac{\pi_k f(x_n; \mu_k, \sigma_k^2)}{\sum_{j=1}^{K} \pi_j f(x_n; \mu_j, \sigma_j^2)} \tag{12}$$

3. **M step**: Re-estimation of the parameters using the current probabilities.

$$\mu_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) x_n \tag{13}$$

$$\sigma_k^{2\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(x_n - \mu_k^{new})(x_n - \mu_\kappa^{new})^T \tag{14}$$

$$\pi_k^{new} = \frac{N_k}{N}, N_k = \sum_{n=1}^{N} \gamma(z_{nk}) \tag{15}$$

4. **Evaluation** of log likelihood

$$\ln(p(X; \mu, \sigma^{2,} \pi)) = \sum_{n=1}^{N} \ln(\sum_{k=1}^{K} \pi_k f(x_n; \mu_\kappa, \sigma_k^2)) \tag{16}$$

Return to step 2 if the convergence criterion is not satisfied for either the parameters or log likeliood.

The EM algorithm was introduced and fully covered in 1977 in [52], while other sources for further information on it are [46] and [53].

3.1.6. Generative & Discriminative Models

Statistical models can fall into two main categories: those of the generative and discriminative models, both of which are useful in solving the classification problem by

learning a rule to map input X (data) to output Y (labels). A **generative** model is able to generate random data, obeying certain rules defined by the model. A **discriminative** model is used only for the tasks of classification and regression of data, while not being able to generate sample datasets. The main difference between these two types of models is that, in the case of the generative models, the joint probability distribution $P(X,Y)$ is specified, while in the case of the discriminative models, the conditional probability $P(Y|X)$ is the one modeled. This makes easy to understand why having a generative model gives us the possibility of generating likely pairs of (X,Y) but also using the Bayes theorem to transform $P(X,Y)$ to $P(Y|X)$, while discriminative models can only be used for predicting Y given X.

However, each type of modeling has its own advantages and disadvantages and it is the specific application requirements that define the choice of the model. Assuming the problem in discussion is classification, discriminative models are considered to perform better from a computational point of view since

> *one should solve the classification problem directly and never solve a more general problem as an intermediate step.*
>
> [54]

On the other hand, a generative model is more flexible and capable of handling missing data as well as it can express more complicated relationships of variables.

Examples of generative models include Gaussians, Mixtures of Gaussians (3.1.4), Mixtures of multinomial, Mixtures of experts, Sigmoid Belief Networks, Bayesian Networks (3.1.8), Markov Random Fields and Hidden Markov Models (3.2), while examples of discriminative models include Logistic Regression, Gaussian Processes, Regularization networks, Support Vector Machines and Neural Networks.

Further details on generative and discriminative models can be found in and [46] and [55].

3.1.7. Probabilistic Graphical Models

Graphical Models or **Probabilistic Graphical Models** (PGM's) constitute a framework for manipulating large-scale numbers of variables connected to each other with complicated relationships, making use of the ideas of logical structures from computer sciences and ending with a powerful and flexible way of representing real world phenomena as large networks, available for performing learning and inference. For this to occur, we consider a complex system as a combination of simpler parts. This idea of modularity, is the fundamental notion behind the framework. PGM's are based on the combination of probability theory with graph theory, thus, they provide tools for dealing with uncertainty and complexity. Probability theory makes sure that each module is connected in a stable and meaningful way with the others, while graph theory provides us with an appealing and motivating representation of the network.

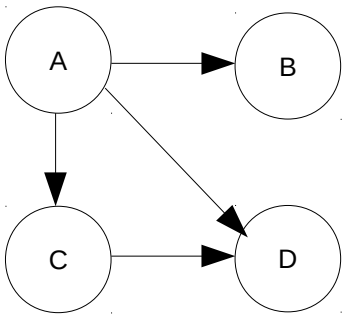

Figure 12: Directed graphical model

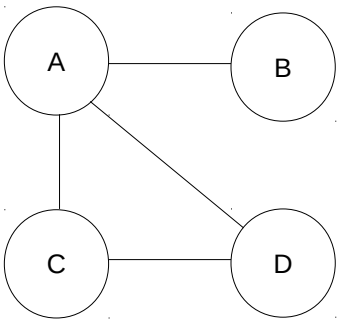

Figure 13: Undirected graphical model

In more practical terms, Graphical Models are graphs with nodes, connected with edges. The nodes represent the various random variables, or a group of random variables, while the edges show the conditional independence between them. Graphical models are divided into **directed** (Figure 12) and **undirected** (Figure 13) ones, according to the existence of directionality or not within the edges, indicated by arrows or not. Markov random fields are undirected models, while bayesian networks, the details of which will be examined later, are directed ones.

Finally, the concept of Graphical Models allows applying the same statistical models and computational techniques in different application domains, as different instances of a common underlying formalism. In this way, techniques and models developed in one research domain can be easily transferred to another, while permitting at the same time the design of new models using the same development frameworks.

Additional information on PGM's can be accessed in [56], [57], [58] and [59].

3.1.8. Bayesian Networks

A Directed Acyclic Graph (DAC) is the one, in which all nodes are connected to each other with directed edges in a way that starting from a specific node X, there is no possible sequence to be followed that can lead to node X again. The joint probability distribution of a network with a structure of the DAC category, can be factored into the product of all its conditional dependencies as follows:

$$P(X_{1,}\dots,X_n)=\prod_{i=1}^{n} P(X_i|Pa_i) \tag{17}$$

where $X_{1,}\dots,X_i$ are all the nodes and Pa_i is the parent node to X_i . The above equation is called the chain rule. In the example above, this can be expressed as:

$$P(A,B,C,D)=P(A)\,P(B|A)\,P(C|A)\,P(D|A,B,C) \tag{18}$$

Such a type of model is called a **Bayesian Network**, and it is the graphical representation of a specific factorization of a joint probability distribution.

Bayesian Networks are fully described in [60], [56] and [57].

3.1.9. Dynamic Bayesian Networks

The extension of bayesian networks, so that they can model time series data is called **Dynamic Bayesian Networks** (DBN). They model probability distributions over semi-infinite sets of variables $Z_t=\{Z_{1,}Z_{2,}\ldots\}$. Every time a new observation arrives, time index t increases by one.

More complete descriptions of DBN theory can be found in [61] and [62].

3.2. Hidden Markov Models

3.2.1. Definition

A **Markov process** is a type of statistical model, in which the system being modeled, satisfies the Markov property. The property is true when the future values of the system output signal depend only on the present state of the system and not to any of the previous states.

Hidden Markov Models (HMM's) are Markov processes, of which, the state sequence is hidden, and the observable output is a probabilistic function of the state. HMM's are considered the simplest form of DBN's (3.1.9).

Here are all the required definitions used to describe an HMM:

N is the number of all the distinct states the system can be in.

M is the number of all the distinct observable symbols the system can output.

$S=\{S_1,S_2,\ldots,S_N\}$ is the set of all the distinct states.

$V=\{V_1,V_2,\ldots,V_M\}$ is the set of all the distinct observation symbols.

$A=\{a_{ij}\}$ is the **transition probability matrix** for the system transmitting from state

S_i to state S_j

$B=\{b_j(k)\}$ is the **emission probability matrix** for the system being at state S_j emitting observation symbol V_k

$\pi=\{\pi_i\}$ is the initial probability of the system being at state S_i

$t=\{1,2,3,\ldots,T\}$ are the time instants we examine

$Q=\{q_t\}$ is the sequence of states of the system at each time instant

$O=\{O_t\}$ is the sequence of observable symbols at each time instant

The complete parameter set used to describe an HMM is $\lambda=(A,B,\pi)$.

Lawrence Rabiner's 1989 tutorial on HMM's [63] remains one of the most complete introductory resources today, while many additional useful information can be found in [64], [65] and [57].

3.2.2. Types of HMM's

HMM's cam be divided in many different categories, depending on the transition matrix, that is to say the permitted connections between the states.

The standard type of an HMM is the **ergodic** or fully connected model. In this type, any state can be reached directly from any other state (Table 1, Figure 14). Using the definitions above, this means that $\{a_{ij}\}$ is positive for any i and j .

	S_1	S_2	S_3	S_4
S_1	a_{11}	a_{12}	a_{13}	a_{14}
S_2	a_{21}	a_{22}	a_{23}	a_{24}
S_3	a_{31}	a_{32}	a_{33}	a_{34}
S_4	a_{41}	a_{42}	a_{43}	a_{44}

Table 1: Transition probability matrix of an ergodic HMM,

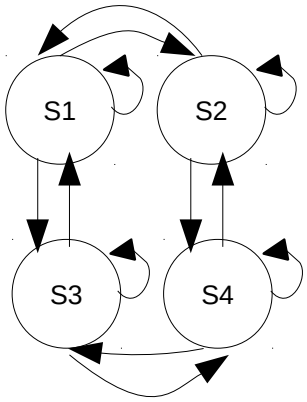

Figure 14: Graphical model of an ergodic HMM

However, judging from the observable outputs of the signals, the ergodic type is not the ideal choice for all applications. So the following variations are also often used.

In the **left-right**, or Bakis model, the state index increases as time instant increases (Table 2, Figure 15). This means that no transitions to states with indexes lower than the current index are allowed, as well as that the state sequence begins with state S_1 and ends with state S_N .

$$a_{ii}=0, j<i \qquad \pi_i=1, i=1 \qquad \pi_i=0, i\neq 1$$

This makes the states move from left to right. It is easy to understand that such a model works ideally for applications with signals with properties that change over time.

	S_1	S_2	S_3	S_4
S_1	a_{11}	a_{12}	0	0
S_2	0	a_{22}	a_{23}	0
S_3	0	0	a_{33}	a_{34}
S_4	0	0	0	a_{44}

Table 2: Transition probability matrix of a left to right HMM

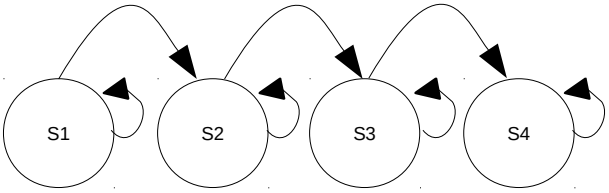

Figure 15: Graphical Model of a left to right HMM

There are several variations of the left-right model, based on different constraints placed on the transition matrix. One of them includes a **jump transition** (Table 3, Figure 16).

$$a_{ij}=0, j>i+\Delta$$

where Δ is the number of states, more of which, a jump is not allowed.

	S_1	S_2	S_3	S_4
S_1	a_{11}	a_{12}	a_{13}	0
S_2	0	a_{22}	a_{23}	a_{24}
S_3	0	0	a_{33}	a_{34}
S_4	0	0	0	a_{44}

Table 3: Transition probability matrix of a left to right HMM with a jump transition

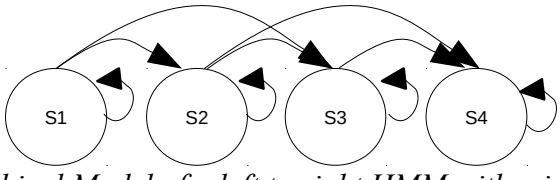

Figure 16: Graphical Model of a left to right HMM with a jump transition

Another variations is the addition of **parallel paths** to a left-right model such as that

of :

	S_1	S_2	S_3	S_4	S_5	S_6	S_7
S_1	a_{11}	a_{12}	0	0	a_{15}	0	0
S_2	0	a_{22}	a_{23}	0	0	a_{26}	0
S_3	0	0	a_{33}	a_{34}	0	0	a_{3}
S_4	0	0	0	a_{44}	0	0	0
S_5	0	a_{52}	0	0	a_{55}	0	0
S_6	0	0	a_{63}	0	0	a_{66}	0
S_7	0	0	0	a_{74}	0	0	a_{7}

Table 4: Transition probability matrix of an HMM with a parallel path

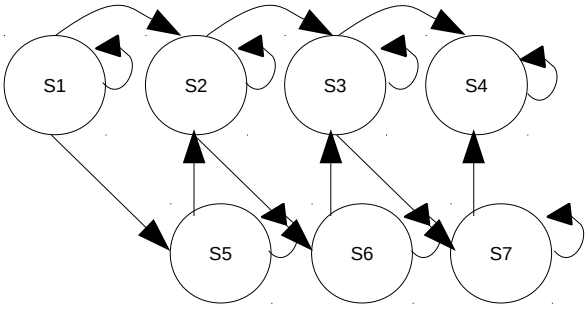

Figure 17: Graphical Model of an HMM with a parallel path

It should be also noted that, regardless of the type of HMM, the last state of the model can only lead to itself:

$$a_{NN}=1 \qquad a_{i}=0, i<N$$

3.2.3. The three problems of HMM's

Here is a real-world example of an HMM. Consider a model that describes the state of the weather at each day. In the simplest form, the weather can vary from sunny to rainy. Thus, the system possible states are $N=2$ and the states set is $S=\{'Sun','Rain'\}$. The state of the weather affects the choice of a person about how to spend the day. His choices could be taking a walk outside, or staying at home and study. His choice is the system observable output $V=\{'Walk','Study'\}$ and the number of output symbols is $M=2$.

This could be the probability matrix for transitions between weather states $A=\{a_{ij}\}$, as shown in Table 4 and Figure 18.

	Sun $j=1$	**Rain** $j=2$
Sun $i=1$	0.7	0.3
Rain $i=2$	0.8	0.2

Table 5: Transition matrix of the Sun/Rain HMM

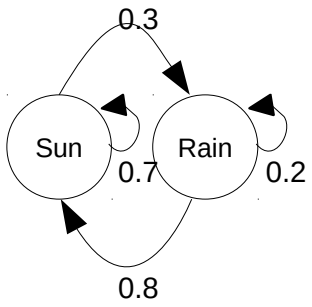

Figure 18: Graphical Model of the Sun/Rain HMM

Table 6 could be the probability matrix $B=\{b_j(k)\}$ for emitting choices of action, depending on the weather states.

	Walk $k=1$	**Study** $k=2$
Sun $j=1$	0.6	0.4
Rain $j=2$	0.1	0.9

Table 6: Emission matrix of the Sun/Rain HMM

Also, Table 7 could be the initial state probability $\pi=\{\pi_i\}$:

Sun $i=1$	0.8
Rain $i=2$	0.2

Table 7: Initial state probability of the Sun/Rain HMM

Summarizing, we have the following:

$$A=\begin{bmatrix}0.7 & 0.3\\ 0.8 & 0.2\end{bmatrix} \quad B=\begin{bmatrix}0.6 & 0.4\\ 0.1 & 0.9\end{bmatrix} \quad \pi=\begin{bmatrix}0.8\\ 0.2\end{bmatrix}$$

These are all the parameters required to describe our model. If we examine our system for $T=5$ days, we will observe an output sequence such as $O=\{'Walk','Study','Study','Walk','Study'\}$, generated by a hidden state sequence $Q=\{q_1,q_2,q_3,q_4,q_5\}$

Given this specific model, we may come with the following questions:

- What is the probability of the person having a specific sequence of choices such as O above?
- If the person's choices sequence is O , what is the most probable weather state sequence Q that caused that?
- What the transition and emission probability matrices should be, so that the probability of the specific sequence of choices O could be maximized?

These three questions correspond to the three basic problems for HMM's that will be

described and be given solutions below.

- **First Problem: Forward-Backward Algorithm**

$O=\{O_t\}$
$\lambda=(A,B,\pi)$
$P(O|\lambda)=?$

The solution to this problem will give us the probability of a specific observation, given the model. It will also make us able choose from many competing models, the one that describes best this specific observation.

If $Q=\{q_t\}$ is the state sequence, as defined above, then:

$$P(O|\lambda)=\sum_{q_1,q_2,\dots,q_T} P(O|Q,\lambda)P(Q,\lambda)=\sum_{q_1,q_2,\dots,q_T} b_{q_1}(O_1)b_{q_2}(O_2)\dots b_{q_T}(O_T)\cdot\pi_{q_1}a_{q_1q} \qquad (19)$$

In order to avoid the $2T\cdot N^T$ calculations needed (which in our simplest weather example of $N=2$ and $T=5$, corresponds to 320 calculations), we will make use of the Forward-Backward algorithm.

At first, we define the **forward variable**:

$$\alpha_t(i)=P(O_1O_2\dots O_t,q_t=S_i|\lambda) \qquad (20)$$

which can be described as the probability of having the partial observation $O_1,\dots O_t$ until time t and having state S_i at that time. The procedure consists of 3 steps:

Step 1: Initialization

$$\alpha_1(i)=\pi_i b_i(O_1),\ \ 1\leqslant i\leqslant N \tag{21}$$

Step 2: Induction

$$\alpha_{t+1}(j)=[\sum_{i=1}^{N}\alpha_t(i)a_{ij}]\,b_j(O_{t+1}),\ \ 1\leqslant t\leqslant T-1,\ \ 1\leqslant j\leqslant N \tag{22}$$

Step 3: Termination

$$P(O|\lambda)=\sum_{i=1}^{N}\alpha_T(i) \tag{23}$$

We can have exactly the same results, by defining a **backward variable**, in the same way:

$$\beta_t=P(O_{t+1}O_{t+2}...O_T|q_t=S_i,\lambda) \tag{24}$$

The procedure, in this case, is as follows:

Step 1: Initialization

$$\beta_T(i)=1,\ \ 1\leqslant i\leqslant N \tag{25}$$

Step 2: Induction

$$\beta_t(i)=\sum_{j=1}^{N}a_{ij}b_j(O_{t+1})\beta_{t+1}(j),\ \ t=T-1,\,T-2\,,\,...,1\ \ 1\leqslant i\leqslant N \tag{26}$$

Step 3: Termination

$$P(O|\lambda)=\sum_{j=1}^{N}\pi_i b_j(O_1)\beta_1(j) \tag{27}$$

Either using the forward or the backward procedure, for the solution to our problem, the number of calculations needed, reduce to $N^2\cdot T$ (which in the example mentioned above, is just 20 in contrast to 320 needed before).

- **Second Problem: Viterbi Algorithm**

$O=\{O_t\}$
$\lambda=(A,B,\pi)$
$Optimal Q=\{q_t\}=?$

The solution to this problem will give us the "optimal" state sequence Q that may have caused the observation O, given model λ. One way of interpreting this "optimality" is by finding individually each most probable state q_t. In order to do that, we will define another variable:

$$\gamma_t(i)=P(q_t=S_i|O,\lambda) \tag{28}$$

which is the probability of being at state S_i at time t, given the observation and the model. Making use of the forward and backward variables defined above, and after a normalization, this can also be expressed as:

$$\gamma_t(i)=\frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N}\alpha_t(i)\beta_t(i)} \tag{29}$$

Thus, the most probable state for each time will be

$$q_t=argmax[\gamma_t(i)] \tag{30}$$

The problem with this solution is that finding each most probable state qt and combining each one of them in a state sequence Q, may result in a sequence not valid by the transition matrix, which will contain successions of states with zero transition probabilities. Again, to avoid this problem we will make use of an available dynamic programming procedure, called the Viterbi algorithm.

What we need is to define a variable that describes the highest probability of having a state sequence that ends with state Si at time t, and a specific observation O, given the model. That is:

$$\delta_t(i)=max\, P[q_1q_2\ldots q_t=S_i,O_1O_2\ldots O_t|\lambda] \tag{31}$$

The Viterbi algorithm is similar to the forward-backward algorithm, but it differs in maximizing instead of summing, in the induction and termination steps. It also keeps track of the maximized argument in the matrix $\psi_t(i)$. The steps of the procedure are the following:

Step 1: Initialization

$$\delta_t(i)=\pi_t b_i(O_1),\ 1\leqslant i\leqslant N \tag{32}$$

$$\psi_1(i)=0 \tag{33}$$

Step 2: Induction

$$\delta_t(j)=max[\delta_{t-1}(i)a_{ij}]b_j(O_t),\ \ 2\leqslant t\leqslant T,\ \ 1\leqslant i,j\leqslant N \tag{34}$$

$$\psi_t(j)=argmax[\delta_{t-1}(i)a_{ij}],\ \ 2\leqslant t\leqslant T,\ \ 1\leqslant j\leqslant N \tag{35}$$

Step 3: Termination

$$\dot{p}=max[\delta_t(i)],\ \ 1\leqslant i\leqslant N \tag{36}$$

$$\dot{q_T}=argmax[\delta_T(i)],\ \ 1\leqslant i\leqslant N \tag{37}$$

Step 4: State sequence backtracking

$$\dot{q_t}=\psi_{t+1}(\dot{q_{t+1}}),\ \ t=T-1,T-2,\ ...,\ 1 \tag{38}$$

- **Third Problem: Baum-Welch Algorithm**

$O=\{O_t\}$

$Maximum\,P(O|\lambda)$

$\lambda=(A,B,\pi)=?$

The solution to this problem will give us a re-estimation of the model parameters, so that the probability of a specific observation sequence is maximized. This is the most difficult, out of the three HMM problems and there is no optimal way of solving it. The most widely accepted procedure for solving it is the Baum-Welch algorithm, which is a generalized EM algorithm (3.1.5) and is the most widely used HMM training algorithm.

First, we have to define again a variable:

$$\xi_t(i,j)=P(q_1=S_i,q_{t+1}=S_j|O,\lambda) \tag{39}$$

which describes the probability of, being at state S_i at time t, and state S_j at time $t+1$, given the observation and the model. It can also be rewritten in terms of the forward and backward variables, and normalized:

$$\xi_t(i,j)=\frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{P(O|\lambda)}=\frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)} \tag{40}$$

If this is summed over j, it gives us the probability of being at state S_i at time t, defined previously as $\gamma_t(i)$:

$$\gamma_t(i)=\sum_{j=1}^{N}\xi_t(i,j) \tag{41}$$

Also, we can get the expected number of transitions from S_i, by summing $\gamma_t(i)$ over t :

$$\sum_{t=1}^{T-1}\gamma_t(i) \tag{42}$$

and the expected number of transitions from S_i to S_j, by summing $\xi_t(i,j)$ over t :

$$\sum_{t=1}^{T-1}\xi_t(i,j) \tag{43}$$

These lead us to to the following method of re-estimating each one of the model parameters:

$$\bar{\pi}_i = \gamma_1(i) \tag{44}$$

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \tag{45}$$

$$\bar{b}_j(k) = \frac{\sum_{t=1}^{T} \gamma_t(j)^*}{\sum_{t=1}^{T} \gamma_t(j)}, \quad * \text{ so that } O_t = V_k \tag{46}$$

- **Real-time implementation: Online Viterbi Algorithm**

The Viterbi algorithm shown above is applied on the whole sequence and requires access to both the previous and the next time instants. In order to use Viterbi decoding in real-time applications, we must use a variation of it, proposed in [66], which will use only the values of the previous states that are available at the current time instant.

Step 1: Initialization

$$\delta_t(i) = \pi_t b_i(O_1), \quad 1 \leqslant i \leqslant N \tag{47}$$

$$\psi_1(i) = 0 \tag{48}$$

Step 2: Induction

$$\delta_t(j) = max[\delta_{t-1}(i) a_{ij}] b_j(O_t), \quad 2 \leqslant t \leqslant T, \quad 1 \leqslant i, j \leqslant N \tag{49}$$

$$\psi_t(j) = argmax[\delta_{t-1}(i) a_{ij}], \quad 2 \leqslant t \leqslant T, \quad 1 \leqslant j \leqslant N \tag{50}$$

Step 3: Termination

$$\dot{p} = max[\delta_t(i)], \quad 1 \leqslant i \leqslant N \tag{51}$$

Step 4: State sequence backtracking

$$\dot{q}_t = \psi_{t+1}(\dot{q_{t+1}}), \quad t = T-1, T-2, \ldots, 1 \tag{52}$$

4. System Implementation

4.1. Choice of methodology

This section outlines the reasons for choosing the stochastic HMM methodology for the system under implementation, as opposed to the alternative computational approaches that were presented in section 2.3

Firstly, the early approaches of string matching and pitch detection have long been regarded as outdated, because of their limited functionality and relatively poor results. The two main competitor approaches were Dynamic Time Warp and Hidden Markov Models. While DTW has been applied by several researchers with good results, it preserves the basic disadvantages of not being able to work in real-time and offers no possibility of a training of the system based on a performances dataset.

On the contrary, the use of Hidden Markov Models seems to have become the standard approach for coping with the ASA task for at least the last decade. Although it was firstly introduced on the second half of the 1990's, the latest advances in the ASA field, such as the anticipatory approaches, are still based on those systems and the use of HMM's. Thus, building an ASA system based upon HMM's can be of good functionality, while it can be the basis for a further developed system in the future.

In addition, there is a wide bibliography available, not only on the basic theory of HMM's but also on the specific application of them in the ASA domain. This makes the first contact with the implementation of an ASA system much more convenient.

Finally, HMM's is a technology widely used on the contemporary and ongoing Information Retrieval and specifically MIR research fields. Thus, the learning of its theory and the familiarization with its use offered us with a good understanding of how similar approaches are used in other fields of interest.

The choice of the implementation details, such as the selection and number of audio features, their interaction, the training process etc. was made after carefully studying the

available bibliography and after several tests and experimentation which will be fully described in 5.2.

The work on which our system was mainly based was the publications concerning the IRCAM score follower and more specifically the publication of Ashia Cont, “Improvement of Observation Modeling for Score Following” [18]. The studying of the source code, written by Antoine Gomas for his master thesis “Audio to Score Alignment for Educational Software” [16], which was also based on IRCAM's work, was also helpful for understanding some basic implementation issues.

4.2. HMM Architecture

4.2.1. Transition matrix

The transition matrix of the HMM is a direct translation of the score. For each note of the score, three states (**Attack**, **Sustain** and **Rest**) are created and transition probabilities between them and between these states and those of the other notes are set. Thus, it is a square matrix with both dimensions equal to the number of note events in the score, multiplied by three, plus one extra beginning rest state.

We can consider a score with three notes: 60, 64 and 67 (in MIDI note numbers).

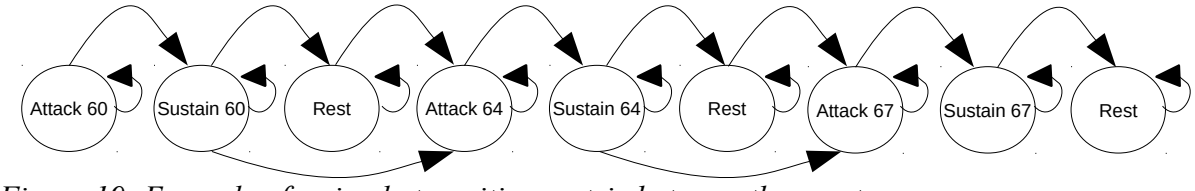

Figure 19: Example of a simple transition matrix between three notes

In the simplest case of the transition matrix, transitions from each state to itself and to

the next state are created. Transitions from each Sustain state, straight to the next Attack, skipping Rest, are also created, in case the notes are played continuously (**legato**).

Probabilities are calculated, dividing 1 with the number of allowed transitions from each state. For example, each of the couple of allowed transitions from Attack will have a probability of 0.5, while each of the three allowed transitions from Sustain will have a probability of 1/3.

Figure 19 showcases the transitions, in the case we expect the performer to play the exact notes of the score, with no skipping or wrong notes, and the goal of the algorithm is just to spot the exact time positions of them in the recording. This transition matrix will be used for the test in 5.2.1

However, for the algorithm to be able to spot mistakes of the performer, some **extra transitions** must be added, that correspond to the mistakes expected. There is also another reason for these extra transitions to be included: the ability for the algorithm to overcome its own errors and continue with the recognition of the rest of the sequence without letting one error destroy the whole process.

Transitions of this kind are the following:

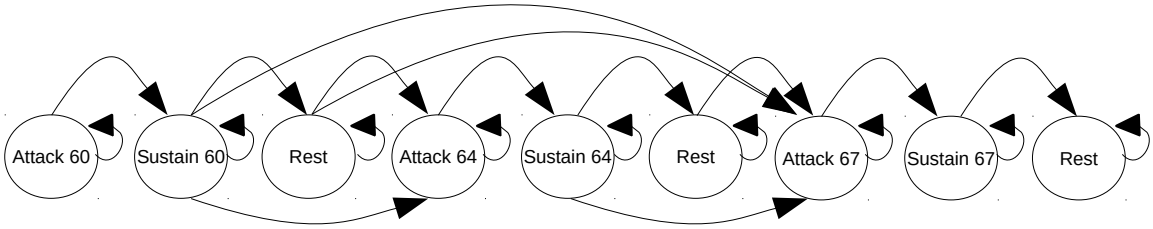

Figure 20: Transitions between three notes with a note jump transition added

In Figure 20, there is the addition of transitions from each Sustain and Rest, 5 and 4 states forward correspondingly (to se*co*nd next Attack) for the case the next note is **skipped**.

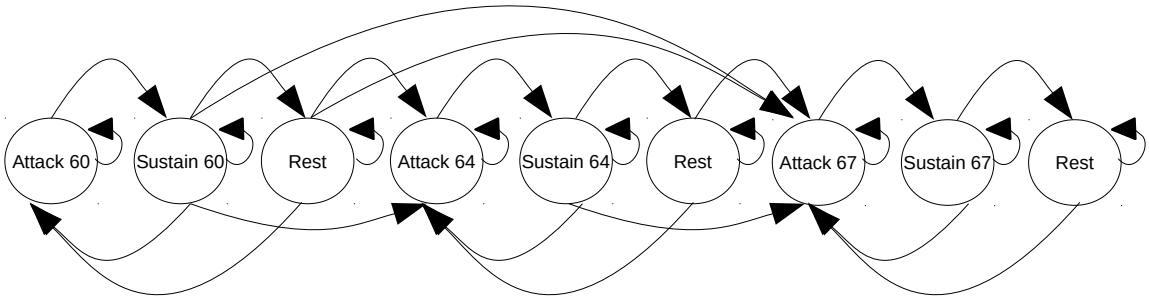

Figure 21: Transitions between three notes with a note repeat transition added

In Figure 21, there is the addition of transitions from each Sustain and Rest, 1 and 2 states backward correspondingly (to the Attack of the same note) for the case the same note is **repeated**.

The transition matrix of Figure 21, will be used for the error handling tests in 5.2.2.

4.2.2. Audio Features

While the score of the piece to be performed, is read and translated into the transition matrix, the audio signal of the performance is read and analyzed into a number of **audio features**. Each of these features is linked to the probability of the existence of one or more states.

The audio features are split into two categories: the pitch independent features and the pitch dependent features. The features of the first category are extracted once for the whole audio signal, while those of the second category are extracted in respondence to a specific note, thus they are extracted as many times as the number of notes of interest.

The signal is segmented into time windows of 512 samples. For each of these widows, each feature takes one value. This results into a considerable reduction of data. The first step for their extraction is the transformation of the signal from the time domain to the frequency domain, through the FFT, as aforementioned in 1.4.1. This results in a picture of the spectra of each of the time frames.

The signal of Figure 22 will be used as an example to showcase the various audio

features.

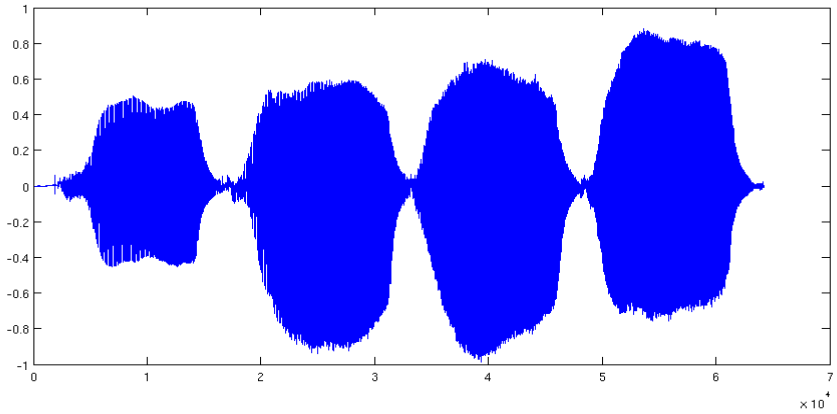

Figure 22: The waveform of a performance of four different notes

The pitch independent features are the following:

- The **log energy** feature is related to the amplitude of the signal and the dimension of loudness. The function from which it is extracted is the following:

$$logEnergy = 10\log\left(\frac{1}{N}\sum_{n=1}^{N} x_n^2\right) \tag{53}$$

x_n is the n^{th} sample of the audio signal, and N is the length of the audio signal in samples.

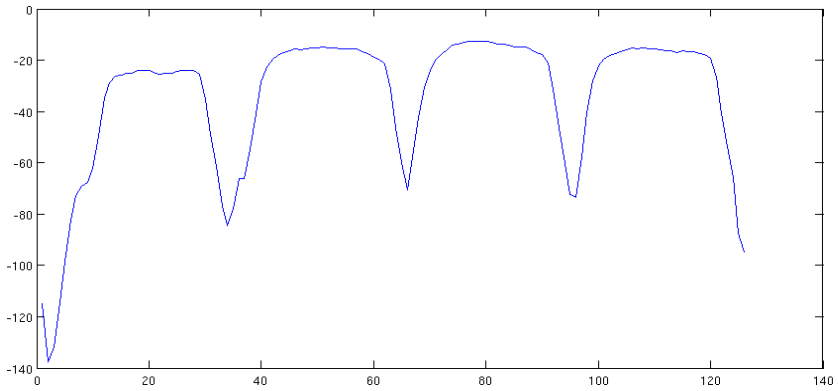

Figure 23: LogEnergy feature

Comparing Figure 22 to Figure 23, we can assume that log energy is a good evident of note activity.

- The **delta log energy** feature, is the difference between the log energy value of a frame and the value of the previous frame. It is an indication of sudden amplitude changes. The equation for its extraction is the following:

$$deltaLogEnergy = logEnergy(n) - logEnergy(n-1) \tag{54}$$

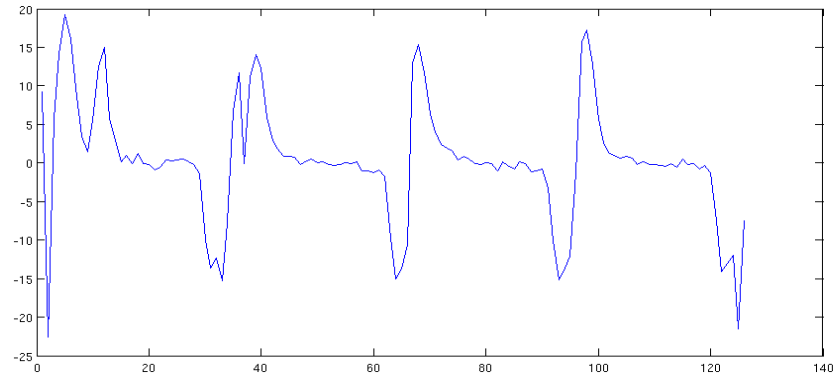

Figure 24: Delta LogEnergy feature

Judging from Figure 24, we can see that delta log energy reaches its peak values at the beginnings of each note.

- The **spectral activity** feature, is a weighted calculation of the energy of different spectral bands and can be considered as an indication of the spectral burstiness [18]. Its function follows:

$$SpectralActivity = \frac{\sum_{m=1}^{M/3} y_m^2 - 2\sum_{(M/3)+1}^{2M/3} y_m^2 + \sum_{(2M/3)+1}^{M-1} y_m^2}{\sum_{m=0}^{M-1} y_m^2} \quad (55)$$

where y_m is the m^{th} sample of the spectra and M is the length of the spectra in samples.

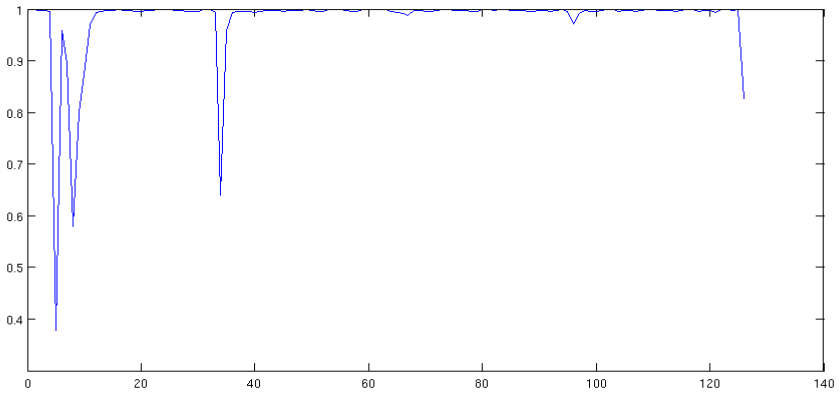

Figure 25: Spectral Activity feature

As Figure 25 suggests, the feature has sudden declines at note onsets.

The pitch dependent features are the following:

- The **Peak Structure Match** (PSM) feature is a measurement of the energy of specific spectral bands that correspond to the harmonics of the note for which it is calculated. Thus, it is a good indication of the existence of the specific note. [15]

$$PSM(w,z)=\frac{\sum_{i=1}^{K} S_i P_i^2}{\sum_{i=1}^{K} P_i^2} \tag{56}$$

This is the Peak Structure Match for note z in time frame w . S_i is the i^{th} sample of the filter spectra and P_i is the i^{th} sample of the original spectra. Both spectras have a length of K samples.

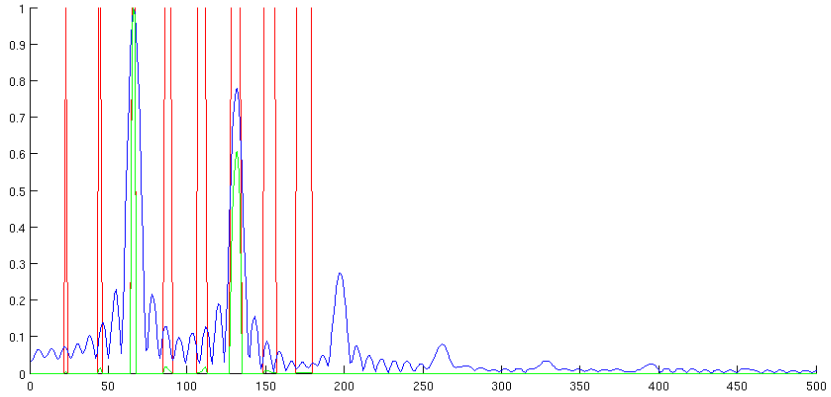

Figure 26: PSM Filter (red) over a spectra (blue) and the remaining spectra (green) for a matching note

As shown in Figure 26, a digital filter is applied on the full spectra of each frame, so that only the energy included in the specific bands of the harmonics of the note is left. This is an example of a matching note, where the two highest peaks of the full spectra are fitting

inside two of the bands of the filter.

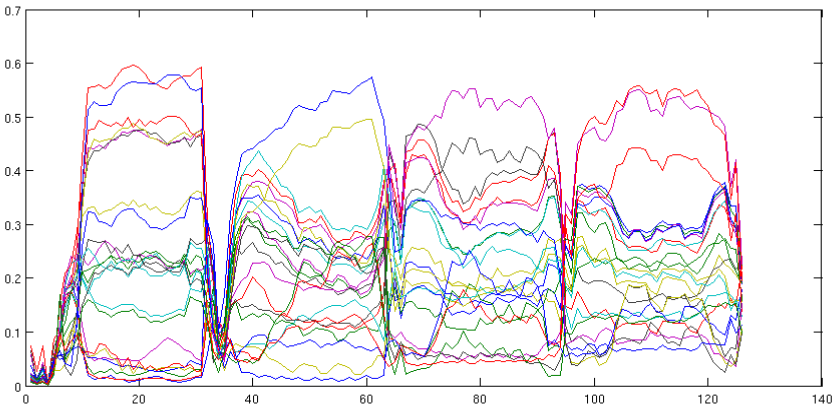

Figure 27: PSM feature for 24 pitches

In Figure 27, PSM values for all the notes of two octaves are depicted (24 notes). It is obvious that at each of the four notes performed, the value of a different PSM function is the highest, being an indication of the current note.

- The **delta Peak Structure Match** feature is, exactly as delta log energy, the difference of two consequent PSM values. It indicates the sudden changes in the pitch dimension.

$$dPSM(w,z) = PSM(w,z) - PSM(w-1,z) \quad (57)$$

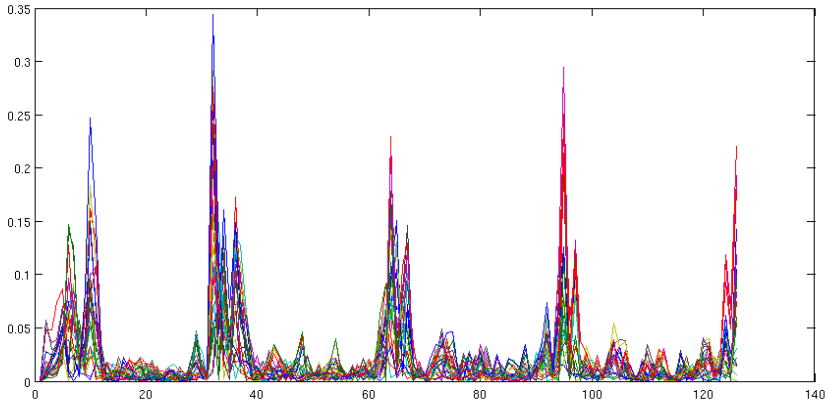

Figure 28: Delta PSM for 24 pitches

Similarly to PSM, Figure 28 shows how a different delta PSM function achieves its peak value at the beginning of each performed note.

4.2.3. Emission matrix

The emission matrix of the model, defines the probabilities of the existence of each of the model states included in the transition matrix, in any of the audio performance time frames. Thus, it is not a square matrix and on one axis there are the states, according to the score, while on the other, there is an index to each frame of the performance.

The probability for a state to exist in a specific frame of the recording is the result of taking into consideration a combination of audio features. The choice of the right final combination of features, is done after several tests (described in 5.2.1) and after studying the semantics of each feature, as aforementioned (4.2.2). Although, each audio feature carries information that indicate the probability of the existence of a specific state, this information has to be extracted by “clearing up” the audio feature signal. This is achieved by mapping the features to the states probabilities according to some Gaussian PDF's and CDF's.

This is done, in order to filter specific value ranges of the feature and map these ranges to the probabilities of the states of interest. In this way boundaries are set for the values a feature can get, within which, it provides a different meaning. For example, we could say that "when energy is within this range or over this value, the probability of having a note sustain is higher, while when energy is within another range or below this value, the probability of having a rest increases". The correct values that set the boundaries are very important, are expressed by the mean and variance of the Gaussian functions and are derived after the training stage (4.2.4).

The mapping functions we examine are three: the normal PDF, the normal CDF and the inverse normal CDF. In this way, we have the ability to limit the probabilities, within, over or under some specific boundaries respectively.

Here is a showcase example of how different kinds of Gaussian functions can be used to map the delta log energy feature of the example in Figure 22 above.

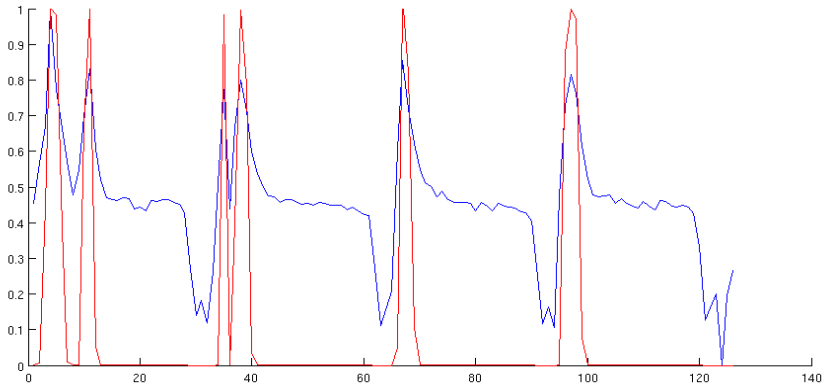

Figure 29: An audio feature (blue) and its mapping according to a CDF (red)

In Figure 29 the feature is mapped according to a normal CDF, with $\mu=10$ and $\sigma^2=2$. What we end up with is an indication of the beginning of each note.

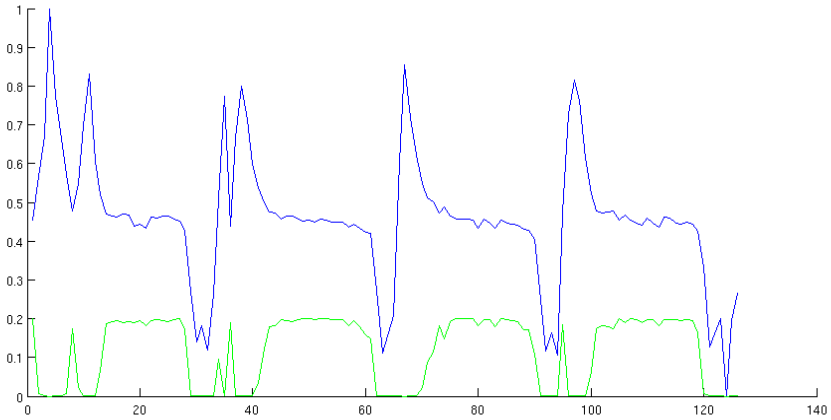

Figure 30: An audio feature (blue) and its mapping according to a PDF (green)

In Figure 30, a normal PDF function is used to map the feature, with $\mu=0$ and $\sigma^2=2$. The derived probability is indicative of the main body of each note.

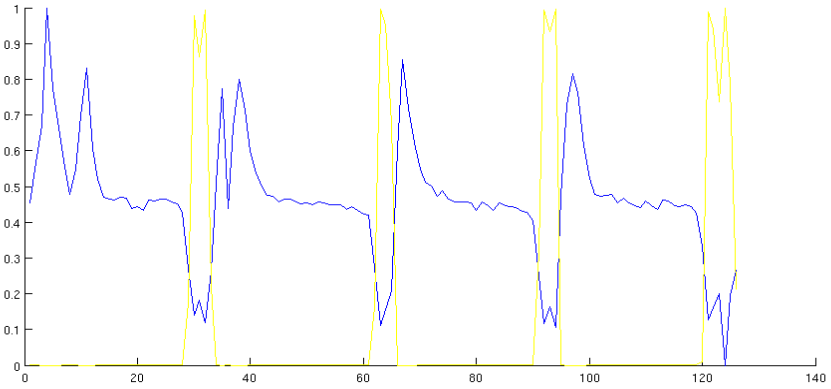

Figure 31: An audio feature (blue) and its mapping according to an inverse CDF (yellow)

In Figure 31, an inverse CDF with $\mu=-10$ and $\sigma^2=2$ is applied with the resulting yellow signal, being indicative of the pauses between each note.

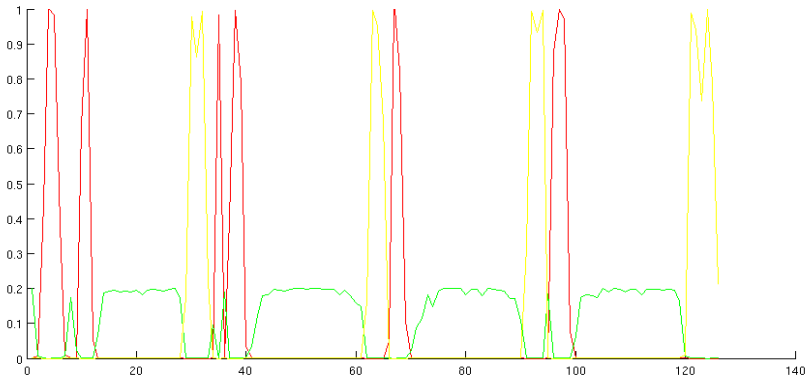

Figure 32: Three different mappings of the same audio feature

Putting these all together, when we look at Figure 32, it is obvious that at different times, another probability function is the highest, each of which has derived by a different mapping function of the same audio feature. In this simple example, we can suppose that the red signal can be considered as an attack probability, the green signal as a sustain probability and the yellow signal as the rest probability.

Taking all these into consideration, we are trying to exploit all the meaningful information that can be derived from the audio features and map them and combine them in the way that they can provide us all the probability functions we need for all the states to fill the emission matrix. Probabilities are separated into states and all probabilities for a specific state are multiplied, so that the final probability results. Pitch dependent probabilities referring to the same note are also multiplied between them. Thus, the resulting emission matrix, for example, for a score including only 2 unique notes would include the following probabilities: Attack of Note X probability, Sustain of Note X probability, Attack of Note Y probability, Sustain of note Y probability, Rest probability.

The choice of the correct mapping functions is again reached after the several tests (5.2.1), as well as semantics as those derived from the example above.

4.2.4. Training

The first stage of the system is the training stage. During this stage, we provide the algorithm with a dataset of audio performances of known content to be used for training the model, hence allowing to decode and align additional performances. To improve performance alignment subsequent to training it is recommended that the training dataset represents the same instrument, acoustic environment and including notes, with the performance to be aligned in the decoding phase. This is the reason why prior rehearsals of the piece to be performed is considered to be an ideal training dataset.

The training process involves analysis of the distributions of each feature values on previously annotated audio performances. This means that the algorithm attempts to estimate the values of each feature for each HMM state, for example attack, sustain, rest of specific notes. Based on the values of features for the audio segments corresponding to each state, the means and variances of Gaussian functions are estimated, using the EM algorithm, as described in 3.1.5.

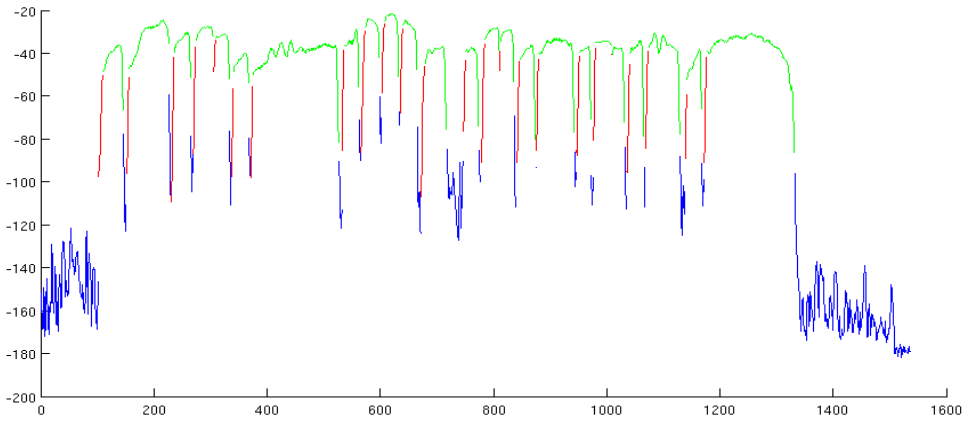

Figure 33: LogEnergy feature, with highlighted areas of rest (blue), attack (red) and sustain (green) frames

Figure 33 is an example of the log energy feature of an audio performance,

discriminated into three different color areas according to the pre-labelled state of the note of each time. This is used to train the pitch independent features. The distribution of the feature values in each of these areas is shown in Figure 34, Figure 35 and Figure 36.

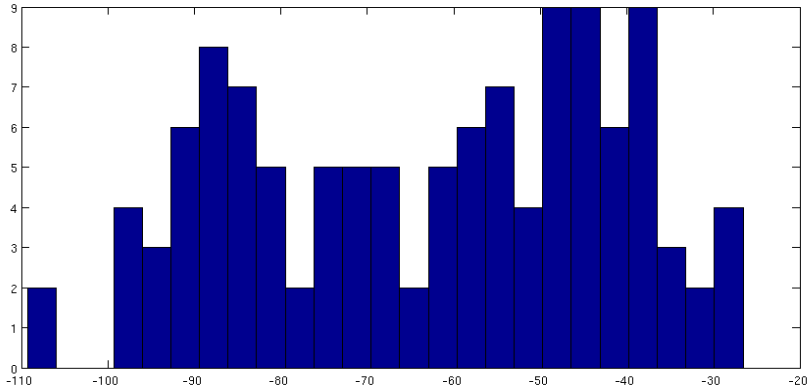

Figure 34: The number of occurrences (vertical axis) of each feature value (horizontal axis) on all the frames labeled as attack

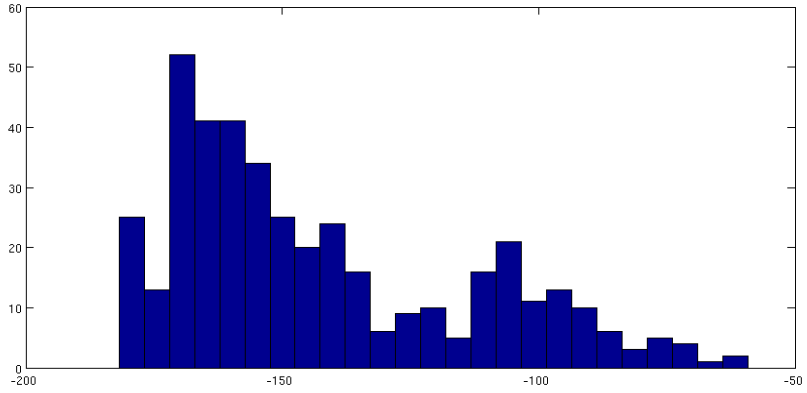

Figure 35: The number of occurrences (vertical axis) of each feature value (horizontal axis) on all the frames labeled as sustain

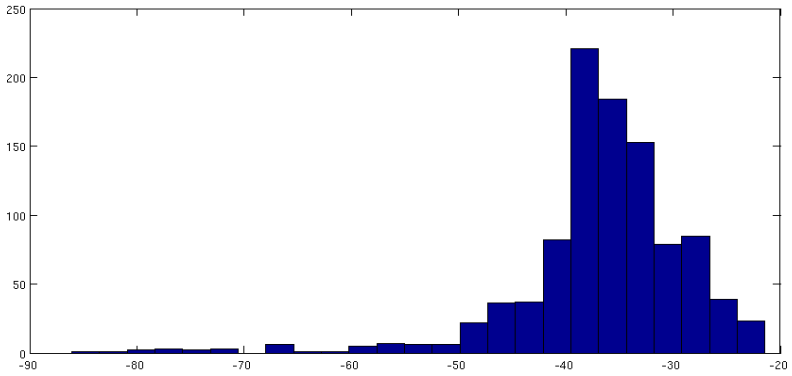

Figure 36: The number of occurrences (vertical axis) of each feature value (horizontal axis) on all the frames labeled as rest

These distributions result in the Gaussian functions of Figure 37. The means and variances of these functions are stored as the trained model. The trained model is used during decoding (alignment) to infer the state that is more likely to be observed for a given distribution of feature values.

In the same manner, pitch dependent features are trained, taking into consideration pitch annotations on training dataset.

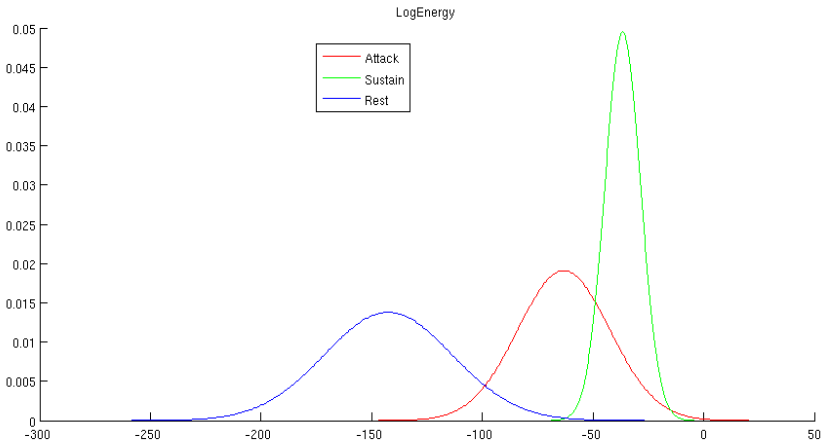

Figure 37: Gaussians derived from the three distributions of the feature values on each frame group

In the proposed approach, observation probabilities are estimated by computing a separate Gaussian distribution for the values of each feature in each HMM state. Although this neither a multivariate Gaussian nor a GMM, the estimation process is equivallent to that of the Expectation-Maximization algorithm, which is conventionally used for training HMMs, however reduced to a single component. In this manner, the system will be compatible with future developments concerning multidimensional models or GMM's with more components. This type of simplified training has been adopted in previous computational methodologies for ASA, for example [18] and [67], where it has been referred to as Discriminative Training.

4.3. Algorithm / Data flow diagrams

The algorithm for the audio to score alignment system has been developed for the MATLAB environment. This is a popular numeric computing and programming environment, which makes direct and easy experimentation and testing, easy. Its versatile matrix manipulation and its easy graphical depiction functions, as well as the HMM and signal processing toolboxes, were proven extremely useful for the implementation of our system.

The main files of the system are the `train` and `asalign` functions. In addition, the `make_data_set` and `prepare` function are used for the labeling and creation of the dataset (5.1.3). Many processes that are repeated within these files are broken into smaller different function files, such as `load_score`, `fextract` etc. Finally, functions for the watching of graphical depictions of the various stages of the algorithm are included, such as `display_gaussians` and `display_classes`.

Third party toolboxes and functions were also used in this project. MIDI Toolbox was used for loading information from MIDI files [68]. `Weka2matlab` functions were used for loading and saving ARFF files [69]. Function `em1_dim` [70] was used for the EM algorithm. Finally, an implementation of the YIN algorithm was used for the creation of the dataset

[71], [72].

4.3.1. Training

The first stage of the implemented system is that of the training and is called by the `train` function as follows:

```
train(arff_file_names_list, trained_gaussians_file_name);
```

It takes two arguments: a cell array with all the names of the files of the dataset, and the name to be used for the Comma Separated Values (CSV) file, where the means and variances of every feature for every state will be saved. This will be used as the knowledge for the trained algorithm.

The files of the dataset must be in the form of Attribute-Relation File Format (ARFF). This is a text file format introduced by the University of Waikato, for their machine learning software WEKA. The structure of an ARFF file is divided into two sections: the header and the data. The header contains the names of all the attributes, their types and the class attribute.

In the case of the Audio to Score Alignment system, each ARFF file corresponds to an audio performance. Each attribute corresponds to an audio feature, independent to pitch or for a specified pitch. The penultimate attribute is the note label in MIDI note numbers. The class attribute corresponds to the note state. Each line of the data contains the values of all the audio features for one frame and a label of the current note and state. An example follows:

```
@relation Ex1_V_take001

@attribute LogEnergy real
@attribute DeltaLogEnergy real
@attribute SpectralActivity real
@attribute PSM57 real
```

```
@attribute DeltaPSM57 real
@attribute PSM58 real
@attribute DeltaPSM58 real
@attribute PSM61 real
@attribute DeltaPSM61 real
@attribute PSM62 real
@attribute DeltaPSM62 real
@attribute PSM64 real
@attribute DeltaPSM64 real
@attribute PSM65 real
@attribute DeltaPSM65 real
@attribute PSM67 real
@attribute DeltaPSM67 real
@attribute PSM69 real
@attribute DeltaPSM69 real
@attribute PSM70 real
@attribute DeltaPSM70 real
@attribute Note real

@attribute Class {Attack, Sustain, Rest}

@data
-167.4266, -2.6188, 0.34157, 0.016061, 0.002689, 0.017311, 0.0023044,
   0.016641, 0.0038058, 0.015864, 0.0043779, 0.013588, 0.0017621,
   0.014051, 0.00010179, 0.016362, 0.0040728, 0.019718, 0.0077481,
   0.013634, 0.0053223, 0, Rest


-45.9281, 5.9246, 33693.5115, 0.18096, 0.040343, 0.16726, 0.011531,
   0.16684, 0.001012, 0.39187, 0.043748, 0.18104, 0.018795, 0.11243,
   0.051482, 0.16719, 0.031699, 0.082242, 0.044748, 0.11041, 0.015902,
   62, Attack

-41.8491, 4.079, 55024.818, 0.19551, 0.014553, 0.12148, 0.045783,
   0.14168, 0.02516, 0.35907, 0.032798, 0.16473, 0.016309, 0.13805,
   0.025617, 0.12351, 0.043686, 0.10234, 0.020095, 0.10711, 0.003297, 62,
   Sustain

-37.1183, 4.7309, 86481.6595, 0.15518, 0.040337, 0.11579, 0.0056918,
   0.13963, 0.0020565, 0.38033, 0.021255, 0.1709, 0.0061651, 0.11019,
   0.027862, 0.15769, 0.034183, 0.067156, 0.035181, 0.11748, 0.010371,
   62, Sustain
...
```

The CSV file name attribute refers to the output file. The means and variances will be saved there as follows:

```
-
   54.623,10.523,1802.8,0.35923,0.10443,0.25016,0.064464,0.26253,0.054719
   ,0.44064,0.0516,0.35992,0.04029,0.33732,0.035154,0.49029,0.13334,NaN,N
```

```
   aN
18.684,6.5511,2338.7,0.095337,0.07363,0.056354,0.040655,0.020901,0.054397
   ,0.054314,0.049455,0.041275,0.030846,0.045914,0.03189,0.026304,0.14232
   ,NaN,NaN
...
```

Each couple of lines corresponds to a state mean and variance and each comma separated value within a line, refers to a feature. The values denoted as NaN (Not a Number) correspond to features not related to the specific state. This information is saved in a file named with the second argument, ending with "_TG.csv".

After the algorithm loads the ARFF file, it proceeds to grouping together all the frames of the same state, or same state and pitch, and calculating the mean and variance of the distribution of each feature in them.

The following listing provides the pseudo-code of this process:

```
For each state
   Find all the frames (lines) labeled as the current state and save
   their...                              indexes in "idx1"
   For each feature
   If the feature is pitch dependent
      Find all the frames labeled with the note of this feature and...
                        save their indexes in  "idx2"
      Find all the reoccurring indexes in both "idx1" and "idx2" and...
                        save them in "idx"   Else if the feature is pitch
   independent
      "idx" is equal to "idx1
   End if
   End for
End for
```

All the frames with the indexes contained in "`idx`" will be analyzed with the EM algorithm.

It should be pointed out that the training can be accomplished, either based on a single audio performance (one ARFF file), or a number of audio performances (list of multiple ARFF files).

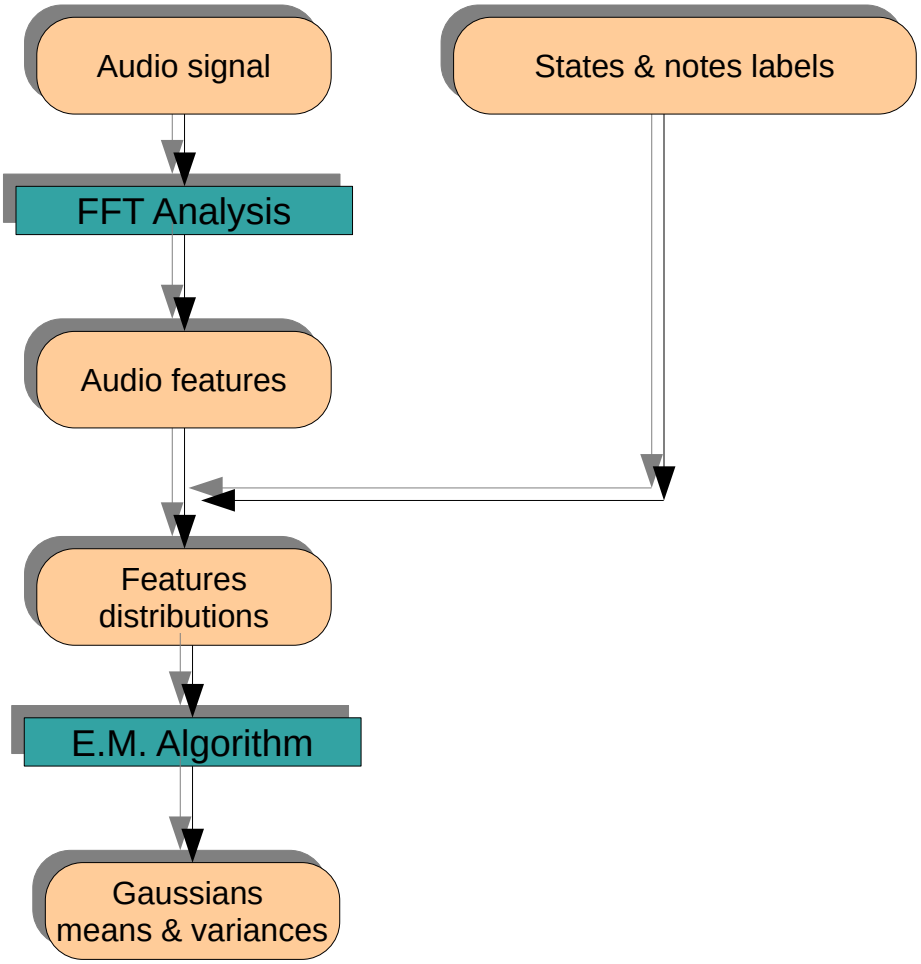

Figure 38: Training function basic data flow diagram

4.3.2. Aligning

The second and main stage of the system is the aligning stage. It is called as follows:

```
[missed misaligned offset]=asalign(audio_file_name, score_file_name,
   trained_gaussians_file_name, transport)
```

It takes four arguments. The first stands for the name of the audio file of the performance to be aligned and it can be in WAV or AIF format. The second argument stands for the file name of the score to be aligned and it must be in MIDI format. The third

argument stands for the CSV file with the trained Gaussians knowledge that was created after the training stage and will be used to map the features of the audio performance to the desired state probabilities. Finally, the last argument refers to the difference in the key between the performance and the score in semitones, in case the piece was transported before performed.

There is also another file loaded with the ground truth positions of the notes of the audio performance. This is used for the tests and the evaluation of the system and it will be fully described on the following chapter. The function returns the percentage of the missed and misaligned notes, as well as the offset, which is the mean distance between the correct and the aligned positions of the notes.

At the beginning of the function a cell array is defined, inside which, there are declared the number of features, their names, whether they are pitch dependent features or not and the type of the gaussian functions according to which, they are related to each probability. This makes easier the addition or removing of features, as well as the experimentation with different gaussian functions. The features are defined as follows:

```
features{X,1}=[a, s, r, pd]; features{X,2}='FeatureName';
```

The variables `a`, `s` and `r` correspond to the mapping function of choice and can take one of the following values:

0: the feature is not taken into consideration for this state

1: the feature is mapped according to a normal CDF

2: the feature is mapped according to an inverse normal CDF

3: the feature is mapped according to a normal PDF

The variable `pd` corresponds to whether the feature is pitch dependent (1), so that it should be extracted for all the notes contained into the score, or it is pitch independent (0), so that it should be extracted only once.

An example of an audio feature definition is the following:

```
features{1,1}=[1, 3, 2, 0]; features{1,2}='LogEnergy';
```

After all the files are loaded and the defined features are extracted from the audio signal, the calculation of the probabilities begins.

This is achieved by mapping all the features, according to the directions of the features cell array, resulting in probabilities for each state of each note coming from each feature. Then, all the probabilities referring to the same state are multiplied, resulting in a final probability. The rest probability is always pitch independent and is calculated only once. For example if the score to be aligned contains three unique notes, the final probabilities will be saved in a cell array similar to the following:

```
final_prob{1,1}, final_prob{1,2}, final_prob{1,3}
```
for the attack probability for each of the three notes

```
final_prob{2,1}, final_prob{2,2}, final_prob{2,3}
```

for the sustain probability for each of the three notes

```
final_prob{3,1}
```

for the rest probability

Afterward, the creation of the HMM matrices takes place. The transition matrix is created according to what has been described in 4.2.1. The emission matrix is created, using the final probabilities mentioned above.

Once, the matrices are filled, the Viterbi algorithm (3.2.3) is called to calculate the best alignment sequence, and the separation of the notes of the performance waveform is graphically depicted. The success evaluation is also calculated, according to 5.1.1, and displayed.

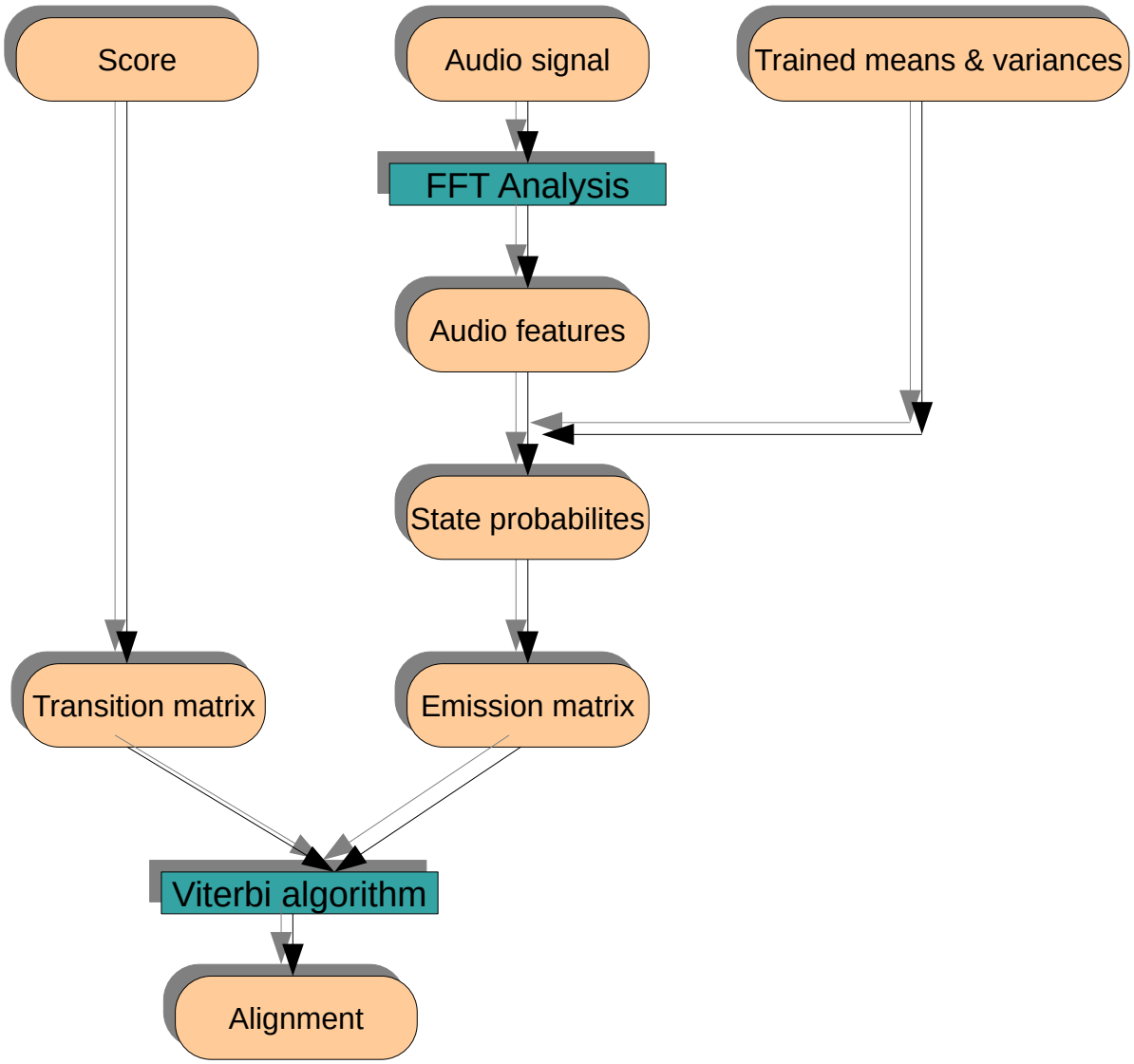

Figure 39: Alignment function basic data flow diagram

5. System Evaluation

5.1. Evaluation process

5.1.1. Approaches / Measures

The evaluation stage of an ASA system is important for judging the success levels of the algorithm. In this manner, the system can be compared with other aligning systems. For this reason, there must be a standardization of the evaluation measures that are used by the several systems. In addition, the evaluation stage can be proven essential for the development of a specific system, giving the developer the opportunity to compare different methods, in order to reach to the optimal approach, as well as to verify the benefits gained by the training stage.

An ASA system can be evaluated in a **subjective** or an **objective** manner. While subjective evaluation is not suitable for comparing different systems, it can contribute considerably in the development process of the system. Methods of subjective evaluation include the displaying of the detected note onsets over the performance signal waveform (Figure 40) or its spectrogram, listening

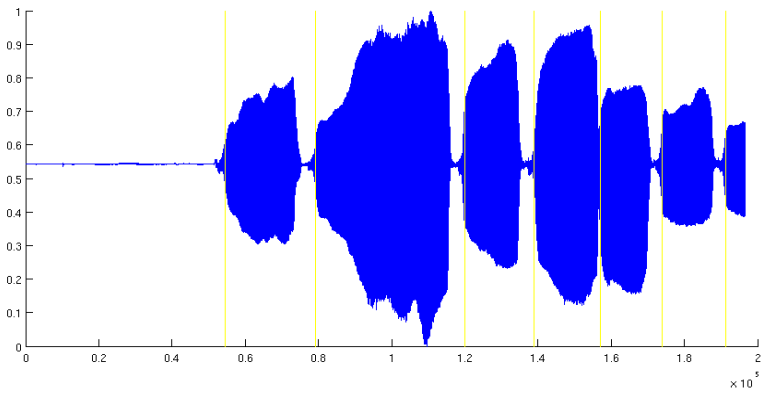

Figure 40: Example of detected note onsets graphically placed upon the performance waveform for subjective evaluation purposes

to each segmented note separately, or listening to the performance, accompanied by a short click sound at each point a note is detected. Finally, the alignment can be exported as a MIDI file, which when listened to, must be in accordance to the performance recording. This last method can be also considered as a primary application of the system by itself.

However, all these subjective methods can be tiring, time consuming and inaccurate when we are confronted with large datasets, that are required for the exact evaluation of ASA systems For this reason, there has been a discussion on some objective and standardized measures for judging the aligning algorithms. The objective evaluation, requires some references. Each audio performance to be aligned must be accompanied with a ground truth file, which includes all the time onsets to be detected by the aligning algorithm. These time onsets is what the evaluation algorithm will use as a reference to judge the success level of the aligning.

Regarding these, the following measures are used for the evaluation of the detection of each note i in the ground truth file, as defined in [73]:

- **Error**

$$e_i = t_i^e - t_i^r \tag{58}$$

This measure shows the time difference between the estimated onset t_i^e in the ground truth file and the onset t_i^r estimated by the algorithm.

- **Latency**

$$l_i = t_i^d - t_i^e \tag{59}$$

This measure is being used by real-time systems and shows the time difference between the estimated onset t_i^e and the time t_i^d the algorithm detects it.

- **Offset**

$$o_i = t_i^d - t_i^r \tag{60}$$

This measure is the final offset between the detection time t_i^d and the reference time t_i^r . In case of an offline system the offset is equal to the error.

- **Missed notes** is the number of note onsets in the ground truth file that are not detected by the algorithm and **misaligned notes** is the number of notes that have been detected, although their offset is bigger than a limit θ_e (e.g. 300 ms).
- The **missed notes rate** p_m and **misaligned notes rate** p_r of the algorithm, derive from the percentages of these two measures. These percentages constitute the final success rate of the algorithm:

$$SuccessRate = 100 - p_m - p_r \tag{61}$$

Other useful measures include:

- **Piece completion** p_c

This is the percentage of the notes in the ground truth file, that are recognized by the algorithm before it hangs or stops detecting correctly notes.

- **Average Latency** μ_l

This is the mean latency of the system (for real-time systems).

- **Average absolute offset** μ_o

This is the average offset of the system for all the performances it is tested on.

- **Average imprecision** μ_e

This is the average error of the system for all the performances.

- **Variance of error** σ_e

This is the spread of the distribution of the this error.

For further discussion on ASA evaluation, one can address to [73], [15] and [74].

5.1.2. Performances dataset

The system developed for this thesis has been tested on a specific dataset of recorded

performances. This dataset consists of two different short monophonic melody lines for flute and for violin. These two monophonic lines are the two parallel voices of the same polyphonic piece "*Example 1*", composed for such purposes by Gabriel Negrin. Part of the tests will be applied on the polyphonic performance too (5.2.3).

Each piece is performed ten times with several interpretation variations. The choice of flute covers the case where the note envelope of the instrument has a strong attack, thus, the detection of the note onsets relies mostly on the energy variation. The choice of violin covers the case where the instrument has no strong attack, thus the detection relies mostly on the pitch variation. In addition, violin playing involves no stable pitch, which adds some extra difficulty, the algorithm will be tested on.

Additionally, some alterations of these performances have been created automatically. These alterations include some note repeats, while some notes are completely removed. This is done for the tests of the error handling part of the algorithm (5.2.2). Specifically, for the performances of each instrument, the first five have a random note repeated once and the last five have a random note removed.

This dataset will not only be used for testing, but as the training set too. As mentioned in 4.2.4, the trained Gaussians data base can be derived by one or more performance. Thus, each of the ten performances of each piece, results in a trained data base. In addition, there is a data base created by the whole set of the performances.

5.1.3. Dataset creation

The creation and preparation of the dataset includes two tasks. Firstly, all the performances must be labelled with the correct states and notes, and the ground truth files must be created. Secondly, their signals must be analysed to all audio features of interest, and their values are saved in A.R.F.F. files. These files are reduced versions of the original performances, and will be used for the training procedure.

The labelling of the performances is achieved a combination of automatic and manual means. YIN [71], a pitch detection algorithm, is firstly applied in each signal. This algorithm returns a value of aperiodicity. The existence of a note in a specific frame of the signal can be considered as positive, when the aperiodicity falls under a predefined limit. All other frames are labelled as rest. When a note is detected on a frame following a rest, this frame and a number of the following frames are labelled as attack.

The next stage for the labelling of the performances is the manual correction of the automatically detected states. The performance is graphically displayed in an application called Sonic Visualizer. This application gives the user the ability to listen to waveforms, and at the same time have graphical depictions of audio features in various ways, either detected by the program itself, or imported. In this case, we import the data detected by our preparation algorithm. What follows is the careful by hand correction of the detected states, watching and listening to the waveform. We consider as rest frames, the frames where the instrument has paused. We also try to mark as attacks the first frames of each note performed, as long as there is a rise in amplitude.

The dataset for the error handling tests (5.2.2) has derived from the editing of the previously used performances. The same flute and violin recordings have been automatically processed, and a unique random note has been repeated (for recordings 1-5) or missed (for recordings 6-10) in each file. The new files including the performance mistakes are saved as the new dataset to be used for the following testings.

The creation of the training set simply includes the conversion of each audio file into an ARFF file, which instead of the signal itself, includes all the audio features describing it. The state and pitch labels are also included in the files, as described in 4.2.4.

5.2. Tests and results

Each test comprises applying the aligning algorithm to each one of the ten

performances included in our dataset. The result of the test is the overall success rate of all the alignments. The tests that were performed are divided into three groups.

Firstly, we have to specify the optimal combination of features that are associated with each state probability, as well as the Gaussian function, according to which, it will be mapped. This is done by altering the variables in the features definitions described in 4.3.2, according to Table 9 and running the align function for each signal.

Secondly, we edit the transition matrix, so that it includes the extra transitions required for coping with playing mistakes and aligning errors (as showcased in 4.2.1 and Figure 21) and perform the align function for all the signals again, as well as the edited signals including mistakes.

Finally, the algorithm is tested on mixed signals of both flute and violin.

5.2.1. Features & Gaussians

At first, we examine how only the energy related features behave. The first test makes use of the LogEnergy and DeltaLogEnergy features. In this first attempt, we apply normal PDF's for the extraction of all probabilities from the features. The results are extremely poor, since the algorithm failed completely on most of the performances. This happened because there are frames where the probabilities derived from the PDF's have zero values for all three states, giving no option to the algorithm for a transition. (Test 1)

We try to eliminate this problem by applying normal CDF's instead. We use the inverse CDF for the rest state on both features, since we expect a fall of the energy when there is no note played. There is a great improvement on the results. As expected, we notice that the detection based on energy related features have very good results on flute, an instrument with sudden dynamic variations, but it seems not suitable for instruments like violin, where the note onsets are not accompanied by sudden raises of the energy. (Test 2)

We also try a combination of CDF's and PDF's for a better discrimination between the probabilities of each state. Although, the results of the tests on flute have a small decrease,

there is a significant increase on the results of the tests on violin, since the use of both a CDF and a PDF seems to contribute remarkably on the discrimination between attack and sustain. (Test 3)

Afterwards, we test how the addition of the spectral activity feature affects the results. We use a CDF for the sustain and rest states, and an inverse CDF for the attack state, since the feature has such a behaviour, as shown Figure 25. The detection of flute is improved in some small amount, since spectral activity is supposed to detect attacks with high frequency rich spectras. On the other hand, violin has no such spectral characteristic, thus the addition of this features seems to deteriorate the results considerably. (Test 4)

Subsequently, the algorithm is tested, making use of only the pitch related features. Again, we firstly apply normal PDF's to all the features and states. The use of the PSM features gives the best results until this point. This is an indication of how important role the detection of the pitch content plays in comparison to the detection of the dynamic content. Moreover, it is the first test that the detection of violin has such a success rate, which is completely attributed to the pitch related feature. Violin note changes can be detected mainly by the changes in pitch. (Test 5)

Next, we test these features, mapped according to CDF's. Only a small improvement on the results has been achieved on the violin performances. (Test 6)

Followingly, we test the combination of PDF's and CDF's. Only a small variation on the results of both instruments has been noted, when using the combination of gaussian functions. (Test 7)

Finally we test the algorithm, making use of both the energy and PSM derived features.

We use the PSM derived features just for the attack and sustain states, since the rest state has no relation to pitch. The use of both energy and pitch features rgives us very good results. (Test 8)

On our next test, we use a combination of PDF's and CDF's, this time for both types of features. The use of PDF's for the sustain probability seems to give a small rise on the

overall success rate. (Test 9)

Finally, we add the Spectral Activity feature. The success rate remains the same, while the overall offset for the violin decreases slightly. (Test 10)

Test	Flute	Violin	Average
1	10,00%	10,00%	10.00%
2	97,08%	24,09%	60.59%
3	92,92%	91,82%	92.37%
4	95,83%	49,55%	72.69%
5	99,17%	94,55%	96.86%
6	93,33%	97,73%	95.53%
7	99,58%	93,18%	96.38%
8	100,00%	98,18%	99.09%
9	100,00%	98,64%	99.32%
10	100,00%	98,64%	99.32%

Table 8: Overall success rates of each set of tests

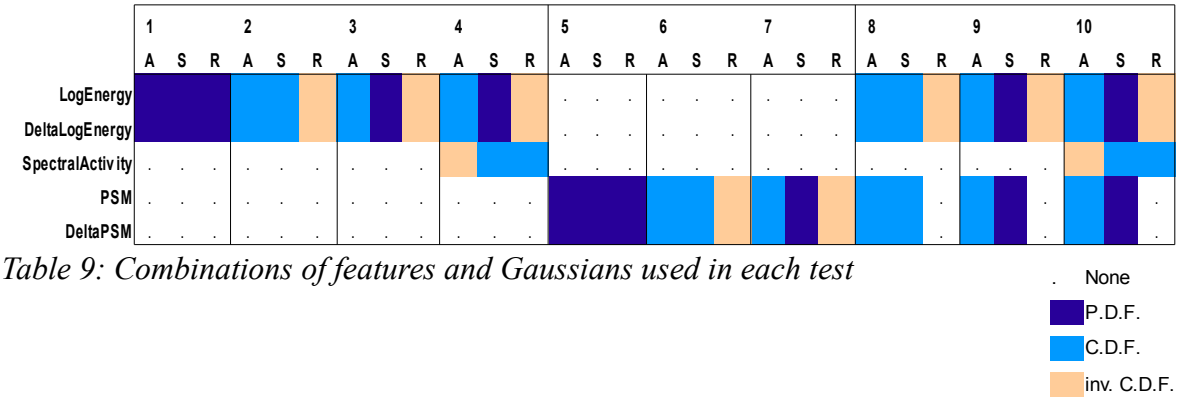

Table 9: Combinations of features and Gaussians used in each test

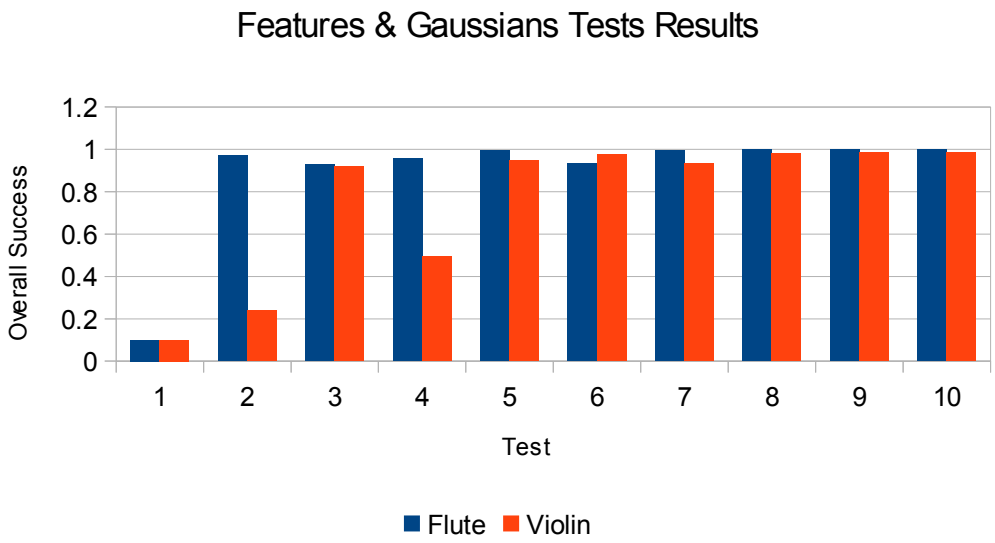

Figure 41: Results for each feautures and gaussians combination test for flute and violin

We can divide all the tests performed into three groups, as shown in Table 9: those using energy related features, those using pitch related features and those using both types of features. Table 8 And Figure 41 informs us about the overall success rates of each test. It is easy to see that pitch related features offer us better results than energy related ones. As we expected, the use of PSM improves the alignment on the violin signals more than the flute. Finally, the combination of all the features in the form of the last test, gives the overall best results.

5.2.2. Error handling

The transition matrix of the algorithm for the tests performed so far, contain no extra transitions, as showcased in Figure 19 of 4.2.1. This means that the performances in the recordings are expected to correspond to the score with no mistakes, while only their timing differs. In the following tests, extra transitions showcased in Figure 21 will be included in

an attempt to handle the case the performer makes mistakes such as missing or repeating a note.

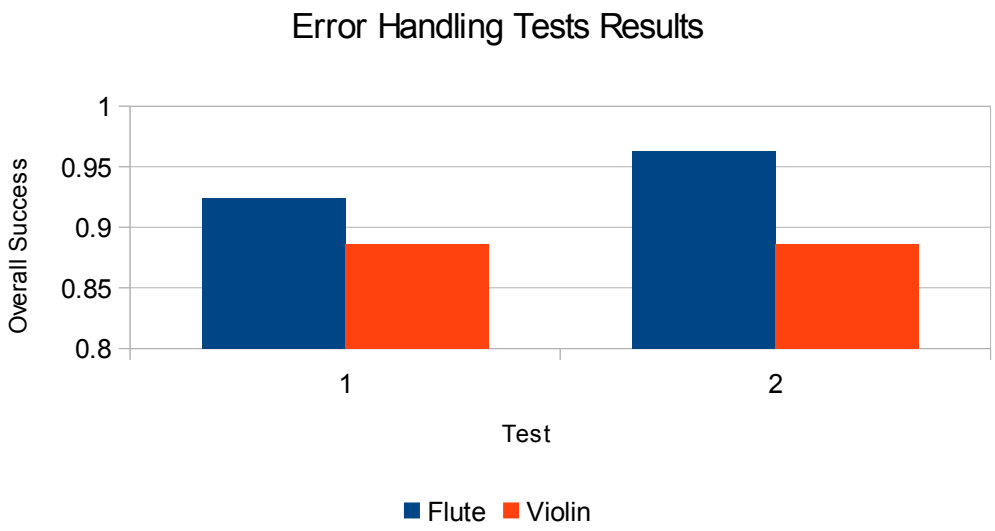

Figure 42: Results for error handling tests on signals containing errors (1) and on the original signals (2)

What we notice from Figure 42 is that there is a decrease on the overall success rate, similar to that of the previous tests. This means that this small fail rate is not attributed to not detecting the performer mistakes. Contrary, this decrease is expected, since now, the algorithm has the ability to detect note repeats or missed notes on frames of the recording where they do not exist, ending up with a small amount of "false alarms".

5.2.3. Polyphonic performances

The algorithm has been also tested on the signals of the polyphonic performances of both flute and violin playing together. However, it should be noted that when the algorithm

is tested on a polyphonic signal, the score given to it concerns only the line of one of the two instruments. The trained Gaussians for this instrument are the ones to be used, and the note onsets of this instrument will be detected. This means that the mixed signal of the other instrument is considered as a background noise, the algorithm is trying to surpass. This is a quite different task from providing the algorithm with the score of all instruments playing and expecting the detection of the onsets of all the notes played.

Both test sets refer to the same audio performances of both flute and violin playing together. However, in the first set, the flute is attempted to be detected, while the violin on the second one.

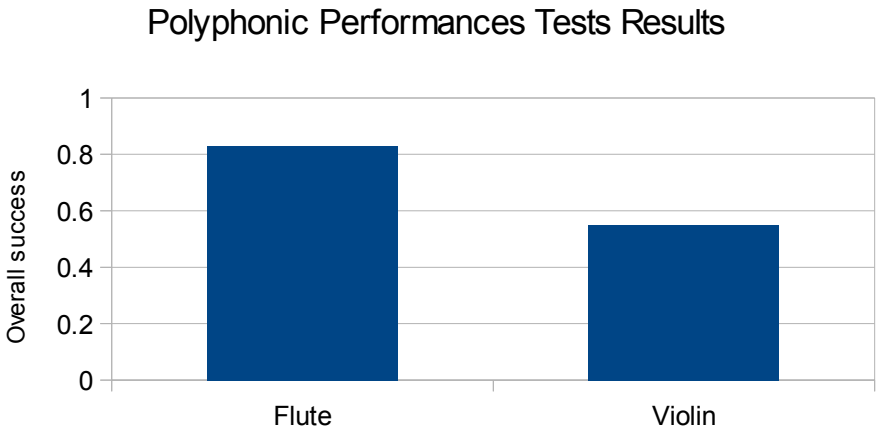

Figure 43: Results for tests on the same polyphonic performance, detecting flute or violin on each test.

The results are generally of medium success. An interpretation of them could be that, the algorithm is not capable enough of following a single instrument, inside a polyphonic performance, but part of unexpected background noises on a performance can be surpassed without affecting the detection process.

6. Conclusions

6.1. Contributions

Although the goal of this thesis was to develop a basic ASA system, there are a few small features of the system that differentiate it from previous related work and can be regarded as possible small contributions.

Firstly, there is a key difference in the way the score is transcribed into the transition matrix. The probabilities for each transition between the states are set by allocated equally to the number of all aloud transitions. Practically, this means that each note duration provided by the score is not taken into account. If this was the case, the number of sustain states for each note would vary according to the note duration and so would be the probabilities of their transitions. Such an approach, is described in detail in [34], accompanied by the related equations. Although, with our approach, we ignore this factor, which might contribute in a better detection, we come up with less biased results. In other words, the score provides us only the right sequence of the events, while only the performance signal constitutes the information used for detecting the correct position of each event on the time scale.

Another unique aspect of our work is the detailed research and testing procedure for the optimal selection of the Gaussian functions for mapping the audio features to meaningful probabilities. This reached a unique combination of PDF's as well as CDF's and inverse CDF's associated with each feature and probability, that was proven to provide the most faithful decoding results.

Finally, the system was tested on polyphonic signals, for which additional melodic lines were not aligned, but instead treated as background noise to be surpassed. The test results were not discouraging.

6.2. Deficiencies / Future work

First of all, we must regard that, although we attempted to cover some of the basic performance characteristics one can be confronted with, the testing performances as well as the training set cannot be considered fully sufficient, especially when compared with those of the participants in competitions such as MIREX. This is the reason why, although the success rates achieved by our system are far larger than that of those systems, an actual comparison with their success rates would not be accurate and meaningful. A larger scale dataset for the system to be trained and tested seems to be an imperative step towards its further evolution. However, the recordings used for this thesis, are realistic examples of real-world music performances, even without trying to include the special cases of the most rare and difficult to handle, characteristics.

Some of the small fail rates of the test results, should be attributed generally to the training stage, and more specifically to the training set creation. The combination of automatic and manual means for the pre-labelling of the performances leaves enough room for tagging imperfections that can affect the trained Gaussians data base.

Another reason could be that the range of audio features that could be used by such an algorithm is much wider than those actually used in our system. What we tried to do, is focusing on the features that have been already proved to be useful for such a task, since an attempt to examine the whole number of features or even trying to create new ones that could capture useful details would be a long and difficult procedure. However, the testing of more features, the creation of new ones and even the use of more than one dimensions of features for one probability, are all steps towards the further development of this ASA system.

Furthermore, part of the fail rates of the algorithm might be avoided, if probabilities distributions are applied in the transition matrix, instead of the straightforward division of the probabilities, as mentioned in 6.1.

However, maybe the most important element that our audio to score alignment algorithm is missing, is that of being able to fully cope with polyphonic performances. This

means that the score to be aligned contains chords and that either that the instrument performing is a polyphonic instrument, such as a piano or a guitar, or that there are more than one instruments performing. From a technical point of view, the basis for such a development is already included in the algorithm and is that of the PSM features. The difference is that the filters for each chord will be combinations of the harmonics of all notes included in the chord. The main structure of the transition matrix also remains the same, with only small differences. Since the chord is not always played simultaneously, the attack of one note may sound at the same time with the sustain of the other. This can occur either because the score of the piece imposes so, or by a performer mistake or even because of the instrument's nature. That creates the need for a slightly more complex structure of the transition matrix.

The other element not found in our algorithm is that of the ability to run in real-time. However, the variation of the Viterbi algorithm for such a case is cited and the only reason it was not finally included in the algorithm is that the programming environment used is not suitable for real-tme applications.

6.3. Summary

In general terms, the system developed for this thesis can be considered as a major first step for the development of a complete and innovative ASA system. In this manner, the main focus was to achieve all the basic features and functionality for such a system, where further work can be applied setting the development in several different and more specialized directions.

As a final conclusion, and taking consideration of all the above, we can assume that the task of the audio to score alignment has been implemented successfully, including the ability to detect performer mistakes without losing track of the evolution of the performance, as well as working without being affected by significant amounts of

background sounds, either they are accompanying instruments or just other noise.

References

[1] Orio, Nicola, 2003, *Music retrieval: A tutorial and a review*, Foundations and Trends in Information Retrieval, p. 1-90

[2] Berger, Dan, 2003, *A music data mining and retrieval primer*

[3] Downie, J. Stephen, 2003, *Music Information Retrieval*, Annual Review of Information Science and Technology Vol. 37, p. 295-340

[4] *Enigma Transportable Format (ETF)*, http://www.music-notation.info/en/formats/ETF.html

[5] *GUIDO Music Notation Format Page*, http://science.jkilian.de/salieri/GUIDO/index.html

[6] *LilyPond - Music notation for eveyrone*, http://lilypond.org/

[7] *MuseData*, http://www.musedata.org/

[8] *MusicXML for Exchanging Digital Sheet Music*, http://www.musicxml.com/

[9] *The Moving Picture Experts Group webpage*, http://mpeg.chiariglione.org/

[10] *FLAC - Free Lossless Audio Codec*, https://xiph.org/flac/

[11] *Monkey's Audio - a fast and powerful lossless audio compressor*, http://www.monkeysaudio.com/

[12] Muller, Meinard, 2011, *New developments in music information retrieval*, AES, p. 1

[13] Casey, Michael A, 2008, *Content-Based Music Information Retrieval: Current Directions and Future Challenges*, Proceedings of the IEEE, p. 668-696

[14] *MIREX*, http://www.music-ir.org/mirex/wiki/MIREX_HOME

[15] Schwarz, Diemo, 2003, *Data-drivern concatenative sound synthesis*, UPMC - Paris 6

[16] Gomas, Antoine, 2007, *Educational software using audio to score alignment*, University of Birmingham

[17] Aceituno, Jonathan, 2011, *Real-time score following techniques*

[18] Cont, Arshia, 2003, *Improvement of observation modeling for score following*, UPMC

- Paris 6
[19] Vercoe, Barry, 1984, *The synthetic performer in the context of live performance*, Proceedings of the ICMC, p. 199-200
[20] Vercoe, Barry, Puckette, Miller, 1985, *Synthetic rehearsal: Training the synthetic performer*, Proceedings of the ICMC, p. 275-278
[21] Dannenberg, R., 1984, *An On-Line Algorithm for Real-Time Accompaniment*, Proceedings of the ICMC, p. 193-198
[22] Dannenberg, Roger and Mont-Reynaud, 1987, *Following an Improvisation in Real Time*, Proceedings of the ICMC, p. 241-248
[23] Dannenberg, Roger and Mukaino, 1988, *New Techniques for Enhanced Quality of Computer Accompaniment*, Proceedings of the ICMC, p. 243-249
[24] Puckette, Miller, 1990, *EXPLODE: A User Interface for Sequencing and Score Following*, Proceedings of the ICMC, p. 259-261
[25] Puckette, Miller and Cort Lippe, 1992, *Score following in practice*, Proceedings of the ICMC, p. 182-185
[26] Puckette, Miller, 1995, *Score following using the sung voice*, Proceedings of the ICMC, p. 199-200
[27] Braid, Bridget, Donald Blevins & Noel Zahler, 1990, *The Artificially Intelligent Computer Performer: The Second Generation*, Interface, Journal of New Music Research, Vol. 19, p. 197-204
[28] Orio, Nicola and Diemo Schwarz, 2001, *Alignment of Monophonic and Polyphonic Music to a Score*, Proceedings of the ICMC, p. 155-158
[29] Dixon, Simon, 2005, *An on-line time warping algorithm for tracking musical performances*, Proceedings of the 19th international joint conference on Artificial intelligence, p. 1727-1728
[30] Grubb, Lorin and Roger Dannenberg, 1997, *System and method for stochastic score following*, US Patent No.5913259, p.
[31] Grubb, Lorin and Roger Dannenberg, 1997, *Stochastic Method of Tracking a Vocal Performer*, Proceedings of the ICMC, p. 301-308

[32] Pardo, Bryan and William Birmingham, 2001, *Following a musical performance from a partially specified score*, IEEE Multimedia TechnologyApplications Conference, p.

[33] Pardo, Bryan and William Birmingham, 2002, *Improved Score Following for Acoustic Performances*

[34] Raphael, C., 1999, *Automatic segmentation of acoustic musical signals using Hidden Markov Models*, p. 360–370

[35] Cano, Perdo, Alex Loscos and Jordi Bonada, 1999, *Score-Performance Matching using HMMs*, Proceedings of the ICMC, p. 441-444

[36] Jordanous, Anna and Alan Smaill, 2008, *Articially intelligent accompaniment using hidden markov models to model musical structure*, Fourth Conference on Interdisciplinary Musicology

[37] Raphael, Christopher, 2001, *A Bayesian network for Real-time Musical Accompaniment*, Neural Information Processing Systems Vol. 14

[38] Cont, Arshia, 2007, *Modeling musical anticipation: From the time of music to the music of time*, UPMC

[39] Cont, Arshia, 2010, *A coupled duration-focused architecture for real time music to score alignment*, IEEE Transaction on Pattern Analysis and Machine Intelligence Vol. 32, p. 974-987

[40] *Antescofo videos*, http://repmus.ircam.fr/antescofo

[41] *ComParser*, http://kmt.hku.nl/~pieter/SOFT/CMP/doc/cmp.html

[42] *About Tonara*, http://tonara.com/about/

[43] *Miso Media Inc.*, https://www.misomedia.com/apps

[44] *PROBADO*, http://www.probado.de/

[45] *SampleSumo*, https://www.samplesumo.com/

[46] Bishop, Christopher M., 2006, *Pattern Recognition & Machine Learning*, Springer

[47] Mitchell, Tom M., 1997, *Machine Learning*, McGraw Hill

[48] Nilsson, N. J., 1998, *Introduction to Machine Learning*

[49] Theodoridis, S. and Koutroumbas, K., 2003, *Pattern Recognition*, Elsevier

[50] McCullagh, P., 2002, *What is a Statistical Model?*, The Annals of Statistics, p. 1225–

1310

[51] Davison, A. C., 2003, *Statistical models*, Cambridge University Press

[52] Dempster, A.P., N. M. Laird, and D. B. Rubin, 1977, *Maximum Likelihood from Incomplete Data via the EM Algorithm*, Journal of Royal Statistical Society Series B, Vol. 39, No. 1, p. 1-38

[53] Bilmes, Jeff A., 1998, *A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models*, ICSI

[54] Vapnik, V. N., 1998, *Statistical Learning Theory*, John Wiley & Sons

[55] Ng, A. Y. and Jordan, 2002, *On Discriminative vs Generative Classifiers*, Advances in Neural Information Processing Systems

[56] Murphy, K., 1998, *A Brief Introduction to Graphical Models and Bayesian Networks*

[57] Stengel , M., 2003, *Introduction to Graphical Models, HMM and Bayesian Networks*, Toyohashi University of Technology

[58] Koller, D. and Friedman, 2009, *Probabilistic Graphical Models - Principles and Techniques*, MIT Press

[59] Jordan, Michael I., 1999, *Learning in Graphical Models*, MIT Press

[60] Heckerman, D., 1995, *A Tutorial on Learning With Bayesian Networks*, Microsoft Research

[61] Ghahramani , Zoubin, 1997, *Learning Dynamic Bayesian Networks*, Adaptive Processing of Sequences and Data Structures, p. 168-197

[62] Murphy, Kevin P., 2002, *Dynamic Bayesian Networks, Representation, Inference and Learning*, University of California, Berkeley

[63] Rabiner, L.R., 1989, *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition*, Proceedings of the IEEE, Vol. 77, p. 257-286

[64] *Algorithms – Hidden Markov Models*, http://www.shokhirev.com/nikolai/abc/alg/hmm/hmm.html

[65] Petrushin, Valery, 2000, *Hidden Markov Models: Fundamentals and Applications. Part 2: Discrete and Continuous Hidden Markov Models*, Online Symposium for Electronics Engineers

[66] Cho, Taemin and Juan Bello, 2009, *Real-Time Implementation of HMM-Based Chord Estimation in Musical Audio*, International Computer Music Conference, p. 117 - 120

[67] Cont Arshia, Schwartz Diemo & Schnell Norbert, 2005, *From Boulez to Ballads: Training IRCAM's Score Follower*, Proceedings of International Computer Music Conference (ICMC)

[68] *MIDIToolbox*, https://www.jyu.fi/hum/laitokset/musiikki/en/research/coe/materials/miditoolbox/

[69] *Weka2Matlab*, http://umslt.googlecode.com/svn/trunk/MatlabMusicSeg/matlab/matlab2weka/weka2matlab.m

[70] *Aqua Phoenix Matlab Lecture*, http://www.aquaphoenix.com/lecture/matlab10/page2.html

[70] de Cheveigne A. and H. Kawahara, 2002, *YIN, A Fundamental Frequency Estimator for Speech and Music*, Journal of the Acoustical Society of America, vol. 111, p. 1917–1930

[71] *YIN*, http://www.ircam.fr/pcm/cheveign/sw/yin.zip

[73] Cont, Arshia, Diemo Schwarz, Norbert Schnell and Christopher Raphael, 2007, *Evaluation of Real-Time Audio-to-Score Alignment*, International Symposium on Music Information Retrieval (ISMIR)

[73] Orio, Nicola, Serge Lemouton and Diemo Schwarz, 2003, *Score Following: State of the Art and New Developments*, Proceedings of the International Conference on New Interfaces for Musical Expression, p. 36-41