Faculdade de Engenharia da Universidade do Porto

FEUP

Towards an Interactive Framework for Robot Dancing Applications

João Manuel Lobato Dias da Silva Oliveira

VERSÃO PROVISÓRIA

Dissertação/Relatório de Projecto realizada(o) no âmbito do
Mestrado Integrado em Engenharia Electrotécnica e de Computadores
Major <Telecomunicações>

Orientadores:
Prof. Dr. Luis Paulo Reis
Prof. Dr. Fabien Gouyon

Junho de 2008

Resumo

O processo de ouvir música consiste em transformações contínuas, em que diferentes agentes especializados monitorizam o sinal musical de entrada produzindo sinais variados à saída sobre a forma de respostas comportamentais. Estas respostas podem ser observáveis, sob a forma de movimentos de dança ou através do mero bater do pé; emocionais, através de reacções de tensão e relaxamento, ou totalmente intrínsecas, demonstradas no reconhecimento e classificação de instrumentos, estilos, e compositores. Estudos psicológicos e sociológicos sugerem que a corporização do ritmo, através de movimentos de dança, é um princípio organizacional elementar e predominante em interações sociais humanas.

Nesta dissertação apresentamos o desenvolvimento dum sistema robótico que utiliza um robô humanoide, baseado no Lego Minstorms NXT, na tentativa de simular a percepção rítmica humana a partir de sinais audio complexos, e o seu comportamento reactivo sob a forma de dança. Para tal desenvolvemos um sistema decomposto em três modulos: *Análise Musical, Controlo Humano, e Controlo Robótico*, que são processados em paralelo numa arquitectura "multithreading", de modo a induzir uma reacção de dança robótica sob uma abordagem reactiva-comportamental.

O Módulo de Análise Musical implementa uma precepção rítmica de baixo-nível baseada na função de detecção de "onsets" do Marsyas (um software open-source para processamento de audio), utilizando "peak-picking" e um "thresholding" adaptativo. O Módulo de Controlo Humano consiste numa interface gráfica interactiva para a definição de dança e dos parâmetros audio, que garante o papel determinístico do utilizador no comportamento do sistema, assegurando o seu dinamismo. O Módulo de Controlo Robótico incita o robô a reagir, em tempo-real, à recepção de eventos rítmicos e sensoriais (nomeadamente a cor do chão e a proximidade a um qualquer obstáculo), encorporando a dança previamente estabelecida.

A dança resultante vai variando dum modo aparentemente autónomo, pela gesticulação de movimentos diversificados acoplados ao rítmo musical, que variam em consonância com a cor do chão do ambiente de dança, sem conhecimento musical prévio. Um comportamento dinâmico e interactivo, em compromisso com o sincronismo, assegura uma relação interessante, a longo termo, entre um humano e um agente artificial.

Abstract

The process of music listening is one of continuous transformation, in which different agencies of musical expertise monitor the input signal and produce various output signals in the forms of behavioural responses. These responses may be overt, through dance movements or mere foot-tapping behaviour; emotional, as in tension and relaxation reactions; or entirely immanent, though recognition and classification of instruments, genres, and composers. Psychological and sociological research suggests that the embodiment of rhythm, through dance movements, is an elementary and pervasive organizing principle for social human interaction.

In this dissertation we present the development of a robotic system using a humanoid robot, based on the Lego Mindstorms NXT, which tries to simulate the human rhythmic perception from complex audio signals and its reactive behaviour in the form of dance. To do so we developed a framework decomposed in three modules: Music Analysis, Human Control, and Robot Control, which are parallelly processed, through a multithreading architecture, to induce a robotic dance performance in a reactive behavioural-based approach.

The Music Analysis Module performs a low-level rhythmic perception based on the Marsyas (an open source software framework for audio processing) onset detection function, with peak picking and adaptive thresholding. The Human Control Module consists on an interactive graphical interface for dance and audio parameters definitions, which grants the user deterministic role in the behaviour of this system, while assuring its dynamic reaction. The Robot Control Module incites the robot to react, in real-time, to the received rhythm and sensorial events (namely the colour stepped on the floor or the proximity to some kind of obstacle), embodying the previously defined dance.

The resulting dance alternates in a seemingly autonomous manner between a diversity of motion styles coupled to the musical rhythm, and varying in consonance with the colour stepped on the dance environment, without any previous knowledge of music. A dynamic and interactive behaviour, in compromise with synchronism, assures an interesting relationship between a human and an artificial agent in the long-term.

Acknowledgments

First of all, I would like to express my gratitude to my supervisors, Prof. Dr. Fabien Gouyon and Prof. Dr. Luis Paulo Reis, for all their support and guidance in the development of this project throughout this semester, helping me whenever needed and making all this work possible. I would also like to thank Luis Gustavo Martins, from INESC Porto, for his explanations on Marsyas operation and functionality.

Finally, but not least, I wish to thank all my family for their lifetime support, Ana for her love and patience, and all my friends for their friendship and encouragement throughout the years, in especial Gustavo Meirinhos and Pedro Vieira for helping me with the robot design and construction, and Pedro Allegro for his companionship and assistance.

Contents

List of Figures

List of Tables

Acronym List

CD	Complex Difference
CM	Complex Mixtures
DFT	Discrete Fourier Transform
FEUP	Faculdade de Engenharia da Universidade do Porto
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
FP	False Positives
GUI	Graphical User Interface
HFC	High-Frequency Content
ICA	Independent Component Analysis
INESC	Institute for Systems and Computer Engineering
ISMIR	International Symposium on Music Information Retrieval
LIACC	Laboratory of Artificial Intelligence and Computer Science
M2K	Music-to-Knowledge
MIREX	Music Information Retrieval Evaluation eXchange
NL	Negative Log-Likelihood
NP	Non-Pitched Percussion
NWPD	Normalised Weighted Phase Deviation
PD	Phase Deviation
PN	Pitched Non-Percussion
PP	Peak-Picking
PP	Pitched Percussion
RCD	Rectified Complex Domain
ROC	Receiver Operating Characteristic
SF	Spectral Flux
STFT	Short Time Fourier Transform

SVM	Support Vector Machine
TFR	Time-Frequency and Time-Scale Methods
TP	True Positives
WAV	WAVEform Audio Format
WPD	Weighted Phase Deviation
WRM	Wavelet Regularity Modulus

Chapter 1

Introduction

More and more AI researchers are trying to make robots dance as they listen to music. And as the ideas and technologies develop, it's clear that dancing robots can be serious indeed. Recent generations of robots ever more resemble humans in shape and articulatory capacities. This progress has motivated researchers to design interactive dancing robots that can mimic the complexity and style of human choreographic dancing, and that even cooperate with musicians.

Musical robots are increasingly present in multidisciplinary entertainment areas, even interacting with professional musicians, as when the ASIMO robot conducted the Detroit Symphony Orchestra in a performance of Mitch Leigh's "The Impossible Dream" from the Man from La Mancha (on May 13th, 2008)[1]. They have even inspired the creation of worldwide robotic dancing contests, as RoboDance (one of RoboCup's competitions) where school teams, formed by children aged eight to nineteen, put their robots in action, performing dance to music in a display that emphasize creativity of costumes and movement.

These public musical robotic applications lack however in perception, presenting mainly pre-programmed deaf robots with few human-adaptive behaviours. This is where we focused our efforts by designing an interactive framework for robot dancing applications based on automatic music signal analysis.

This chapter aims to contextualize our research as a robot dancing framework, presenting our motivations toward robot dance through rhythmic perception embodiment, and our methodology, in consideration to the used tools and chosen architecture. As starting points to this dissertation we introduce some basic thoughts and definitions followed by the main objectives involved.

---

[1] See http://www.autoblog.com/tag/asimo+orchestra/.

1.1 Motivations

The motivations which supported the topic of this dissertation are induced by a convergence of several factors, represented by the steps involved in the embodiment of musical rhythm, from a computational (machine) point of view.

To better define this topic and in order to motivate the reader to follow our approach through this report, in this section we summarize the preliminary overview, presenting the scientific and technological contextualization and introducing the hosting institutions and the author's trajectory behind the project. As one last point of interest we introduce some of the thoughts behalf the consummation of our vision.

1.1.1 Scope

More and more "Dancing Robots" and "Human-Robot Musical Interaction" are becoming very common terms. In an increasing number of research labs around the world (especially in Japan), researchers follow a quest to find the perfect solution to achieve a rhythmic perceptive and interactive dancing robot.

Robot Dancing is an interdisciplinary entertainment area which increasingly enthusiasms fanciers worldwide, while assisting people's life, being further enjoyable. As the term points out, it embraces several disciplines: Musical Rhythm, Computational Rhythmic Perception, Autonomous Robotic Systems, Synchronous Reactive Behaviour, and Dance Embodiment.

Human-Robot Interaction can be achieved through dance movements tuned to musical rhythm. This interaction depends on two essential factors: Entrainment through synchronism and Dynamism through flexibility.

This dissertation addresses both of these topics towards the development of an interactive (flexible) framework for robot dancing applications.

1.1.2 Research at LIACC and INESC Porto

This research was carried out at LIACC, under the supervising of Prof. Dr. Luis Paulo Reis, in association with INESC Porto, under the supervising of Prof. Dr. Fabien Gouyon.

LIACC (Laboratory of Artificial Intelligence and Computer Science of the University of Porto) was created in 1988 to promote the collaboration of researchers that were separately working in the fields of Computer Science and Artificial Intelligence in different Faculties.

LIACC aims at helping to solve general computer science problems, from security to software reliability. These hard, real-world problems can only be solved in the long term by combining the power of formal methods with more technology-oriented approaches and were used as a frame of reference in defining the LIACC short-term goals.

Since June 2007 the LIACC activities are organized around four research groups: Advanced Programming Systems, Distributed Artificial Intelligence and Robotics, Formal Models of Computation, and Language, Complexity and Cryptography.

INESC (Institute for Systems and Computer Engineering) of Porto is an institution created to act as an interface between the academic world, the world of industry and services, as well as the public administration, in the framework of the Information Technologies, Telecommunications and Electronics. Its activities range from research and development, to technology transfer, consulting and advanced training. Its main research areas of interest are Telecommunications and Multimedia, Power Systems, Manufacturing Systems Engineering, Information and Communication Systems, and Optoelectronics.

Among the various areas of expertise within these institutions, our work was specially related to Autonomous Robotic Systems, covered by Distributed Artificial Intelligence and Robotics, in LIACC; and Automatic Rhythm Description, covered by Telecommunications and Multimedia, in INESC.

1.1.3 Personal Trajectory

This dissertation reports all the research and work developed in the last five months, as part of the final course of the 5th year of the Integrated Masters in Electrical and Computer Engineering at FEUP (Engineering Faculty of the University of Porto). It represents the final project in an academic five years' cycle that comes to an end. A cycle founded on the cultural accumulation of knowledge through which I cultivated the skills needed to the development of such work.

Everything else came, since early years, from the natural dance performance in complex human interactive environments (e.g. discos and festivals). A performance that incited curiosity on all the processes involved in the rhythmic perception of music and its subsequent embodiment in the form of dance.

1.1.4 Thoughts to Consummate a Vision

"The goal of AI has been characterized as both the construction of useful intelligent systems and the understanding of human intelligence…trying to build truly intelligent autonomous robots." [1 p. 1].

"Intelligent systems are decomposed into independent and parallel activity producers which all interface directly to the world through perception and action…in a behavioural reactive manner." [2 p. 1].

"The intelligence does not come from a set of rules that describe "how music works." Rather, the intelligence comes from continuous transformations of the signal, and the way

that these transformations are manifested as musical behaviours (…) There is no function in which it is decided what is the "right thing to do" as the signal is being processed." [3 p. 75].

"The computational model and the psychoacoustic experiment play overlapping and complementary roles in advancing our knowledge about the world." [3 p. 66].

"A computational theory of music cognition indeed, any computational theory of perception or cognition in any modality must be validated through comparison to meaningful experimental data. (…) The model builder might have implemented a large lookup table and listed all the stimulus patterns and the appropriate responses." [4 p. 5].

Resembling human perception and behaviour (as human intelligent capabilities), the research program we envisioned aims at developing a rhythmic perceptual robotic system that evaluates the relevance of embodied (behavioural-based) cognitive science. In the system, a humanoid robot reacts, autonomously, in real-time, to complex (real) auditory stimuli in the forms of behavioural responses, through dance movements. In this context we developed a flexible framework which addresses the issue of robot dancing, and grants human control through full but flexible dance definitions.

The experiments were executed in a real world environment. The results were compared to the meaningful experimental data of human dance performance.

1.2 Dissertation Aims and Outline

"*The fundamental slicing up of an intelligent system is in the orthogonal direction dividing it into activity producing subsystems. Each activity (pattern of interactions with the world), or behaviour producing system individually connects sensing to action... The advantage of this approach is that it gives an incremental path from very simple systems to complex autonomous intelligent systems*." [2].

Approaching this incremental intelligence we decomposed this dissertation's objectives in a series of "activities", which we'll name modules, and that were developed subsequently:

- **Music Analysis Module:** Module responsible for the musical rhythm analysis, through a low-level perception model based on onset detection.

- **Robot Control Module:** Module responsible for robot control, by inciting its reaction in synchrony to the former detected rhythm.

- **Human Control Module:** Module responsible for the user interface (GUI) through which the user has a deterministic role by defining the onset function parameters and all the dance movements to be performed by the robot.

The interconnection of these three modules constituted our framework, which aimed to control a humanoid robot to perform seemingly autonomous dance movements in synchrony to musical rhythm, without former knowledge of the music.

Further objectives are analyzed in future work (see section 5.1).

1.3 Methodology and Tools

The realization of the presented objectives was only possible due to a conjunction of software applications, which worked together to develop, experiment and ultimately control the hardware, formed by the robot. Following we present the main used tools and our methodology through their application.

1.3.1 Marsyas

Marsyas (Music Analysis, Retrieval and Synthesis for Audio Signals)[2] is an open source software framework for rapid prototyping and experimentation with audio analysis and synthesis with specific emphasis to music signals and Music Information Retrieval. Its basic goal is to provide a general, extensible and flexible architecture that allows easy experimentation with algorithms and provides fast performance that is useful in developing real time audio analysis and synthesis tools. A variety of existing building blocks that form the basis of most published algorithms in Computer Audition are already available as part of the framework and extending the framework with new components/building blocks is straightforward.

It has been designed and written by George Tzanetakis with help from students and researchers from around the world. Marsyas has been used for a variety of projects in both academia and industry.

Our Music Analysis Module was essentially founded on Marsyas (v0.23), combining the required blocks to develop an onset detection function.

1.3.2 Microsoft Visual Studio – Visual C++

Microsoft Visual Studio[3] is the main Integrated Development Environment (IDE) from Microsoft. It can be used to develop console and Graphical user interface applications along

[2] Consult http://marsyas.sness.net/.
[3] Check Microsoft Visual Studio Developer Center at http://msdn.microsoft.com/en-us/vstudio/default.aspx

with Windows Forms applications, web sites, web applications, and web services in both native code as well as managed code for all platforms supported by Microsoft.

Visual C++ is Microsoft's implementation of the C and C++ compiler and associated languages services and specific tools for integration with the Visual Studio IDE. It can compile either in C mode or C++ mode.

Visual Studio (2008) was the foremost used software of our work. Through this IDE we've integrated all the existing algorithms, namely Marsyas onset detection function and NXT Remote API, and programmed all the remnant code to develop our framework. Our framework's user interface (GUI) was also designed through this IDE with Microsoft Foundation Class (MFC) library.

1.3.3 MATLAB

MATLAB[4] is a numerical computing environment and programming language. Created by The MathWorks, MATLAB allows easy matrix manipulation, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs in other languages.

In this project MATLAB (R2008a), acting as a simulative output (through plotting), was responsible for all the experimentation made in order to properly "calibrate" our *Music Analysis Module*.

1.3.4 Lego Mindstorms NXT

Lego Mindstorms NXT[5] is a programmable robotic kit designed by Lego (see Figure 1). It is composed by a brick-shaped computer, named NXT brick, containing a 32-bits microprocessor, flash and RAM memory, a 4 MHz 8-bit microcontroller and a 100x64 LCD monitor. This brick supports up to four sensorial inputs and can control up to three servo-motors. It also has an interface displayed by the LCD and controlled with its four buttons, and a 16 kHz speaker.

Lego NXT supports USB 2.0 connection to PC and presents a Bluetooth wireless communication system, for remote control and data exchange. It offers many sensor capabilities through its ad-hoc sensors. In the scope of this project we provided our robot with a colour sensor, to detect and distinguish visible colours, and an ultrasonic sensor, capable of obstacle detection, retrieving the robot's distance to it in inches or centimetres.

Based on this technology we built a humanoid-like robot using two NXT bricks that controls six servo motors (one for each leg and each arm, one for a rotating hip and one for the head) and the two referred sensors.

---

[4] Check MATLAB official web site at http://www.mathworks.com/.
[5] For more information consult http://mindstorms.lego.com/eng/default.aspx.

Figure 1 - Lego NXT brick and some of its sensors and servo-motors.

1.3.5 NXT Remote API

The NXT Remote API[6] is a C++ library, designed by Anders Søborg, to remotely control the LEGO NXT brick over Bluetooth, using any C++ compiler on a Windows OS (the library can be also used with Pocket PCs running Windows Mobile using MS Visual Embedded C++). This API is decomposed in nine classes, which make it possible to:

- Open and close Bluetooth connections with multiple NXT units;
- Control the motors;
- Send and receive messages from the NXT;
- Read sensor values - both mode-dependent and raw;
- Set Brick name, get battery level, read firmware version etc;
- Control the NXT speaker;
- Play sound files;
- Start and stop on-brick programs;
- Use compass and sonar sensors;
- Communicate with I2C sensors;
- Direct commands for the PCF8591 A/D converter;
- Direct commands for the PCF8574 I/O Chip.

This library (v0.3) was embedded in our *Robot Control Module* to remotely receive sensing (ultrasonic and colour sensors) information from the robot and send motor (actuators) outputs correspondent to the embodiment of proper dance movements.

1.3.6 Methodology

In this section we present our methodology as the interconnection between all the present tools, joint together to achieve an ultimate goal.

---

[6] For more information and download consult http://www.norgesgade14.dk/index.php.

To clarify our method we present it through a representative diagram (see Figure 2).

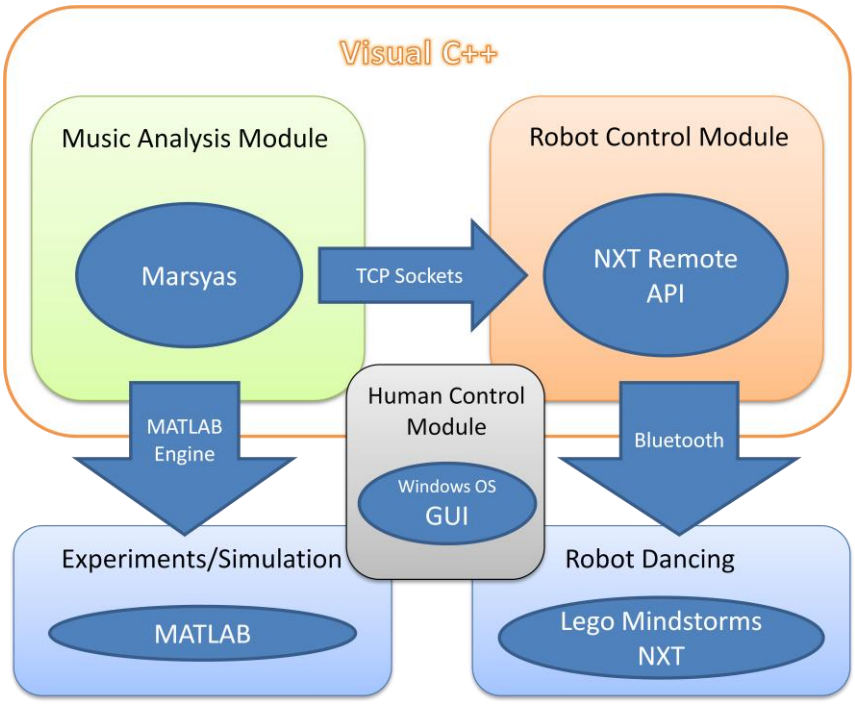


Figure 2 - Our methodology through a conjunction of tools.

1.4 Dissertation Structure

This dissertation is organized in five chapters, the first of which is this introduction to the thesis motivation, aims, outline, methodology and tools.

In the following chapter (*Chapter 2*) we present the background work done on the areas of onset detection and dancing interactive robots, presenting all the first relevant definitions to a complete understanding on these topics, and a state-of-the-art overview.

In *Chapter 3* we define our project's approach, with focus on the designed system architecture.

In *Chapter 4* we present the main experiments and the achieved results, analyzing them in comparison to meaningful data (human dance) on a real world dance environment.

Finally, *Chapter 5* concludes this dissertation by summarizing general conclusions and proposing a path for future work.

Chapter 2

State Of The Art

This section presents the state of art related to all topics of interest to the development of this thesis.

It is divided in two main distinct subsections: "*Audio Onset Detection*" and "*Dancing and Rhythmic Interactive Robots*". A third section presents "*A Turning Point*", defining our approach in contrast to the former reviewed.

2.1 Audio Onset Detection

Many music signal analysis applications require the accurate detection of onsets of musical tones, and it is not surprising that several different methods have been proposed for performing onset detection. At first sight, onset detection is a well-defined task: the aim is to find the starting time of each musical note (where a musical note is not restricted to those having a clear pitch or harmonic partials), [5]. However, in polyphonic music, where nominally simultaneous notes (chords) might be spread over tens of milliseconds, the definition of onsets starts to become blurred.

In order to clarify the concept of onset time and introduce the process of its detection in any application, based on [6] we define the concepts of *transients*, *onsets* and *attacks*. Due to the importance in distinguish the similarities and differences between these key concepts as the categorization of all related approaches, following we present some fundamental considerations.

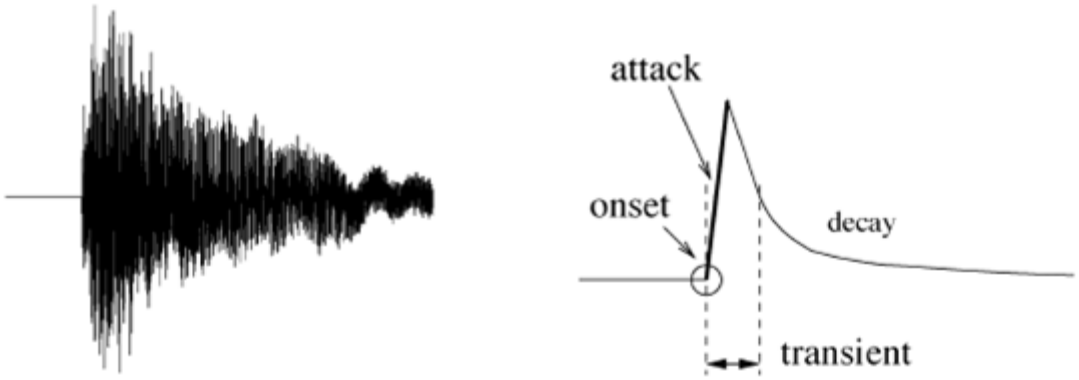


Figure 3 - "Attack", "transient", "decay", and "onset" in the ideal case of a single note, [6].

Figure 3 shows, in the simple case of an isolated note, how one could differentiate these notions. The attack of the note is the time interval during which the amplitude envelope increases. Transients are short intervals during which the signal evolves quickly in a relatively unpredictable way. In acoustics, the transient often corresponds to the period during which the excitation is applied and then damped, leaving only the slow decay at the resonance frequencies of the body. This consideration shall take into account a time resolution of 10 ms due to the assumption that human ears cannot distinguish between two transients occurring in time intervals less than this. The onset of the note is a single instant chosen to mark the temporally extended transient. It will mostly coincide with the start of the transient, or the earliest time at which the transient can be reliably detected.

Previously, in *2.1.1 Basic Definitions and General Scheme of Onset Detection Algorithms* we review the basic concepts in onset detection and introduce a general categorization of onset detection algorithms. In *2.1.2 Audio Onset Detection Functions: A State-Of-The-Art Review*, we present a review on some of the research made on this area. In *2.1.3 Results Comparison* we conclude the state-of-the-art in this research area by presenting a result comparison upon some of the most prominent recent proposed models.

2.1.1 Basic Definitions and General Scheme of Onset Detection Algorithms

As observable in Figure 4, onset detection algorithms are normally split into three components: the pre-processing of the original audio signal to improve the performance of subsequent stages; the detection function, a signal representing the changing state of a musical signal, typically at a lower sampling rate; and a second stage of peak picking within the detection function to find onset times. However, frequently the pre-processing stage is ignored in order to simplify the algorithm.

Following Bello *et al.* work I decomposed this overview in the three subsequent onset detection stages.

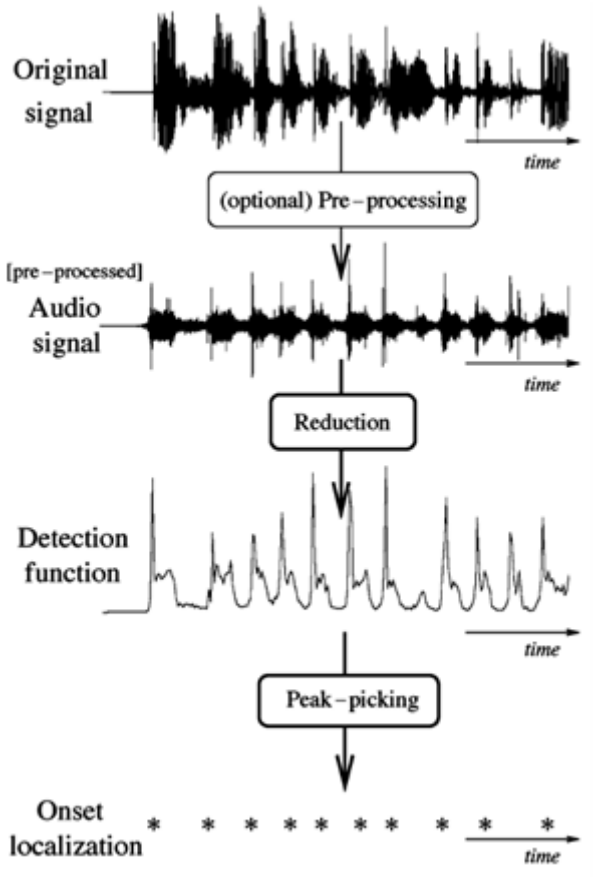


Figure 4 - Flowchart of a standard onset detection algorithm, [6].

2.1.1.1 Onset Detection: Pre-Processing

The concept of pre-processing implies the transformation of the waveform to isolate different frequency bands in order to accentuate or attenuate various aspects of the signal according to their relevance to the task in hand. It is an optional step that derives its relevance from the process or processes to be subsequently performed [6].

Attending to the literature and [6], this review is decomposed in two processes that appear to be of particular relevance to onset detection schemes: *Multiple Bands* and *Transient/Steady-State Separation*.

The pre-processing based on *multi bands* is normally used to satisfy the needs of specific applications that require detection in individual sub-bands to complement global estimates, as a way of increasing the robustness of a given onset detection method. This process involves the use of filter banks' conjugations, which might be fed into comb-filter resonators in order to estimate the tempo of the signal.

The process of *transient/steady-state* separation is usually associated with the modelling of music signals. One can refer the sinusoidal models, such as "additive synthesis", which represent an audio signal as a sum of sinusoids with slowly varying parameters, and the spectral modelling synthesis (SMS), which considers the residual of the synthesis method as a Gaussian white noise filtered with a slowly varying low-order filter. One should also refer the

transient modelling synthesis (an extension of SMS), and the discrete cosine transform of the residual (in a pseudo-temporal domain) in which transient signals are analyzed by a sinusoidal analysis/synthesis similar to SMS. Due to the irrelevance of this pre-processing scheme in the scope of this thesis, a state-of-the art review shall not be presented.

2.1.1.2 Onset Detection: Reduction

In this context, the concept of reduction refers to the process of transforming the audio signal into a highly sub-sampled detection function which manifests the occurrence of transients in the original signal, and enhances the bands (obtained with pre-processing) such that note onsets are more salient. This consists in the onset scheme basis promoting a wide class of onset detection methods, which will be the focus of most of our review (see 2.1.2).

Based on [6], [5], [7], we decomposed this reduction analysis in six different method types, according to their basis (on signal features or statistics): *Temporal Methods*, *Energy-Based* (*Spectral Weighting*) *Methods*, *Phase-Based Methods*, *Complex Methods*, *Time-Frequency and Time-Scale Methods (TFR)*, and *Statistical Methods*.

- **Temporal Methods:** When observing the temporal evolution of simple musical signals, it is noticeable that the occurrence of an onset is usually accompanied by an increase of the signal's amplitude [6]. Early temporal methods of onset detection used a detection function which follows the amplitude envelope of the signal. Such an "envelope follower" can be constructed by low-pass filtering the signal:

$$E_0 = \frac{1}{N}\sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1}|x(n+m)|\omega(m) \tag{1.1}$$

  where *ω(m)* is an *N*-point window or smoothing kernel, centred at *m=0*. A variation on this is to follow the local energy, rather than the amplitude, by squaring, instead of rectifying, each sample:

$$E(n) = \frac{1}{N}\sum_{m=-\frac{N}{2}}^{\frac{N}{2}-1}[x(n+m)]^2\omega(m) \tag{1.2}$$

  This reduced signal in its raw form is not usually suitable for reliable onset detection by peak picking. A further refinement, included in a number of standard onset detection algorithms, is to work with the time derivative of the energy so that sudden rises in energy are transformed into narrow peaks in the derivative. The energy and its derivative are commonly used in combination with pre-processing, both with filter-banks and transient/steady-state separation.

- **Energy-Based (Spectral Weighting) Methods:** A new note will always lead to an increase in signal energy. In the case of strong percussive note attacks, such as drums, this increase in energy will be very sharp. For this reason, energy has proved to be a useful, straightforward, and efficient metric by which to detect percussive transients,

and therefore certain types of note onset. Considering the L_2 norm squared energy of a frame of the signal, ***x(m)***:

$$E(m) = \sum_{n=(m-1)h}^{mh} |x(n)|^2 \quad (2.1)$$

where *h* is the hop size, *m* the hop number and *n* is the integration variable. Taking the first derivative of *E(m)* it produces a detection function from which peaks may be picked to find onset locations. This is one of the simplest approaches to note onset detection. This idea can be extended to consider frames of an FFT (Fast Fourier Transform). Considering that all of these methods make use of a time-frequency representation of the signal based on a short time Fourier transform (STFT) using a Hamming window *ω(m)*, and calculated at a frame rate of 100 Hz, if ***x(mh)*** is a time-domain signal then:

$$STFT = X(n,k) = \sum_{m=-\frac{N}{2}}^{\frac{N}{2}-1} x(hn+m)\omega(m)e^{\frac{2j\pi mk}{N}} \quad (2.2)$$

where *k = 0, 1,…, N-1* is the frequency bin index, and n the frame number. It follows that the *amplitude difference* (*spectral flux*) measures the change in magnitude in each frequency bin, and if this is restricted to the positive changes and summed across all frequency bins, it gives the onset function SF, [8]:

$$SF(n) = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} H(|X(n,k)| - |X(n-1,k)|) \quad (2.3)$$

where $H(x) = (x + |x|)/2$ is the half-wave rectifier function.

- **Phase-Based Methods:** Intuitively, Fourier analysis proposes that a signal can be represented by a group of sinusoidal oscillators with time-varying amplitudes, frequencies and phases. During the steady-state part of the signal these oscillators will tend to have stable amplitudes and frequencies. Therefore, the phase of the k^{th} oscillator at a given time (frame) *n* could be easily predicted according to:

$$\varphi_k(n-1) - \varphi_k(n-2) = \varphi_k(n) - \varphi_k(n-1) \quad (3.1)$$

where the φ operator denotes phase unwrapping (cf. Figure 5). This implies that the actual *phase deviation* (PD) between the target and the real phase values is given by the term [9]:

$$PD = d_\varphi = princarg\left[\Delta\varphi_k(n)\right] \quad (3.2)$$

where, $$\Delta\varphi_k(n) = \varphi_k(n) - 2\varphi_k(n-1) + \varphi_k(n-2) \cong 0\ , \quad (3.3)$$

$\varphi_k(n)$ is the k^{th} frequency bin of the *nth* time frame from the STFT of the audio signal. The operator *princarg* maps the angle to the [−π, π] range, and d_φ will tend to zero if the phase value is accurately predicted and will deviate from zero otherwise (the case for most oscillators during attack transients). By measuring the spread of the distribution an accurate onset detection function can be constructed.

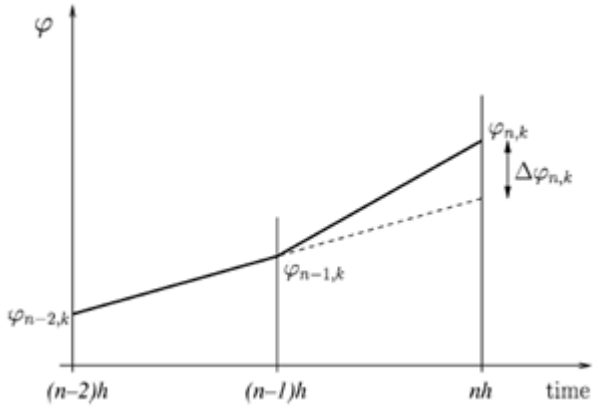


Figure 5 – Phase diagram showing instantaneous frequencies as phase derivative over adjacent frames, [6]. For a stationary sinusoid this should stay constant (dotted line).

- **Complex Methods:** There are a number of reasons that justify combining phase and energy information for onset detection: while energy-based approaches favour strong percussive onsets, phase-based approaches emphasize soft, "tonal" onsets. The two methods are more reliable at opposite ends of the frequency axis. Let's consider a simpler measure of the spread of the distribution calculated as the mean absolute phase deviation:

$$\zeta_p(n) = \frac{1}{N}\sum_{k=1}^{N}\left|\Delta\varphi_k(n)\right|. \tag{4.1}$$

This method, although showing some improvement for complex signals, is susceptible to phase distortion and to noise introduced by the phases of components with no significant energy. As an alternative we introduce an approach that works with Fourier coefficients in the complex domain, [10]:

$$\Gamma_k(n) = \left\{\left|\hat{X}_k(n)\right|^2 + |X_k(n)^2| - 2\left|\hat{X}_k(n)X_k(n)\cos\left(\Delta\varphi_k(n)\right)\right|\right\}^{\frac{1}{2}} \tag{4.2}$$

where the k^{th} spectral bin is quantified by calculating the Euclidean distance $\Gamma_k(n)$ between the observed $X_k(n)$ and that predicted by the previous frames, $\hat{X}_k(n)$; both derived from complex considerations based on Figure 6:

$$\begin{cases}\hat{S}_k(n) = \hat{X}_k(n)e^{j\hat{\phi}_k(n)} \\ S_k(n) = X_k(n)e^{\phi_k(n)}\end{cases}, \tag{4.3}$$

where $$\hat{\phi}_k(n) = princarg\left[2\varphi_k(n-1) - \varphi_k(n-2)\right]. \tag{4.4}$$

These distances are summed across the frequency-domain to generate a *complex difference* (CD) onset detection function:

$$CD(n) = \sum_{k=1}^{N}\Gamma_k(n)\ . \tag{4.5}$$

In a simpler equivalent way, Dixon [5] defined this function as:

$$CD(n) = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} |X(n,k) - X_T(n,k)| \quad (4.6)$$

where the target value $X_T(n, k)$ is estimated by assuming constant amplitude and rate of phase change:

$$X_T(n,k) = |X(n-1,k)| e^{\psi(n-1,k)+\psi'(n-1,k)}. \quad (4.7)$$

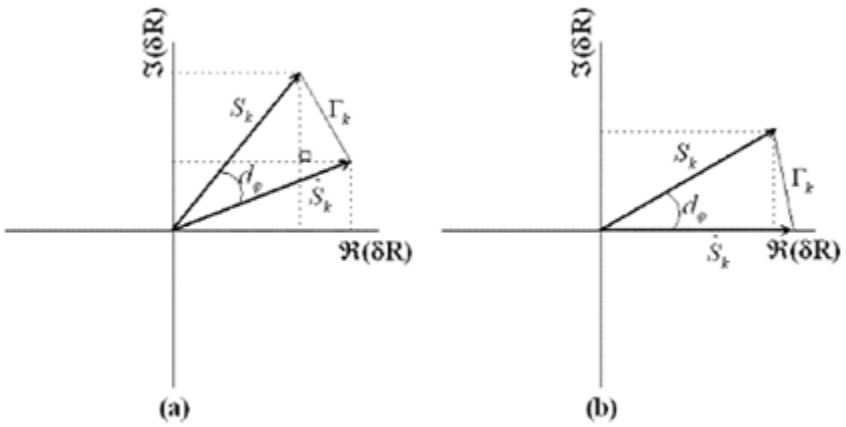


Figure 6 - Phasor diagram in the complex domain showing the phase deviation between target and current vector, and the Euclidean distance between them [10]: (a) normal diagram and (b) rotated diagram.

- **Time-Frequency and Time-Scale Methods (TFR):** An alternative to the analysis of the temporal envelope of the signal and of Fourier spectral coefficients is the use of time-scale or time frequency representations (TFR). The most notable algorithm implementing this scheme is the *wavelet regularity modulus*, which is a local measure of the regularity of the signal given by [11]:

$$WRM = \sum_{(j,k)\in ß[i]} 2^{js} d_{j,k} \quad (5.1)$$

where $d_{j,k}$ are the wavelet coefficients, ß[i] is the full branch leading to a given small-scale coefficient $d_{1,i}$, and *s* is a free parameter used to emphasize certain scales (*s=0* is often used in practice). A more detailed review on this approach is presented in *2.1.2*.

Since increases of WRM are related to the existence of large, transient-like coefficients in the branch ß[i], the *regularity modulus* can effectively act as an onset detection function.

- **Statistical Methods:** Statistical methods for onset detection are based on the assumption that the signal can be described by some probability model [6]. These schemes look for abrupt changes in the signal and register their likely times in a probabilistic manner. These can be quantified using likelihood measures or Bayesian

model selection criteria, and their success is intimately related to the proximity on the probability distribution described by the model to the "true" distribution of the data.

Based on [6], probabilistic schemes can be decomposed in *model-based change point detection* methods ([12]), through which change points are detected when the given likelihood ratio surpasses a fixed threshold (looking for an instantaneous switch between two distinct models); and in approaches based on *"surprise signals"* ([13]), which look for surprising moments relative to a single global model, through a detection function that trace of the negative log-probability of the signal given its recent history, according to a global model. A more detailed review is presented in *2.1.2*.

2.1.1.3 Onset Detection: Peak-Picking

The onsets are selected from the detection function by a peak-picking algorithm which finds local maxima in the detection function, subject to various constraints. The thresholds and constraints used in peak-picking have a large impact on the results, specifically on the ratio of false positives (reported detections where no onset exists) to false negatives (missed detections) [6]. This procedure can be decomposed into two subsequent processes: *thresholding* followed by *peak-picking*.

- **Thresholding:** The best values for thresholds are dependent on the application and the relative undesirability of false positives and false negatives. Therefore, it is necessary to define a threshold which effectively separates event-related and non-event-related peaks. There are two main approaches to defining this threshold: *fixed thresholding* and *adaptive thresholding*.

  Fixed thresholding define onsets as peaks where the detection function, *d(n)*, exceeds the threshold, δ (positive constant) ($d(n) \geq \delta$). However this approach is inefficient in the presence of dynamic music signals, tending to miss onsets, generally in quiet passages, while over-detecting during the loud ones.

  This invokes the use of a signal *adaptive threshold δ[n]*, generally computed as a smoothed version of the detection function. This smoothing can be linear, e.g. using a low-pass FIR-filter:

$$\delta[n] = \delta + \sum_{i=0}^{M} a_i d(n-i), \tag{6.1}$$

  with $a_0=1$; or non-linear, e.g. using the square of the detection function:

$$\delta[n] = \delta + \lambda \sum_{i=-M}^{M} \omega_i d^2(n+i) \tag{6.2}$$

  where λ is a positive constant and $\{\omega_i\}_{i=-M\ldots M}$ is a (smooth) window. Alternatively, in order to reduce the fluctuations, due to the presence of large peaks, the thresholding can be defined in percentiles, based, for instance, in local median:

$$\delta[n] = \delta + \lambda median\{|d(n-M)|, \dots, |d(n+M)|\}. \quad (6.3)$$

Dixon [5] used an *adaptive thresholding* function defined by:

$$\delta[n] = \max\left(d(n), \delta * \delta[n-1] + (1-\delta)d(n)\right). \quad (6.4)$$

- **Peak-Picking:** peak-picking is reduced to identifying local maxima above the defined threshold. As a representative example I will define Dixon's [5] approach where each onset detection function d(n) is normalised to have a mean of 0 and standard deviation of 1. In his scheme a peak at time $t = \frac{nh}{r}$ is considered an onset if it fulfils the following three conditions:

$$\begin{aligned} &1)\ d(n) \geq d(k) for\ all\ k\ such\ that\ n-\omega \leq k \leq n+\omega \\ &2)\ d(n) \geq \frac{\sum_{k=n-m\omega}^{n+\omega} d(k)}{m\omega+\omega+1} + \delta \\ &3)\ d(n) \geq \delta[n] \end{aligned} \quad (7.1)$$

where *ω=3* is the size of the window used to find a local maximum, *m = 3* is a multiplier so that the mean is calculated over a larger range before the peak, δ is the threshold above the local mean which an onset must reach, and δ[n] is the used threshold function (6.4).

For a review of a number of peak-picking algorithms for audio signals, see [14].

2.1.2 Audio Onset Detection Functions: A State-Of-The-Art Review

Earlier algorithms developed for onset detection focused mainly on the variation of the signal energy envelope in the time domain.

Based on the (instantaneous short-term) spectral structure of the signal, Masri [8] proposes a *high frequency content* (HFC) function with a linear frequency dependent weighting, which linearly weights each bin's contribution in proportion to its frequency. The *HFC* function produces sharp peaks during attack transients and is notably successful when faced with percussive onsets, where transients are well modelled as bursts of white noise. In a more general approach Masri, based on changes in the spectrum to formulate the detection function as a "distance" between successive short-term Fourier spectra, treating them as points in an N-dimensional space, developed a *spectral flux* (SF) onset detection method which calculates the spectral difference using the L_1-norm of the difference between magnitude spectra, (2.3).

Later, Duxbury [15] used a pre-processing scheme based on a constant-Q conjugate quadrature filter bank to separate the signal into five sub-bands. Using this approach he developed a hybrid scheme that considers energy changes in high-frequency bands and spectral changes (spectral flux) in lower bands. In order to calculate this spectral flux, the author proposed the use of the L_2-norm on the rectified difference:

$$SF(n) = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} \{H(|X(n,k)| - |X(n-1,k)|)\}^2 \quad (8.1)$$

where $(x) = (x + |x|)/2$, i.e., zero for negative arguments. This rectification has the effect of counting only those frequencies where there is an increase in energy, and is intended to emphasize onsets rather than offsets. By implementing a multiple-band scheme, the approach effectively avoids the constraints imposed by the use of a single reduction method, while having different time resolutions for different frequency bands.

Scheirer [16] demonstrated that much information from the signal can be discarded while still retaining the rhythmical aspect. On a set of test musical pieces, Scheirer filtered out different frequency bands using a filter bank (pre-processing). He extracted the energy envelope for each of those bands, using rectification and smoothing. Finally, with the same filter bank, he modulated a noisy signal with each of those envelopes and merged everything by summation. With this approach, rhythmical information was retained. On the other hand, care must be taken when discarding information. In another experiment, he shows that if the envelopes are summed before modulating the noise, a significant amount of information about rhythmical structure is lost.

Klapuri [17] used the psychoacoustical model developed by Scheirer to develop a robust onset detector, propounding the difference of the log spectral power in bands as a more psychoacoustically relevant feature related to the discrimination of intensity (simulating the ear's perception of loudness). Hence, to get better frequency resolution, he employed a pre-processing consisting in a filter bank of 21 filters. The author points out that the smallest detectable change in intensity is proportional to the intensity of the signal. Thus $\Delta I/I$ is a constant, where I is the signal's intensity. Therefore, instead of using $(d/dt)A$ where A is the amplitude of the envelope, he used

$$\frac{1}{A}\left(\frac{d}{dt}A\right) = \frac{d}{dt}\log(A). \quad (8.2)$$

This provided more stable onset peaks and allowed lower intensity onsets to be detected. Later, Klapuri *et al.* [18] used the same kind of pre-processing and won the ISMIR[7] (International Symposium on Music Information Retrieval) 2004 tempo induction contest.

Jehan [12], also motivated on psychoacoustically factors, forms an event detection function by taking power in Bark bands and applying a spectral masking correction based on spreading functions familiar from the perceptual coding of audio, and post masking with half cosine convolution. His applications are in event sensitive segmentation.

In contrast to Scheirer's and Klapuri's works, Bello *et al.* [9] took advantage of phase information to track the onset of a note (*phase deviation*). They found that at steady state, oscillators tend to have predictable phase. This is not the case at onset time, allowing the decrease in predictability to be used as an indication of note onset. To measure this, they

[7] For information consult http://www.ismir.net/ .

collected statistics on the phase acceleration, as estimated by (3.2). To detect the onset, different statistics were calculated across the range of frequencies including mean, variance, and kurtosis. These provide an onset trace, which can be analyzed by standard peak-picking algorithms. The function considers all frequency bins *k* equally.

Dixon [5], declaring that the energy of the signal is concentrated within the bins which contains the partials of the currently sounding tones, developed an improvement in this phase-based detection function, proposing weighting the frequency bins *k* by their magnitude. Based on this he built a new onset detection function, called the *weighted phase deviation* (WPD), and defined by:

$$WPD(n) = \frac{1}{N}\sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} |X(n,k)\varphi''(n,k)|. \tag{8.3}$$

This is similar to the complex functions (analysed below), in which the magnitude and phase are considered jointly, but with a different manner of combination. The author further proposed another option to define a weighted phase deviation function, but in a normalised way (NWPD), where the sum of the weights is factored out to give a weighted average phase deviation:

$$NWPD(n) = \frac{\sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} |X(n,k)\varphi''(n,k)|}{\sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} |X(n,k)|}. \tag{8.4}$$

Subsequently Bello *et al*. presented three studies, [10], [7], [19], on the combined use of energy and phase information for the detection of onsets in musical signals, in the form of complex domain methods, developing some *complex difference* (CD) algorithms. They showed that by combining phase and energy approaches it's possible to achieve a more robust onset detection scheme, enjoying from both performances: energy-based onset detection functions perform well for pitched and non-pitched music with significant percussive content, while phase-based approaches provide better results for strongly pitched signals and are less robust to distortions in the frequency content and to noise. This was corroborated by the experimental results achieved for a large range of audio signals.

Dixon [5] has also proposed an improvement to these complex methods, trying to resolve their absence in distinguish increases from decreases in the amplitude of the signal (i.e. onsets from offsets). They formulated such resolution in the form of a *rectified complex domain* (RCD) function, by applying a half-wave rectification to a spectral flux-based function, which grants exclusive consideration to increases in energy in spectral bins. Therefore the RCD is basically the incorporation of this scheme in a CD method, being described as follows:

$$RCD(n) = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}} RCD(n,k) \tag{8.5}$$

where $$RCD(n,k) = \begin{cases} |X(n,k) - X_T(n,k), & if\,|X(n,k)| \geq |X(n-1,k)| \\ 0, & otherwise \end{cases}. \tag{8.6}$$

Approaching a time-frequency/time-scale analysis, Daudet *et al.* [11] developed a transient detection based on a simple dyadic wavelet decomposition of the residual signal. This transform, using the Haar wavelet[8], was chosen for its simplicity and its good time localization at small scales. The scheme takes advantage of the correlations across scales of the coefficients. The significance of full-size branches of coefficients, from the largest to the smallest scale, can be quantified by a regularity modulus, which is a local measure of the regularity of the signal, (5.1).

Now considering the probability models I shall refer again the work of Jehan [12], which also developed a statistical scheme based on the comparison between two auto-regressive models of the signal, forming a model-based change point detection method. Like other similar approaches he uses two parametrical Gaussian statistical models, $\mathcal{A}$, $\mathcal{B}$, where their log-likelihood ratio *s* is defined as:

$$s = log \frac{p_{\mathcal{B}}(x)}{p_{\mathcal{A}}(x)}, \tag{8.7}$$

assuming that *s* will change sign, due to the signal convergence from model $\mathcal{A}$ to model $\mathcal{B}$, at some unknown time. In its approach both models parameters and the change point are then optimized to maximize the log-likelihood ratio between the probability of having a change at and the probability of not having an onset at all. Change points are detected when this likelihood ratio surpasses a fixed threshold.

In a distinct statistic way, based on "surprise signals", Abdallah and Plumbley [13] introduced the *negative log-likelihood using an ICA* (Independent Component Analysis) model scheme. Their approach was developed on the notion of an observer which builds a model of a certain class of signals, such that it is able to make predictions about the likely evolution of the signal as it unfolds in time. Such an observer will be relatively surprised at the onset of a note because of its uncertainty about when and what type of event will occur next. However, if the observer is in fact reasonably familiar with typical events (i.e., the model is accurate), that surprise will be localized to the transient region, during which the identity of the event is becoming established. Thus, a dynamically evolving measure of surprise, or onset (novelty), can be used as a detection function.

Ultimately, some recent models, despite being rare, approach onset detection mixed to supervised (machine) learning. Davy and Godsill [20] also based on time-frequency and time-scale analysis (TFR) developed an audio segmentation algorithm using a support vector machine[9] (SVM). They classify spectrogram frames into being probable onsets or not. The SVM

[8] The Haar wavelet is the first known wavelet and was proposed in 1909 by Alfréd Haar. It is the simplest possible wavelet, but has the advantage of not being continuous and therefore not differentiable.

[9] Support vector machines (SVMs) are a set of related supervised learning methods used for classification and regression. They belong to a family of generalized linear classifiers. A tutorial on SVMs has been produced by C.J.C Burges, http://research.microsoft.com/~cburges/papers/SVMTutorial.pdf .

was used to find a hypersurface delimiting the probable zone from the less probable one. Unfortunately, no clear test was made to outline the performance of the model.

Kapanci and Pfeffer [21] also used SVM, on a set of frame features to estimate if there is an onset between two selected frames. Using this function in a hierarchical structure, they were able to find the position of onsets. Their approach mainly focuses on finding onsets in signals with slowly varying change over time such as solo singing.

Marolt et al. [22] used a neural network approach for note onset detection. The model used the same kind of pre-processing as Scheirer's in [16], with a filter bank of 22 filters. An integrate-and-fire network was then applied separately to the 22 envelopes. Finally, a multi layer perception was applied on the output to accept or reject the onsets. Results were good but the model was only applied to mono-timbral piano music.

In a similar approach, in its use of neural networks, Eck *et al.* [23], [24] most notably developed two onset detection algorithms that have participated to the MIREX 2005 onset detection contest, yielding the best and second best performance. Both proposed algorithms, first classify frames of a spectrogram into onset or non-onset, using a feed-forward neural network. From the classification of each frames, they extract the onset times, using a simple peak picking algorithm, based on a moving average. Distinctly the first version, SINGLE-NET (Figure 7a), is comprised of a time-space transform (spectrogram) which is in turn treated with a feed-forward neural network (FNN), and the resulting trace is fed into a peak-picking algorithm to find onset times (OSTs). The second one, MULTI-NET (Figure 7b), repeats the SINGLE-NET variant multiple times, applying different hyper-parameters. A tempo detection algorithm is run on each of the resulting feed-forward neural network (FNN) outputs, and the SINGLE-NET outputs and the tempo-detection outputs are then combined using a second neural network.

With this work, they concluded that a supervised learning approach to note onset detection performs well and warrants further investigation.

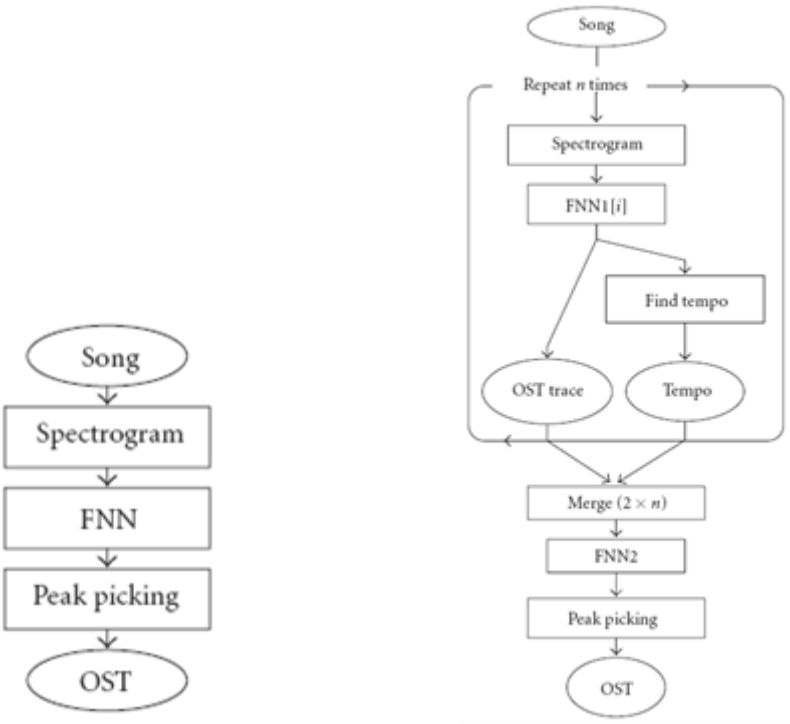


Figure 7 – Algorithms description, [24]: **a)** SINGLE-NET flowchart: uses a single neural network classifier. **b)** MULTI-NET flowchart: combines the predictions of several networks, SINGLE-NETs, trained using different hyper-parameters.

With so many possible algorithms for the detection of musical events in an audio signal now published, which some are referred above, research questions are turning to the comparative evaluation of such methods, [25]. Therefore, below I present some approaches on state-of-the-art comparative evaluation reviews addressing onset detection methods.

Bello *et al.*, in [6], presented a tutorial on onset detection in music signals, where they reviewed, categorized, and compared some of the most commonly used techniques for onset detection, also presenting possible enhancements, and providing some guidelines for choosing the appropriate method for a given application. The discussed methods were based on several predefined signal features, namely the signal's amplitude envelope (wavelet methods), spectral magnitudes and phases (spectral and phase-based methods), time-frequency representations (temporal methods); and methods based on probabilistic signal models (statistical methods). The achieved results are discussed in the following subsection, *2.1.3 Results Analysis and Comparison*.

Based on Bello *et al.* work, Collins, in [25], and then Dixon, in [5], sought to extend their results and reviewed them, as a benchmark for comparison.

Collins besides reviewing and extending their work he also explored the potential of psychoacoustically motivated models such as those of Klapuri [17] and Jehan [12], referred

above. In this research they investigated 16 detection functions, including a number of novel and recently published models.

Dixon [5] complemented and extended Bello *et al.*'s work by introducing new onset detection functions, already referred, and by testing the new methods alongside independent implementations of a subset of the published methods on the same data set and on a second data set which is two orders of magnitude larger. They restricted their comparison to methods based on short term spectral coefficients, which are the most widely used methods, and the most successful according to the 2005 MIREX audio onset detection evaluation. The achieved results are also discussed in the following subsection, in comparison to the previous ones.

2.1.3 Results Analysis and Comparison

In signal processing, onset detection is an active research area leading to worldwide contests as the Audio Onset Detection contest featured by the MIREX[10] (Music Information Retrieval Evaluation eXchange) annual competition, an ISMIR member since 2005. These contests aim to find the more efficient onset detection function at retrieving the time locations at which all musical events in a recording begin. We begin this subsection by announcing the main difficulties on onset detection model's evaluation and then we introduce some methods to overcome this with the use of tools for building a set of reference onset times. Next, based on [6], [5], we present a comparison between some of the referred (above) reduction models, emphasizing the ones with best performance, depending both on its definition and on the nature of the signals to be analyzed. We finish this comparison by introducing some guidelines for choosing the right detection function. Ultimately, as a point of interest, we present the overall scores from the MIREX 2005 Audio Onset Detection Contest.

2.1.3.1 Methodology and Tools for the Evaluation of Automatic Onset Detection Algorithms

The main difficulty with the evaluation of onset detection algorithms is that of obtaining a significantly large and balanced set of recordings for which the onset times are known (ground truth data). Precise measurements of onset times are only available for a small fraction of music, such as piano music recorded on computer-monitored pianos, and music generated with a MIDI synthesizer. Other data must be labelled by hand, which is a laborious and error-prone task.

A second methodological problem is determining how to report and compare results. Each onset detection function has parameters which can be tuned to alter the proportion of false

[10] For additional information consult MIREX 2008 web page at http://www.music-ir.org/mirex/2008/index.php/Main_Page .

positives and false negatives. These proportions are often expressed in terms of precision and recall whose relationship is often shown graphically in a receiver operating characteristic (ROC) curve, and if a single scalar statistic is desired (to make comparisons simple), the precision and recall can be combined into a single value such as the area under the ROC curve or the F-measure, which represents the optimal point on this curve.

A third problem is how to deal with situations where a number of onsets are very close together, for example when a chord is played on a guitar or piano. Depending on the time between the notes, one or more onsets might be perceived, but this is dependent on the instrument and presence of other simultaneous sounds. The MIREX[10] [26] onset detection evaluation addressed this problem by counting the number of merged onsets (two onsets detected as a single onset) and double onsets (a single onset recognized as two) in addition to the standard counts of correct detections, false positives and false negatives.

Addressing these issues we present two distinct frameworks to improve the performance of the evaluation of algorithms for the automatic note onset detection in music signals.

Leveau *et al.* [27] developed a carefully designed software tool, called SOL (Sound Onset Labellizer, Figure 8), which combines the three most used hand-label methods (signal plot, spectrogram, listening to signal slices), to provide a methodology to construct the set of reference onset times and cross-validate it amongst different expert listeners. With this application they've objected to build a common methodology and a common annotation tool, which in turn can be used to build a common database of onset-annotated files. In order to be shared by the widest community they disposed this software and files online[11], freely available.

In order to enable, coordinate and evaluate submissions to MIREX, a software framework was developed by J. Downie [26] in association with the IMIRSEL team. Their final solution is based in the Data-to-Knowledge (D2K) Toolkit and is included as part of the Music-to-Knowledge (M2K) Toolkit[12] (both implemented in JAVA). M2K modules are connected by an XML-based itinerary which describes the particular process flow for each evaluation task. These frameworks are extremely flexible and can be customized by participants to suit the specific topologies of their submissions. Figure 9 represents the Audio Onset Detection contest M2K MIREX evaluation itinerary.

This represents a significant advance over traditional evaluation frameworks and supports the central evaluation paradigm necessitated by the unique challenges posed by MIR evaluation.

[11] Available at http://perso.telecom-paristech.fr/~grichard/ISMIR04/ .
[12] M2K is an open-source initiative, and is freely available from http://music-ir.org/evaluation/m2k .

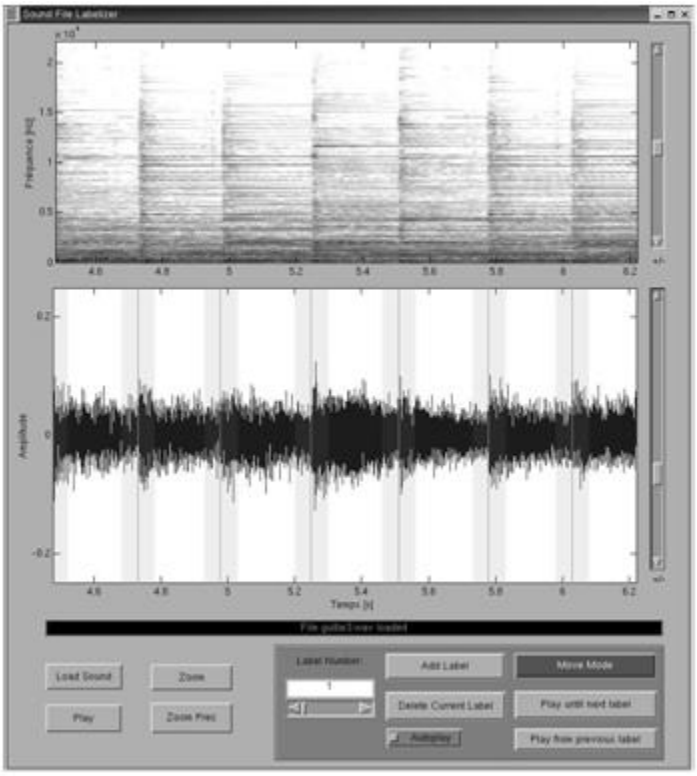


Figure 8 - *Sound Onset Labellizer* interface, [27]. The screen of the GUI is divided into three parts: the upper one represents the spectrogram of the signal, the middle one its time domain waveform, and the lower the controls to manipulate sound files, labels and visualization windows.

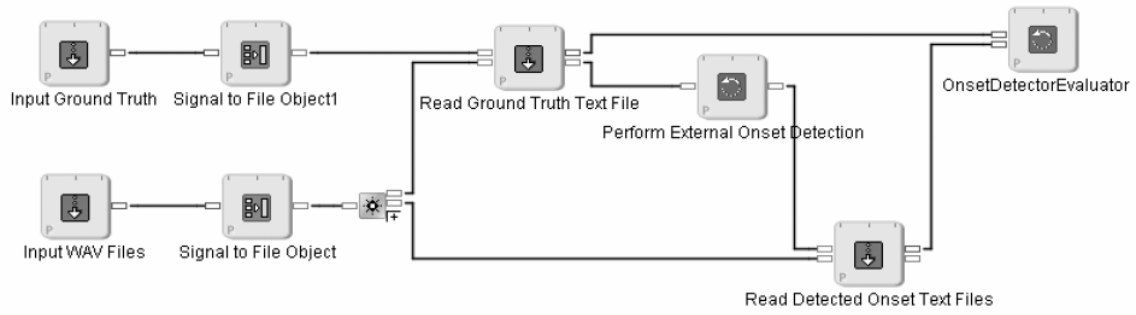


Figure 9 - MIREX evaluation framework implemented in M2K for the Audio Onset Detection contest, [26].

2.1.3.2 Onset Detection Function: Comparison of Performance Results

Following the previous analysis, in this subsection we present Bello *et al.* [6] and Dixon's [5] experimental results comparing some of the onset detection approaches described.

To test every relevant scheme, they have both used the same mono data set of 44.1KHz 16 bit sound files, with reference onsets marked up by hand by a single expert. The tests were composed by 4 sets of short excerpts from a range of instruments, classed into the following groups:

- NP — non-pitched percussion, such as drums (119 onsets);
- PP—pitched percussion, such as piano and guitar (577 onsets);
- PN — pitched non-percussion, in this case solo violin (93 onsets);
- CM — complex mixtures from popular and jazz music (271 onsets).

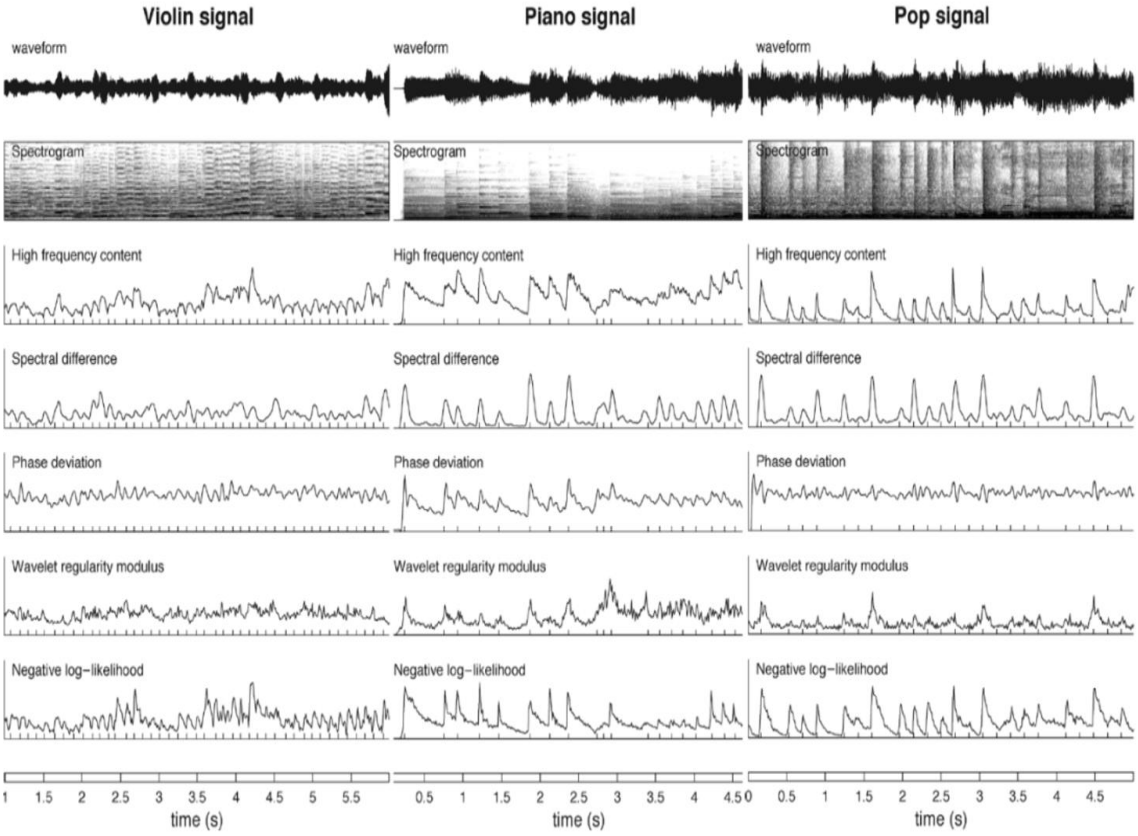


Figure 10 - Comparison of different detection functions for 5 s of [6]: **a)** a solo violin recording. **b)** a piano violin recording **c)** a pop song. From top to bottom: time-domain signal, spectrogram, high-frequency content, spectral difference, spread of the distribution of phase deviations, wavelet regularity modulus, and negative log-likelihood using an ICA model. All detection functions have been normalized to their maximum value.

In [6] the onset labelling was done mostly by hand, which is a lengthy and inaccurate process, especially for complex recordings such as pop music: typically including voice, multiple instruments and post-production effects. A small subsection of the database corresponds to acoustic recordings of MIDI-generated piano music which removes the error introduced by hand-labelling. In a way to allow for its inaccuracy, the authors considered correct matches the ones which target and detected onsets were within a 50-ms window.

Peak-picking was accomplished using the moving-median adaptive threshold method[13]. Table 1 present their achieved results depending on the characteristic of each method: Spectral Weighting Methods - *Spectral Flux* (SF) [8], High-Frequency Content (HFC) [8]; Phase-Based Methods - *Phase Deviation* (PD) [19]; Time-Frequency and Time-Scale Methods - *Wavelet Regularity Modulus* (WRM) [11]; Statistical Methods - *Negative Log-Likelihood* (NL) [13]. For the sake of a fair comparison between the detection functions, the authors opted to use a common post-processing and peak-picking technique. However, the performance for each detection function could be improved by fine tuning the peak-picking algorithm for specific tasks.

A behavioural overview of every analysed methods when applied to different types of audio signals (violin, piano, and pop music) is presented in Figure 10.

Figure 11, then, present the results achieved in Table 1 in the form of a ROC curve, comparing the performance of each tested scheme. To compose this curve, all peak-picking parameters (e.g., filter's cutoff frequency, λ) were held constant, except for the threshold δ which was varied to trace out the performance curve. Better performance is indicated by a shift of the curve upwards and to the left. The optimal point on a particular curve can be defined as the closest point to the top-left corner of the axes, where the error is at its minimum.

Table 1 - Onset Detection Results from [6]. Columns show the percentage of True Positives (TP%) and percentage of False Positives (FP%) for each method: HFC, SF, PD, WRM, NL.

PITCHED NON-PERCUSSIVE - 93 ONSETS

METHOD	TP %	FP %
High frequency content	81.7	14.7
Spectral difference	87.1	8.6
Phase deviation	95.7	4.3
Wavelet reg. modulus	92.5	10.1
Neg. log-likelihood	96.8	3.2

NON-PITCHED PERCUSSIVE - 212 ONSETS

METHOD	TP %	FP %
High frequency content	96.7	0.0
Spectral difference	81.6	5.5
Phase deviation	80.7	5.5
Wavelet reg. modulus	88.7	2.2
Neg. log-likelihood	92.9	1.7

PITCHED PERCUSSIVE - 489 ONSETS

METHOD	TP %	FP %
High frequency content	94.1	5.4
Spectral difference	94.9	1.6
Phase deviation	95.5	0.3
Wavelet reg. modulus	92.7	5.1
Neg. log-likelihood	92.4	3.1

COMPLEX MIX - 271 ONSETS

METHOD	TP %	FP %
High frequency content	84.5	10.8
Spectral difference	80.4	10.4
Phase deviation	80.1	24.7
Wavelet reg. modulus	81.9	27.7
Neg. log-likelihood	86.0	10.8

[13] Based on X. Rodet and F. Jaillet, "Detection and modeling of fast attack transients," in Proc. Int. Computer Music Conf., Havana, Cuba, 2001, pp. 30–33.

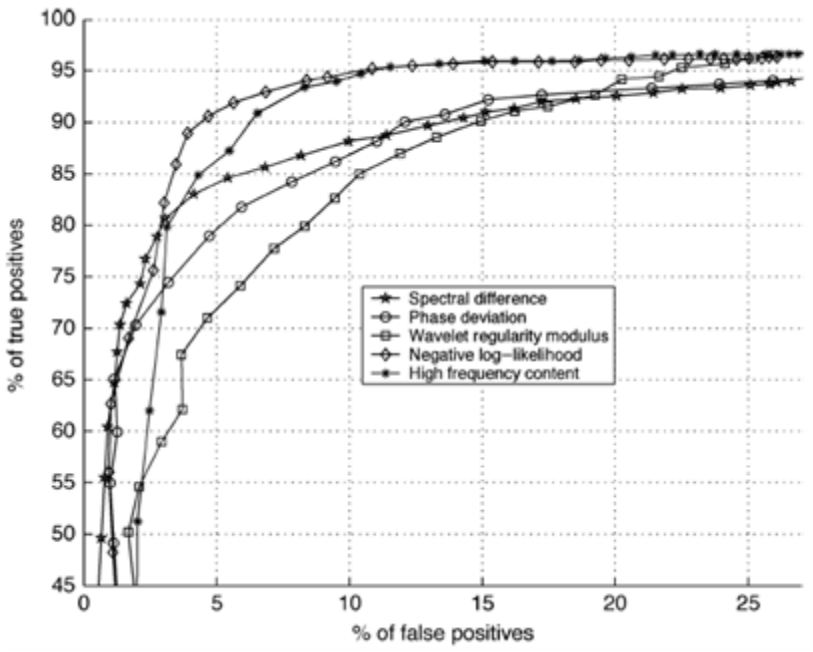


Figure 11 - ROC curve comparison of the onset detection presented in Table 1, [6]: HFC, SF, PD, WRM, NL.

Following [6] considerations, by reading the different optimal points we can retrieve the best set of results for each onset detection method. For the complete database, the *negative log-likelihood* (90.6%, 4.7%) performed the best, followed by the *HFC* (90%, 7%), *spectral flux* (83.0%, 4.1%), *phase deviation* (81.8%, 5.6%), and, finally, the *wavelet regularity modulus* (79.9%, 8.3%).

By analysing the shape of each curve (as it contains useful information about the properties of each method): the *negative log-likelihood* shows appeal for a number of applications by remaining close to the top-left corner of the axes ([100% TP, 0% FP] point), and by successfully characterizing all types of onsets while producing little unrelated noise; the *HFC* is able to retrieve a large proportion of the existing onsets for relatively few false positives, reaching 95% true positives for 10% false positives; the *wavelet regularity modulus* present a similar performance, although the corresponding performance curve rises more slowly as the percentage of false positives increases; the *spectral flux* and the *phase deviation* (methods that take information from a number of temporal frames into consideration) present a smoother detection function profile, minimizing the amount of spurious detections. This was reflected in a ROC curve that manages relatively high correct onset detection rates for low numbers of false positives, while obtaining comparatively less good detections for high rates of false positives (more than 25%).

According to Table 1, and quoting [6], we present a performance analysis depending on the type and quality of the input signal. The *phase deviation* performed successfully for pitched sounds (both PP and NP) where tonal information is the key to the detection of onsets, while returning poor results for purely percussive sounds and complex mixtures. On

the other hand, the *HFC* performed better for highly percussive sounds and complex mixtures (with drums) than for music with softer onsets. The *spectral flux* sits in the middle, slightly below *phase deviation* for pitched sounds and just under-performing *HFC* for more percussive and complex sounds. The *wavelet regularity modulus* performance was at its best when dealing with simple percussive sounds, otherwise performing poorly with respect to the other methods. Notably, the *negative log-likelihood* performed relatively well for almost all types of music.

Following this results, I present Dixon's [5]. In each case, the results are shown for the point on the ROC curve (which gives the maximum value of the F-measure). The ground-truth data was then used to select optimal values of α (positive thresholding constant) and δ (the threshold above the local mean which an onset must reach). Similarly to [6] he considered an onset to be correctly matched if a detected onset is reported within 50 ms of the ground-truth onset time. However he didn't penalize merged onsets considering that the analyzed data contained many simultaneous or almost simultaneous notes (and he wasn't attempting to recognize the notes). Contrarily to [6] he presented the achieved results, in Table 2, in function of their precision P, recall R and F-measure F (for the optimal parameter settings), which were given by:

$$P = \frac{c}{c+f^{+}} \tag{9.1}$$

$$R = \frac{c}{c+f^{-}} \tag{9.2}$$

$$F = \frac{2PR}{P+R} = \frac{2c}{2c+f^{+}+f^{-}} \tag{9.3}$$

where c is the number of correct detections, f^{+} is the number of false positives and f^{-} is the number of false negatives. Parameters were chosen to maximize F.

In this context and in order to compare its results with [6], he re-tested the *spectral flux* (SF), the *phase deviation* (PD), and also tested his proposed models (*WPD*, *NWPD*, *and RCD*) in contrast to former similar ones (respectively, *PD*, *PD*, and *CD*).

Therefore, tested on the 4 data sets used in [6], the Table 2 shows the results for 8 different onset detection functions: Energy-Based Methods - *Spectral Flux* (SF* in [6] and SF in [5]) [8]; Phase-Based Methods - *Phase Deviation* (PD* in [6] and PD in [5]) [19], *Weighted Phase Deviation* (WPD) [5], *Normalised Weighted Phase Deviation* (NWPD) [5]; Complex Methods - *Complex Domain* (CD) [10], [7], *Rectified Complex Domain* (RCD) [5].

Table 2 - Results of onset detection tests in [5], showing precision (P), recall(R) and F-measure (F) for the data sets pitched non-percussive (PN), pitched percussive (PP), non-pitched percussive (NP) and complex mixture (CM), for 8 different onset detection functions (presented above). The functions marked with asterisks are results in [6].

	PN data			PP data			NP data			CM data		
	P	R	F	P	R	F	P	R	F	P	R	F
SF*	0.914	0.871	**0.892**	0.984	0.949	**0.966**	0.945	0.816	**0.876**	0.896	0.804	**0.848**
SF	0.938	0.968	**0.952**	0.981	0.988	**0.984**	0.959	0.975	**0.967**	0.882	0.882	**0.882**
PD*	0.957	0.957	**0.957**	0.997	0.955	**0.976**	0.945	0.807	**0.871**	0.753	0.801	**0.776**
PD	0.654	0.935	**0.770**	0.482	0.865	**0.619**	0.750	0.933	**0.831**	0.663	0.749	**0.704**
WPD	0.937	0.957	**0.947**	0.899	0.925	**0.912**	0.974	0.958	**0.966**	0.843	0.830	**0.836**
NWPD	0.909	0.968	**0.938**	0.961	0.981	**0.971**	0.950	0.966	**0.958**	0.916	0.845	**0.879**
CD	0.946	0.946	**0.946**	0.971	0.984	**0.978**	0.948	0.924	**0.936**	0.941	0.819	**0.876**
RCD	0.948	0.978	**0.963**	0.983	0.979	**0.981**	0.944	0.983	**0.963**	0.945	0.819	**0.877**

By analyzing Table 2 is observable that there are some large discrepancies between Bello's results (marked with * - *SF** and *PD**) and Dixon's own implementations of the same functions (SF and PD).

Quoting [5] analysis, SF* performed particularly well with the data set PP in comparison with the PN and NP data sets, but his implementation (SF) showed much smaller performance differences across these 3 sets of excerpts. SF also achieved better performance across the entire range of data, presumably due to a better peak-picking function. Even greater differences are evident in the results of the phase deviation functions, where the author's PD function achieved much worse performance than Bello's PD* results. The closeness of the PD* results to the WPD and NWPD results raised the suspicion that perhaps some weighting scheme have been used in the PD* algorithm, and this was later confirmed by one of the authors from [6], who had mistakenly thought it was an unimportant detail.

WPD and NWPD are both very significant improvements on the PD function, but the normalization was only an improvement on the WPD in two cases (PP and CM), while for the other two cases a slight degradation in performance resulted. Finally, the RCD method offered a small improvement on the CD on this data, but considering the small size of the data set, this difference might not be significant.

Overall, these results showed that *spectral flux*, *weighted phase deviation* and *complex domain* methods can all achieve a similarly high level of performance on these data sets. Since the data sets are small and not sufficiently general, Dixon didn't draw further conclusions about the differences between these methods, except to state that spectral flux has the advantage of being the simplest and fastest algorithm.

Summarizing, some of these results contradicted Bello's and suggest that a similarly high level of performance can be obtained with a magnitude-based (*spectral flux*), a phase-based (*weighted phase deviation*) or a complex domain (*complex difference*) onset detection function.

As conclusion one shall say that all the discussed results (from [6] and [5]), while depicting a general trend in the behaviour of these approaches, are not absolute, due to the method's signal dependencies and to the chosen peak-picking and post-processing algorithms. The hand-labelling of onsets, in the ground-truth definitions, can also be ambiguous and subjective, especially in complex mixes.

2.1.3.3 Onset Detection Functions: Comparative Analysis and Methods Applications

When picking the most accurate method to an application the general rule of thumb is that one should choose the method with minimal complexity that satisfies the requirements of that application, in a balance of complexity between pre-processing, construction of the detection function, and peak-picking [6].

What follows is a summary discussion of the merits of different reduction approaches and some guidelines to find the appropriate method for a specific application, with an emphasis on the ones that have been previously compared, founded mainly on [6] and [5] results.

We decomposed this general discussion in six different methods, according to *2.1.1.2*:

- **Temporal Methods (** [17]**):** these are simple and computationally efficient. Their functioning depends on the existence of clearly identifiable amplitude increases in the analysis signal, which is the case only for highly percussive events in simple sounds. Amplitude-based onset detection schemes decreases in robustness when facing amplitude modulations (i.e., vibrato, tremolo) or the overlapping of energy produced by simultaneous sounds. These methods present low precision in onsets time localization.

  Bello *et al*. consider these methods especially adequate to very percussive (PP – e.g., drums) music signals.

- **Energy-Based (Spectral Weighting) Methods (** [8]**,** [15]**):** from these we shall refer the commonly used *HFC* (High Frequency Content, [8]). It is successful at emphasizing the percussiveness of the signal but less robust at detecting the onsets of low-pitched and non-percussive events, where energy changes are at low frequencies and hence de-emphasized by the weighting. In some signals, even broadband onsets are susceptible to masking by continuous high-frequency content such as that due to open cymbals in a pop recording. This problem can be overcome by using *spectral difference* (*spectral flux*) methods such as the L_1-norm of the difference between magnitude spectra (2.3), [8], or the L_2-norm of the rectified spectral difference (8.1), [15], as these can respond to changes in the distribution of spectral energy, as well as the total, in any part of the spectrum. However, the difference calculation only relies on magnitude information.

Bello *et al*. consider the *SF*, similarly to *PD* (below), especially adequate to strongly pitched transients.

- **Phase-Based Methods (** [9]**,** [19]**,** [5]**):** these were designed in order to compensate the former scheme shortcomings. We shall refer the spread of the distribution of *phase deviations* (PD) (3.2), [19], which are successful at detecting low and high-frequency tonal changes regardless of their intensity. Yet they are especially susceptible to variations introduced by the phases of noisy low-energy components, and to phase distortions common to complex commercial music recordings. The *WPD* (8.3) and *NWPD* (8.4), proposed in [5], are both very significant improvements on the *PD* function, but the normalization is only an improvement on the *WPD* in the presence of pitched-percussive (PP) or complex mixture (CM), while for pitched non-percussive (PN) and non-pitched percussive (NP) data a slight degradation in performance results.

  As referred, Bello *et al*. also consider the *PD* especially adequate to strongly pitched transients.

- **Complex Methods (** [10]**,** [7]**,** [5]**):** In the complex domain, both phase and amplitude information work together, offering a generally more robust onset detection scheme. This algorithm is both straightforward to implement, and computationally cheap. Despite this, it proves effective for a large range of audio signals. Some complex domain approaches, like the *complex difference* (CD) (4.5), [7], currently performing better on the lower frequency components of the spectrum, may be beneficial to incorporate them within a multi-resolution scheme. This has the advantage that high frequency noise bursts may be used to improve time localization of hard onsets. The *RCD* (8.5) method [5] has revealed to offer small performance improvement, presenting non-significant differences.

  Bello *et al*. consider the *CD* a good choice to any application in general, at the cost of a slight increase in computational complexity.

- **Time-Frequency and Time-Scale Methods (TFR) (** [20]**,** [11]**):** As representation we shall refer the *wavelet regularity modulus* (WRM) (5.1), [11], being an example of an approach using an alternative time-scale representation that can be used to precisely localize events down to a theoretical resolution of as little as two samples of the original signal, which for typical audio sampling rates is considerably better than the ear's resolution in time. The price of this is a much less smooth detection function, therefore emphasizing the need for post-processing to remove spurious peaks. The

method provides an interesting alternative to other feature-based methods, but with an increase in algorithmic complexity.

Bello *et al*. consider the *WRM* especially useful, possibly in combination with another method, to applications requiring precise onsets time localization.

- **Statistical Methods (** [13]**):** Approaches based on probabilistic models provide a more general theoretical view of the analysis of onsets. Previous reduction methods can be explained within the context of measuring surprise relative to a probabilistic model, while new methods can be proposed and evaluated by studying refinements or alternatives to existing models. An example is the surprise-based method using *ICA* to model the conditional probability of a short segment of the signal, calculated as the difference between two negative log-likelihoods [13]. If the model is adequate, then robust detection functions for a wide range of signals can be produced. However, for adaptive statistical models, such as *ICA*, these advantages accrue only after a potentially expensive and time-consuming training process during which the parameters of the model are fitted to a given training set.

  Bello *et al*. consider *ICA* the best option, due to their best overall scores and less dependence on a particular choice of parameters, if a high computational load is acceptable, and a suitable training set is available.

Contesting some of [6] presented results and conclusions, Dixon [5] argued that the SF, PD, and the CD onset detection functions achieved higher level of performance, where SF has proved to be the best scheme, proving to be the most accurate with the lowest computational complexity.

Based on Dixon's conclusions this was the implemented method in Marsyas, for onset detection, as a low-level rhythmic perception model.

Either way, by using Marsyas in the development of my Music Analysis Module (see *chapter 3*), this was necessarily the basis of my robot's rhythm perception.

2.1.3.4 MIREX 2005 Audio Onset Detection Contest Overall Scores

Following I preset the results obtained in the MIREX 2005 Audio Onset Detection Contest.

On MIREX the results are given by comparing the detected onset times with the ground-truth ones[14]: for a given ground-truth onset time, determined with the evaluation framework implemented in M2K [26] (see subsection 2.1.3.1), if there is a detection in a tolerance time-window around it, it is considered as a correct detection (CD). If not, there is a false negative

[14] The presented information in summarized from: http://www.music-ir.org/evaluation/mirex-results/audio-onset/index.html

(FN). The detections outside all the tolerance windows are counted as false positives (FP). Doubled onsets (two detections for one ground-truth onset) and merged onsets (one detection for two ground-truth onsets) will be taken into account in the evaluation. Doubled onsets are a subset of the FP onsets, and merged onsets a subset of FN onsets. On MIREX 2005, the dataset consisted of 85 audio files (14.8 minutes total) from 9 classes: complex, poly-pitched, solo bars and bells, solo brass, solo drum, solo plucked strings, solo singing voice, solo sustained strings, and solo winds.

Table 3 - Overall scores from the MIREX 2005 audio onset detection contest. Overall average F-measure, overall average precision, and overall average recall are weighted by number of files in each of nine classes (retrieved from [24]).

Rank	Participant	Avg. F-measure	Avg. precision	Avg. recall
1	Lacoste & Eck (MULTI-NET)	80.07%	79.27%	83.70%
2	Lacoste & Eck (SINGLE-NET)	78.35%	77.69%	83.27%
3	Ricard, J.	74.80%	81.36%	73.70%
4	Brossier, P.	74.72%	74.07%	81.95%
5	Röbel, A. (2)	74.64%	83.93%	71.00%
6	Collins, N.	72.10%	87.96%	68.26%
7	Röbel, A. (1)	69.57%	79.16%	68.60%
8	Pertusa, Klapuri, & Iñesta	58.92%	60.01%	61.62%
9	West, K.	48.77%	48.50%	56.29%

Table 4 - F-measure percentages for all nine classes from the MIREX 2005 audio onset detection contest. Best performance for each class is shown in bold. The number of pieces for each class is shown in parentheses (retrieved from [24]).

	Complex (15)	Poly-pitched (10)	Bars and bells (4)	Brass (2)	Drum (30)	Plucked string (9)	Singing voice (5)	Sust. strings (6)	Wind (4)
MULTI-NET	**78.85**	**86.31**	86.55	70.25	91.40	81.84	**45.33**	56.68	58.75
SINGLE-NET	77.02	85.93	86.37	67.88	89.91	83.49	34.35	52.87	56.48
Ricard, J.	71.90	83.26	87.17	**72.66**	90.97	77.85	27.59	38.45	38.57
Brossier, P.	76.16	80.88	73.97	64.88	86.28	79.99	22.16	**57.92**	52.08
Röbel, A. (2)	62.84	76.24	90.34	68.32	89.96	**84.20**	40.68	36.18	**66.01**
Collins, N.	60.25	75.70	**99.28**	69.09	**92.31**	81.97	29.34	14.74	47.57
Röbel, A. (1)	59.76	69.29	97.92	61.87	86.29	77.58	42.69	17.35	51.13
Pertusa et al.	50.16	59.37	60.22	54.41	77.22	67.74	11.12	38.45	25.59
West, K.	47.13	39.98	34.58	33.94	71.61	39.85	12.07	32.12	18.11

As observable in Table 3 and Table 4, on this dataset, the MULTI-NET and SINGLE-NET algorithms [24] yielded the best and second best performance, respectively, for the contest.

The MUTI-NET, yielding an F-measure of 80.07% performed slightly better than the SINGLE-NET algorithm which achieved an F-measure of 78.35%.

2.2 Dancing and Rhythmic Interactive Robots

This subsection presents the solutions achieved by some of the most notable current researchers on the area of robotic dancing and interactive robots. In order to clarify this overview, it is divided into three main topics: “Dancing Robots”, “Human-Robot Rhythmic Interaction”, and “Robot Cross-Modal Rhythmic Perception”.

2.2.1 Dancing Robots

Nakazawa, Nakaoka *et al.*, [28], [29] presented an approach that lets a biped robot, HRP-2 (see Figure 12) imitate the spatial trajectories of complex motions of a Japanese traditional folk dance by using a motion capture system. To do that they developed the *learning-from-*observation (LFO) training method that enables a robot to acquire knowledge of what to do and how to do it from observating human demonstrations. Despite the flexibility of motion generation, a problem is that these robots cannot autonomously determine the appropriate timing of dancing movements while interacting with auditory environments, i.e., while listening to music.

Figure 12 - Humanoid robot HRP-2 (retrieved from [30]).

Tanaka *et al.* from Sony, developed the software that turns QRIO (see Figure 13) into a dancing robot which interacts with children through a posture mirroring dance mode [31], [32]. This interactive mode was developed using an Entrainment Ensemble Model which relies on the repetition of sympathy, between the robot and the child, and dynamism. To keep the synchronism they used a “Rough but Robust Imitation” visual system through which QRIO mimics the detected human movements.

Figure 13 – Sony Entertainment Robot 'QRIO' [32].

More recently, in 2007, Aucoutuier *et al.* [33] developed a robot designed by ZMP, called MIURO (see Figure 14), in which they built basic dynamics through a special type of chaos (specifically, *chaotic itinerancy* (CI)) to let the behaviour emerge in a seemingly autonomous manner. CI is a relatively common feature in high-dimensional chaotic systems where an orbit wanders through a succession of low-dimensional ordered states (or attractors), but transits from one attractor to the next by entering high-dimensional chaotic motion. The robot motor commands are generated in real time by converting the output from a neural network that processes a pulse sequence corresponding to the beats of the music.

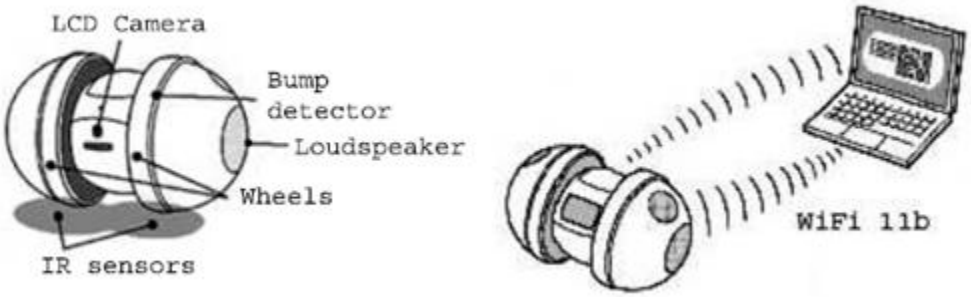


Figure 14 - The MIURO robotic platform manufactured by ZMP Inc. [33] is a two-wheeled musical player equipped with an IPod mp3 player interface and a set of loudspeakers. Wheel velocities can be controlled in real-time through wireless communication with a computer.

Burger and Bresin [34] also used the Lego Mindstorms NXT to design a robot, named M[ε]X (see Figure 15), which expresses movements to display emotions embedded in the audio layer, in both live and recorded music performance. Their robot had constraints of sensors and motors, so the emotions (happiness, anger and sadness) were implemented taking into account only the main characteristics of musicians' movements.

Figure 15 - The M[ε]X emotionally expressive robot [34].

Yoshii *et al.* [35] used Honda's ASIMO, in Figure 16, to develop a biped humanoid robot that stamps its feet in time with musical beats like humans. They achieved this by building a computational mechanism that duplicates the natural human ability in terms of associating intelligent and physical functions. The former predicts the beat times in real time for polyphonic musical audio signals. The latter then synchronizes step motions with the beat times by gradually reducing the amount of errors in intervals and timing. Their robot represents the significant first step in creating an intelligent robot dancer that can generate rich and appropriate dancing movements that correspond to properties (e.g., genres and moods) of musical pieces, in a human-like behaviour.

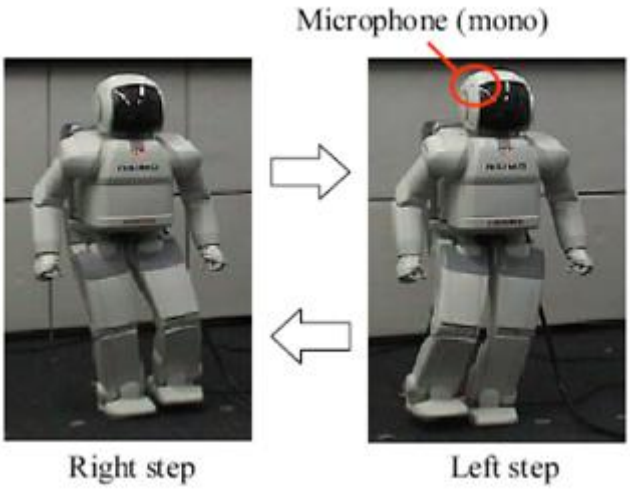


Figure 16 - Foot stamping of ASIMO robot while listening to music with its head-embedded microphone [35].

Takeda *et al.* [36] had proposed a dance partner robot referred to as MS DanceR, which has been developed as a platform for realizing effective human-robot coordination with physical interaction (see Figure 17). MS DanceR consists of an omni-directional mobile base to realize various dance steps in a ballroom dance and the force/torque sensor referred to as *Body Force Sensor* to realize compliant physical interaction between the robot and a human

based on the force/moment applied to the human. The dance partner focuses on an estimation method for dance steps, which estimates a next dance step intended by a human.

Figure 17 - The Dance Partner Robot. A force-torque sensor between the robot's upper and lower body measures the human leading force-moment. An omnidirectional mobile base uses special wheels to move along dance-step trajectories. (retrieved from [30]).

2.2.2 Human-Robot Rhythmic Interaction

Michalowski *et al.* [37], [38], [39] investigated the role of rhythm and synchronism in human-robot interactions, considering that rhythmicity is a holistic property of social interaction. To do so they developed perceptive techniques and generated social rhythmic behaviours in non-verbal interactions through dance between Keepon [37], [38] (see Figure 18a) or Roillo [39] (see Figure 18b) (a robotic virtual platform) and children.

With this research they corroborated the fundamental role of rhythm in the establishment of social interactions. They've however faced some rhythmic synchronism issues due to the delay involved in the whole process, from rhythm data reception to the correspondent action. To overcome this delay they proposed to investigate the application of prediction in their rhythm perception model.

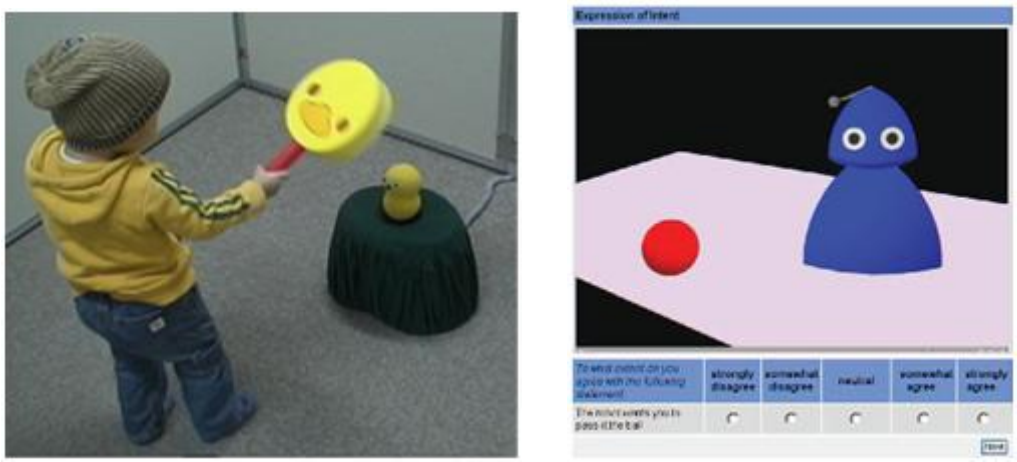


Figure 18 – Rhythm and Synchrony in human-robot interactions. **a)** Keepon dancing with a child [38]. **b)** Roillo requesting the ball using a deictic gesture [39].

Weinberg *et al.* [40], [41], developed a humanoid robot, Haile, which plays percussion instruments in synchrony with a musician (percussionist) (see Figure 19). Their robot listen this percussionist, analyses musical cues in real-time, and uses the result of it to cooperate in a rhythmic and diversified manner. To perform that they used two Max/MSP objects, bonk~ to detect the music beat onsets and pitch~ to collect pitch and timbre information from it, granting synchronous and sequential rhythmic performance.

Figure 19 - Human-robot interaction with Haile in a performance [41].

Their rhythmic interaction model is based on their rhythmic system for interdependent group collaboration, [42]. *Interconnected Musical Networks* (IMNs) are live performance systems that allow various individual players to form large-scale collaborative compositions by interdependently sharing and developing each other's motifs, in real-time.

These systems, whereas they operate in the same physical space or through WANs, provide an interdependent framework which grants the creation of social musical experiences, motivating group collaboration.

Based on this architecture Weinberg *et al* created the Beatbugs (see Figure 20a) which are hand-held percussive instruments that allow the creation, manipulation, and sharing of rhythmic motifs through a simple interface. The IMN is formed by a network of players handling each individual Beatbug (see Figure 20b). Each player can decide whether to develop the motif further (by continuously manipulating pitch, timbre, and rhythmic elements using two bend sensor antennae) or to keep it in their personal instrument (by entering and sending their own new motifs to the group.) The tension between the system's stochastic routing scheme and the players' improvised real-time decisions leads to an interdependent, dynamic, and constantly evolving musical experience.

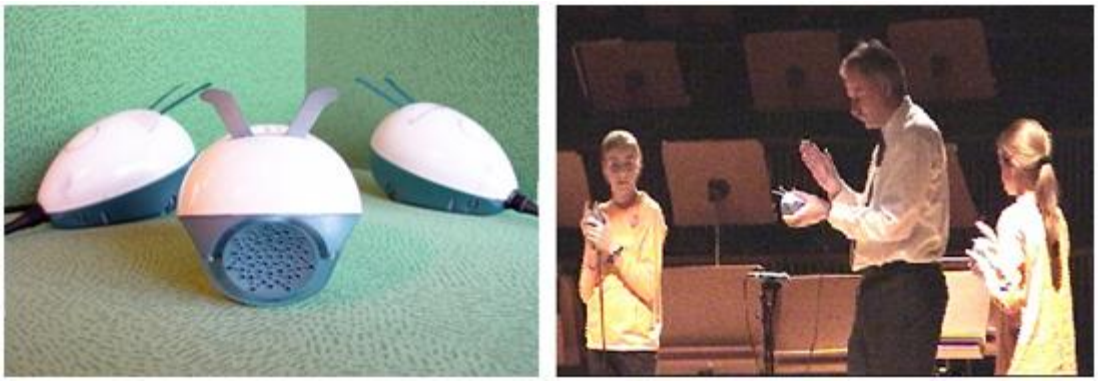


Figure 20 – The BeatBug Network [42]. **a)** Three Beatbugs. **b)** Interaction between children and a percussionist from Deutsches Symphonie-Orchester Berlin.

2.2.3 Robot Cross-Modal Rhythmic Perception

In the cross-modal rhythmic perception research I refer the work developed by Arsenio and Fitzpatrick [43], [44] and, again, Michalowski et al. [37], [38].

Based on studies realized on children, Arsenio and Fitzpatrick developed a cross-modal rhythmic perception model (from visual and audio data) to teach a robot (an upper-torso humanoid named Cog[15]) to manipulate tools and toys through demonstration (in Figure 21). Their perception model relies on detecting, segmenting, and recognizing rhythmically moving objects that generate repeated sounds as they move, at frequencies relevant for human interaction, using both visual and acoustic information.

Through the detection of visual periodic events, the robot has the ability to localize an object in its visual field and extract information concerning its trajectory over time, as well as to segment a visual representation of an object from an image – object segmentation. In addition, sound segmentation, the identification of the frequency bands that best characterize an object, was also possible from just acoustic information. At last they

[15] For more information consult R. A. Brooks, C. Breazeal, M. Marjanovic, e B. Scassellati. The Cog Project: Building a Humanoid Robot. Lecture Notes in Comp. Sci., 1562:52-87, (1999).

discussed how to reliably bind the visual appearance of objects to the sound that they generate, and to achieve selectivity.

This way they showed the necessity and advantages of using a cross-modal strategy as it manifested more robustness to disturbances either in sound or vision, and providing a better characterization of objects.

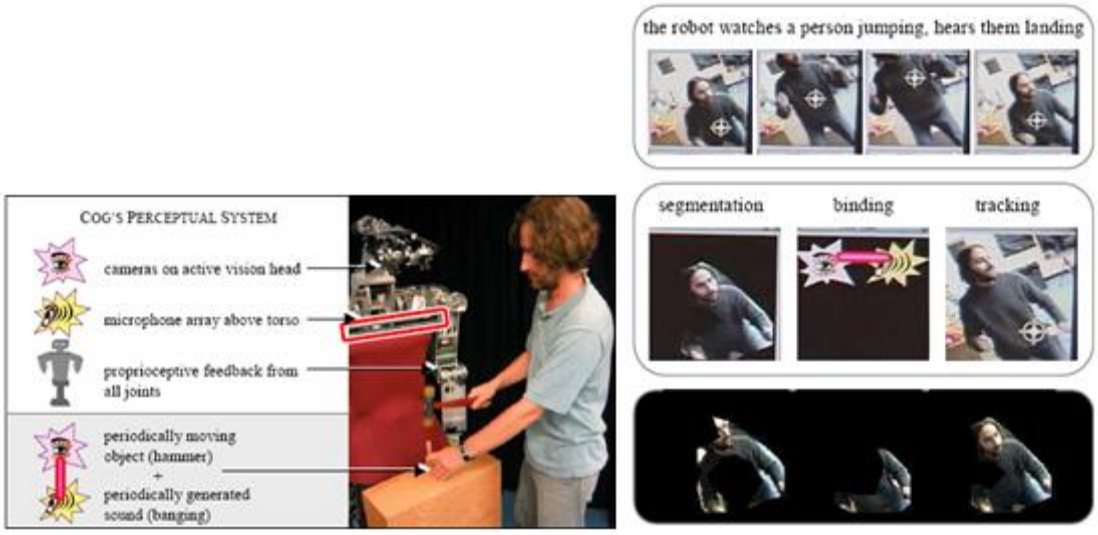


Figure 21 – The humanoid robot Cog cross-modal perception model [44]. **a)** A human demonstrates some repetitive action to the robot, such as using a hammer, while the robot watches and listens. **b)** The recorded state (middle) of the robot during an event where a human actor jumps up and down in front of the robot (up). Recorded segmentations for these experiments are shown on the lower row.

Michalowski *et al.*, as referred above, presented two complementary approaches for a cross-modal rhythmic analysis applied to their robot, Keepon, in its social interaction with children. Both of them are implemented in Max/MSP through a couple of objects that use a metronome to produce a succession of beats separated by a given time interval (granting the desired beats per minute – BPM). The frequency of the master beat to which Keepon dances can be obtained either from auditory and visual [37] or auditory and spatial (through accelerators) [38] sensing.

In [37] the movement data is, therefore, achieved by computing the average optical flow in a region of interest of an incoming video stream. In [38] they used a battery-powered three-axis accelerometer, with wireless Bluetooth data transfer, implanted in a toy, that detects rhythmic movements by finding magnitude[16] peaks, after applying a zero-crossing or low-pass filter to the retrieved data. These peaks are then treated as "beats" in the same way as musical beats or visual movement direction changes, as above. A Max/MSP object,

[16] They consider the magnitude of the overall acceleration as the Euclidian norm of the vector defined by three axes of movement (x, y, z).

sync~, receives the stream of beats and produces an oscillator that is synchronized with the tempo. This oscillator drives a stream of commands that cyclically move Keepon's bobbing and rocking degrees of freedom

At last a sequencer is used to record aligned streams of beats, sensor data, and motor commands for later playback and analysis.

2.3 A Turning Point

Most of these musical robotic applications lack however in flexibility and human control, presenting mainly reactive individual robots manifestly stiffed to their pre-programmed functions (i.e. through a fixed sequence of motor commands). Their perceptive systems are typically mere applications of existing models, with the dance movements' being rendered to a given piece of music by adapting the execution speed of the sequence to the musical tempo (automatically extracted from the audio signal).

These approaches have merits, with a notable convincing effect of synchronisation, but typically fail at sustaining long-term interest, since the dance repertoire of the robotic is rapidly exhausted and frequent patterns begin to reoccur without any variation.

Improving these absences, by developing a flexible interactive framework in which users have a deterministic role, by dynamically defining the robot choreography through selected individual dance movements, seems the perfect start to a new era on robot dancing applications.

Chapter 3

System Architecture

In this chapter we carefully described our approach by defining our system architecture. This definition is presented in three sections: *Hardware – Robot Constitution and Movement Capabilities*, *Dance Environment*, and *Software – System Framework*.

3.1 Hardware – Robot Constitution and Movement Capabilities

As already referred, we have built a humanoid-like robot using two Lego Mindstorms NXT bricks that controls six servo motors: one for each leg and each arm, one for a rotating hip and one for the head; and two sensors: a colour sensor and an ultrasonic sensor. The robot design and construction took about one week to produce, and involved the usage of almost two full Lego Mindstorms NXT base kits[17] in Lego Technic pieces.

Its final version is shown in Figure 22, and its correspondent degrees of freedom (DOFs), which shall embody a considerable diversity of dance movements, are represented in Figure 23.

[17] For the Lego Mindstorms NXT complete kit set constitution and overview (pieces and correspondent references), consult http://www.peeron.com/inv/sets/8527-1.

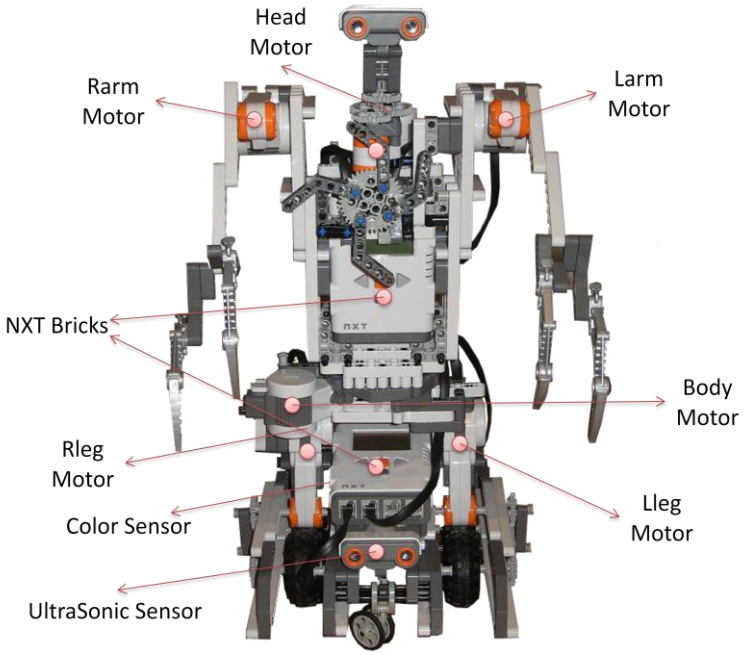


Figure 22 - Our Lego-NXT-based humanoid robot and its components.

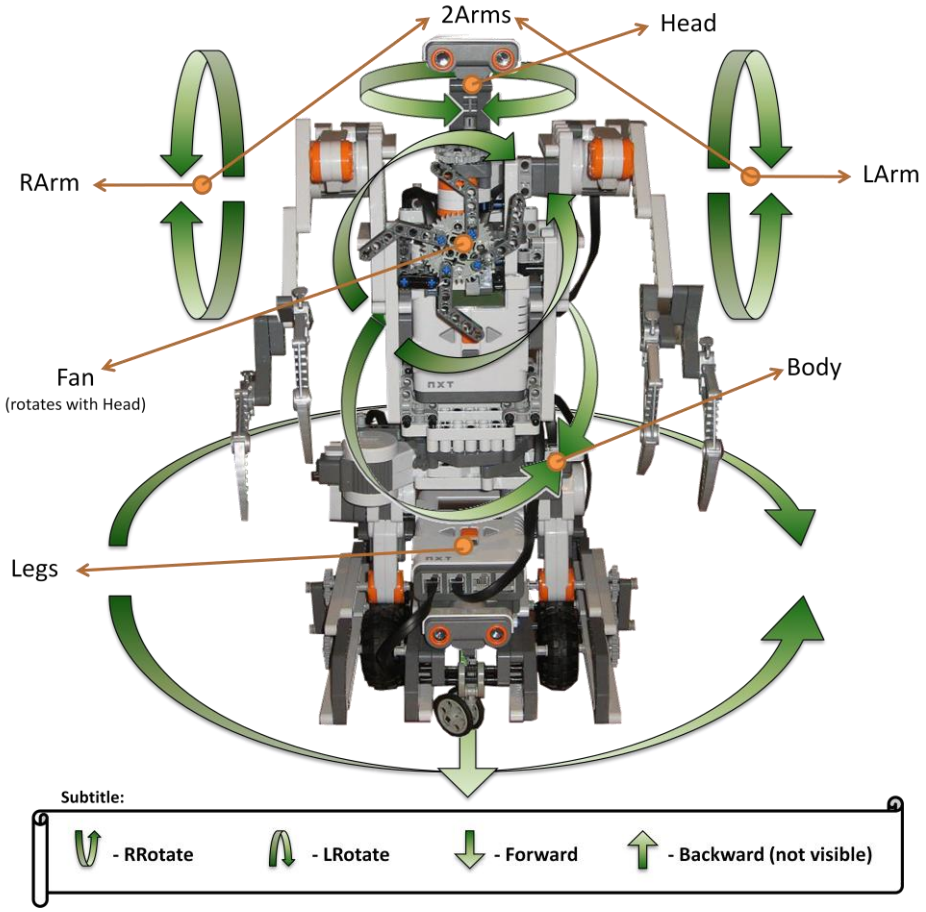


Figure 23 - The robot's degrees of freedom (DOFs) to the embodiment of dance movements.

As observable, this robot design grants 16 distinct dance movements defined as "*BodyPart-Movement*" (to the Left - *L*, Right - *R*, or one part to each side - *Alternate*): *Legs-RRotate, Legs-LRotate, Legs-Forward, Legs-Backward, Head-RRotate, Head-LRotate, Body-RRotate, Body-LRotate, RArm-RRotate, RArm-LRotate, LArm-RRotate, LArm-LRotate, 2Arms-RRotate, 2Arms-LRotate, 2Arms-RAlternate, 2Arms-LAlternate*.

Attending to Figure 22 and Figure 23, bellow we present a table which addresses each movement to its correspondent motor (actuator), direction, and the number of rotations to its completion.

Table 5 - Robot movement's description: correspondent motor(s), direction, and number of rotations to completion.

Movement	Motor	Sign (Direction)	Nr. Rotations
Legs-RRotate	RLeg	+	30
	LLeg	-	30
Legs-LRotate	RLeg	-	30
	LLeg	+	30
Legs-Forward	RLeg	+	30
	LLeg	+	30
Legs-Backward	RLeg	-	30
	LLeg	-	30
Head-RRotate	Head	+	5
Head-LRotate	Head	-	5
Body-RRotate	Body	+	3
Body-LRotate	Body	-	3
RArm-RRotate	RArm	+	5
RArm-LRotate	RArm	-	5
LArm-RRotate	LArm	+	5
LArm-LRotate	LArm	-	5
2Arms-RRotate	RArm	+	5
	LArm	+	5
2Arms-LRotate	RArm	-	5
	LArm	-	5
2Arms-RAlternate	RArm	+	5
	LArm	-	5
2Arms-Lalternate	RArm	-	5
	LArm	+	5

3.2 Dance Environment

Our dance environment (see Figure 24) was designed in order to simulate a real world environment, incorporating a multi-colour floor, which shall induce dance variations dependent on the stepped colour, and a covering wall to delimit the dancing space. The dance floor was created with four paperboards (one red, one yellow, one green, and one blue) and it was surrounded with cardboards to fulfil the walls.

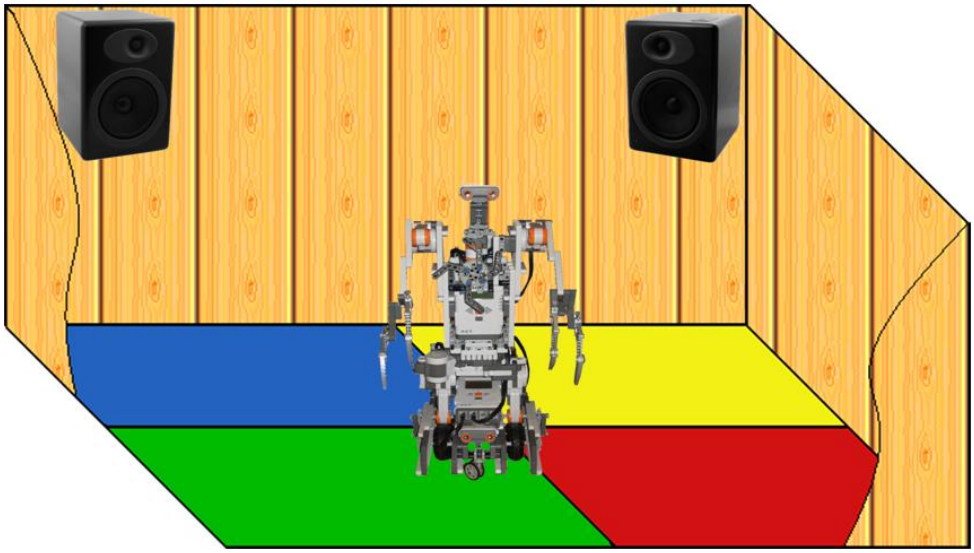


Figure 24 – Representation of the designed real world dancing environment.

This way we created a representative real world environment which interacts with the robot and grants a meaningful context to experiment the robot dancing, comparing its results with a real human performance.

3.3 Software – System Framework

Based on the presented objectives we decomposed our framework in three distinct modules, each one responsible for the treatment of specific events. All the modules are then interconnected to achieve the primary goal of robot dancing, in synchrony to the analyzed rhythm and with flexible human control. This interconnection was achieved through a multithreading processing architecture, which submits each thread (task – *Module function*) to a time-division multiplexing ("time slicing"), in very much the same way as the parallel execution of multiple tasks (computer multitasking), (see Figure 25). This structure grants the (illusion of) parallelism and synchronism between the three modules. The use of multithreading and its results shall be discussed in *4.2*.

The system architecture is then divided in the following parallel modules represented in **Figure 26**.

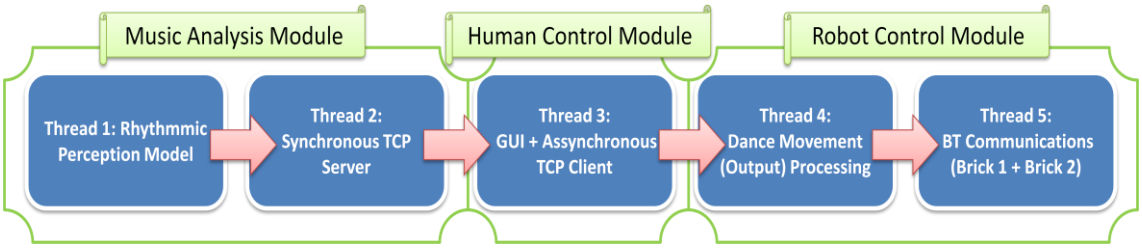


Figure 25 -Multithreading processing architecture between the three framework modules.

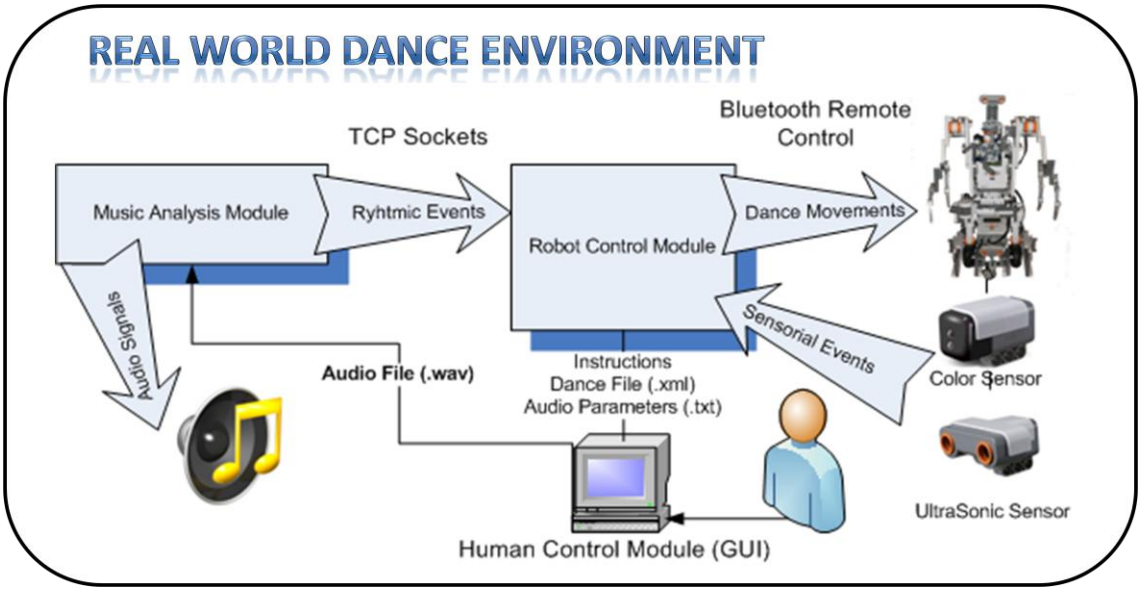


Figure 26 - System Framework: all the processes involved, and their interaction, in the achievement of a common goal - embodiment of robotic dance movements, in synchrony to the analyzed rhythm, while allowing flexible human control.

We shall now present each one of these modules: Music Analysis Module, Robot Control Module, and Human Control Module, in a separate sub-section.

3.3.1 Music Analysis Module

Music is generically an event-based phenomenon for both performer and listener, formed by a succession of sounds and silence organized in time. We nod our heads or tap our feet to the rhythm of a piece; the performer's attention is focused on each successive note [6]. In dance, body movements emerge as a natural response to music rhythmic events.

To obtain these intended events we focused our analysis on the detection of the music onset times (starting time of each musical note) through an onset detection function (a function whose peaks are intended to coincide with the times of note onsets) representing the energy variation along time, on music data composed by digital polyphonic audio signals.

This onset detection function is built through the conjunction of several specific Marsyas functional blocks/classes (MarSystems) (see Figure 27).

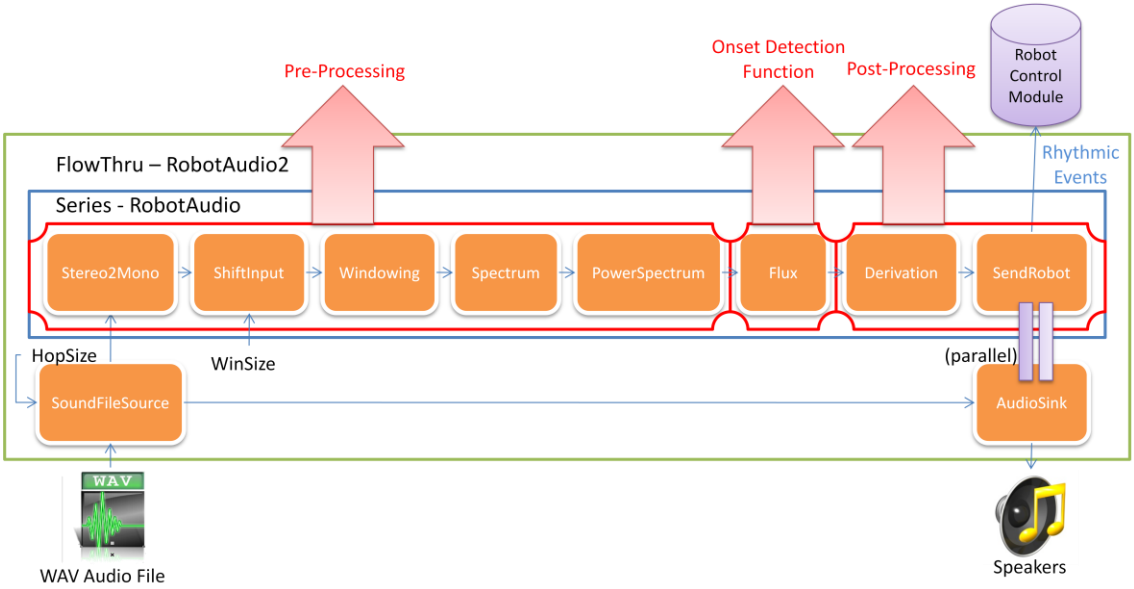


Figure 27 – Onset Detection Function represented through its Marsyas' block diagram.

An explanation of every block and composite is presented below:

- **MarSystem:** This composite represents the abstract base class for any type of system. Basically a *MarSystem* takes as input a vector of float numbers (entitled realvec) and produces a new vector (possibly with different dimensionality). Different types of computation can be used.

  MarSystems are the core processing blocks of Marsyas. All the used blocks, shown in Figure 27: *SoundFileSource*, *Stereo2Mono*, *ShiftInput*, *Windowing*, *Spectrum*, *PowerSpectrum*, *Flux*, *Derivation*, *SendRobot*, *AudioSink*; among others which involve some kind of signal transformation, are *MarSystems*.

- **Series:** This structure combines a series of *MarSystem* objects to a single *MarSystem* corresponding to executing the system objects one after the other in sequence.

- **FlowThru:** This composite combines a series of *MarSystem* objects to a single *MarSystem* corresponding to executing the system objects one after the other in sequence, but forwards the original composite input flow to the output. This structure grants that the sound file is both analyzed and reproduced, and that the *SendoRobot* (*Robot Control Module* input) and *AudioSink* (speakers input) occur at (almost) exactly the same time (marked as "parallel" in Figure 27), to assure the synchrony of the performed dance to the reproduced audio.

- **SoundFileSource:** This *MarSystem* represents the first process, which consists on reading and loading the chosen audio (WAV) file input to be further analyzed.

- **Stereo2Mono:** This class converts the stereo input file to a mono output, in order to simplify the signal processing.

- **ShiftInput:** This block overlaps each consecutive frame of the signal in order to grant a smoother analysis. Its output emerges in the form of overlapped windows of the signal input, with their size adjusted by the defined WinSize (windows size). The analysis step is called hop size and equals the frame size minus the overlap (typically 10 ms). The hop size (HopSize) defines the data granularity. In general, more overlap will give more analysis points and therefore smoother results across time, but the computational expense is proportionately greater.

- **Windowing (HW):** Windowing of a simple waveform causes its Fourier transform to have non-zero values (commonly called leakage) at frequencies other than ω. It tends to be worst (highest) near ω and least at frequencies farthest from ω. Windowing in the time domain results in a “smearing” or “smoothing” in the frequency domain.

  In this scheme was used a Hamming window (HW), due to its moderation. This window is in the family known as "raised cosine". Its equation is described as follows:

$$HW(n) = 0.53836 - 0.46164\ cos(\frac{2\pi n}{N-1}) \tag{10.1}$$

  The Hamming does not have as much side-lobe suppression as other windowing functions (like e.g. Blackman), but its main lobe is narrower. Its side-lobes “roll off” very quickly versus frequency.

- **Spectrum (S):** A periodic signal can be defined either in the time domain, as a function, or in the frequency domain, as a spectrum. In order to transform the signal from time to frequency domain a Fourier Transform (FT) shall be applied. Its discrete version is defined as follows:

$$X(f) = \sum_{n=-\infty}^{\infty} x[n]\, e^{-i2\pi \frac{f}{f_s} n} \tag{10.2}$$

  where $f = \frac{\omega}{2\pi}$ represents the frequency, ω the angular frequency, $f_s = \frac{1}{T_s}$ the sampling frequency, *t* the time domain and ***x[n]*** represents the samples of ***x(t)***, given by (the phase offset is given by ϕ):

$$x(t) = ze^{i\omega t} = z(cos(\omega t) + isen(\omega t)), \tag{10.3}$$

$$z = |z|e^{i\theta} = |z|(\cos(\theta) + isen(\theta)), \tag{10.4}$$

$$\theta = \omega t + \phi \ . \tag{10.5}$$

Therefore this block applies the Fast Fourier Transform (FFT) to compute the complex spectrum (N/2+1 points) of each input window (given by the former block). Its output is a N-sized column vector (where N is the size of the input audio vector and N/2 the Nyquist bin), using the following format:

[Re(0), Re(N/2), Re(1), Im(1), Re(2), Im(2), ..., Re(N/2-1), Im(N/2-1)]

A signal's spectrum is then displayed by a plot of the Fourier coefficients as a function of the frequency index, where the FFT is defined as:

$$S(n,k) = \sum_{m=-\frac{N}{2}}^{\frac{N}{2}-1} x(hn+m)HW(m)e^{-\frac{2\pi i}{N}mk}, \quad k=0,\dots,N-1 \tag{10.6}$$

where n is the frame number, k the frequency bin, h the hop size, and N the window size, which are parameters already defined in the former blocks.

Any aspect of the signal can now be found from this given spectrum.

- **PowerSpectrum (PS):** The power spectrum of a signal, also referred as the energy/power spectral density, represents the contribution of every frequency of the spectrum to the power of the overall signal. It is useful because many signal processing applications, such as onset detection, are based on frequency-specific modifications of signals.

This class computes then the magnitude/power of the complex spectrum (in decibels - dBs), by taking N/2+1 complex spectrum bins and processing the corresponding N/2+1 decibel's real values. It function its described as follows:

$$PS_{dB}[n] = 10log_{10}(E[n]) \tag{10.7}$$

where E[n] is the energy/power of the signal, given by:

$$E[n] = \sum_{n=n_1}^{n_2} |S(n,k)|^2 \tag{10.8}$$

Therefore, the given output data in each frame represents the power spectrum, or contribution of every frequency to the power of the original signal, for a given window.

- **(Spectral) Flux (SF):** This block represents the actual onset detection function, which is given by (2.3). As observed, in section *2.1.1.2*, the Spectral flux measures the change in magnitude in each frequency bin *k*, given by the *PowerSpectrum*, restricted to the positive changes and summed across all frequency bins.

- **Derivation (Drv):** This functional block retrieves only the crescent *Flux* output, by subtracting the *n* frame to the *n-1* one:

$$\mathrm{Drv[n]} = \mathrm{SF[n]} - \mathrm{SF[n-1]}. \tag{10.9}$$

- **SendRobot (SR):** This last block acts as our Peak Picking function and TCP client. It applies peak picking with an adaptive thresholding algorithm to distinguish three rhythm events: Strong, Medium and Soft onsets, which are then sent to the *Robot Control Module* via TCP sockets. Our peak picking (PP) always look for the highest onset detected so far through the following function:

$$\mathrm{PP[n]} = \max\left(\mathrm{PP[n-1]}, \mathrm{Drv[n]}\right), \tag{10.10}$$

assigning the PP[n] value every 5 frames. Initially, in order to normalize this process due to potential inconsistency in the beginning of some music data, the function waits 35 frames (equal to 2.43s due to fs_{Flux} = 14.36Hz) to initialize the onset detection, initializing with

$$PP[n] = \frac{1}{2} * PP[n] \tag{10.11}$$

In order to distinguish the three referred rhythmic events, our adaptive thresholding algorithm is defined as follows:

$$SR(x) = \begin{cases} Strong, & if\ x > \delta_3 \\ Medium & if\ \delta_2 < x < \delta_3 \\ Soft & if\ \delta_1 < x < \delta_2 \end{cases} \tag{10.12}$$

where,

$$\begin{cases} \delta_1 = thres_1 * PP[n] \\ \delta_2 = thres_2 * PP[n] \\ \delta_3 = thres_3 * PP[n] \end{cases}, 0 < thres_i < 1. \tag{10.13}$$

The values of $thres_1$, $thres_2$, and $thres_3$, are constants which can be dynamically assigned through the user interface (see *3.3*).

- **AudioSink:** This class consists on a real-time *audio sink* based on RtAudio, a Microsoft produced adaptive wide-band speech codec. It is responsible for sending its output to the speakers in order to reproduce the given (WAV) audio file, as music.

Given this decomposition, some last considerations shall be presented:

- **Pre-Processing:** The pre-processing (see *2.1.1.1* for clarification) of our onset detection scheme consists in the series composed by the following blocks: *Stereo2Mono*, *ShiftInput*, *Windowing*, *Spectrum*, and *PowerSpectrum*. These processes properly prepare the analyzed audio signal to be submitted to our *Spectral Flux Onset Detection Function*.

- **Onset Detection Function:** This function (see *2.1.1.2* for clarification) is fully determined by the *Flux* functional block, which measure the variation of the energy between consecutive frames, in order to detect the required onsets.

- **Post-Processing:** The post-processing (see *2.1.1.3* for clarification) is represented by the *Derivation* and *SendRobot* blocks, which are responsible for selecting the onsets from the detection function, by a peak-picking algorithm which finds local maxima in the detection function; and an adaptive thresholding algorithm, which defines the respective constraints (given by $thres_1$, $thres_2$, and $thres_3$).

All the parameter values, namely the *WinSize*, *HopSize*, *thres1*, *thres2*, and *thres3*, shall be discussed in the next chapter through some detailed experiments, presenting the respective results. These parameters can be loaded through a proper text file (.txt) for each music (WAV) file input (see next section).

As conclusion, the use of this rhythmic perception model induces our human-like robot to reactively execute proper dance movements in a time-synchronous way, but individually spontaneous, trying to simulate the dynamic movement behaviour typical from human beings.

3.3.2 Robot Control Module

Rhythm is the key component that forms the symbiotic relationship between dance and music, dating back to prehistoric times; body movements and music are closely linked in a dynamic relationship between acting and listening, cause and effect [45].

In order to perform this reaction, our *Robot Control Module*, as referred, is responsible for controlling the robot in the proper embodiment of dance movements, synchronously to the rhythmic events input received from the Music Analysis Module, and dynamically due to

the received Sensorial Events (colour and ultrasonic sensing), and pre-defined dance (see *3.3.3*).

Bellow, in Figure 28, we present a flow chart which carefully explains the entire algorithm showing all the processes involved in this module.

As observable, the robot's body movement reacts to a conjunction of stimuli formed by three rhythmic events, namely: *Low*, *Medium*, *Strong* (Onsets), or Silence; and two sensorial event groups defined by the detected colour (Colour Sensing): *Blue*, *Yellow*, *Green*, *Red*; and by the proximity to an obstacle: *Too Close*, *OK*.

All the dance movements, defined for each conjunction of rhythmic plus sensorial event, as the velocity of their independent performance, can be dynamically and flexibly assigned in the Human Control Module, through a proper user interface (see next section). Each full dance definition is saved in a proper XML file (see *Appendix C* for clarification).

The bi-directional interaction between this module and the robot is achieved via Bluetooth, thanks to the NXT Remote API (see *1.3.3*) through the following classes (see Figure 29):

- **Serial Class:** This class is responsible for the Bluetooth communication, ensuring the connection (through *connect()* function) between this module and each of the robot's NXT bricks through two defined serial COM ports.

- **Brick Class:** Through this class we can retrieve and set any information related to a NXT brick (*name*, *battery level*, *firmware version*, and *start* or *stop programs*). This class was used to retrieve the battery level (*battery_level()* function) in order to check (and consequently assure) the correct connection to each brick, in every attempt. The Bluetooth connection state is visible through the user interface (see next section).

- **Motor Class:** This class is responsible for controlling every robot motor (actuator). It sets the defined motor direction and speed (*on(speed)* function) and retrieves the number of rotations (with *get_rotation()*), in order to perform complete movements (with a beginning, when reacting to the occurrence of a specific conjunction of events; and an end, when it's reached the defined number of rotations).

- **Sonar Class:** This class is the sensor class for 9 Volts sensors like the ones we used: colour sensor and ultrasonic sensor. It retrieves the values given by each sensor (*distance()* function) in the chosen scale (centimetres for the ultrasonic distance and

colour reference for the detected colour). The ultrasonic values are given from 0-255 cm and the colour ones from 0-17 (see *appendix B* for the colour number chart).

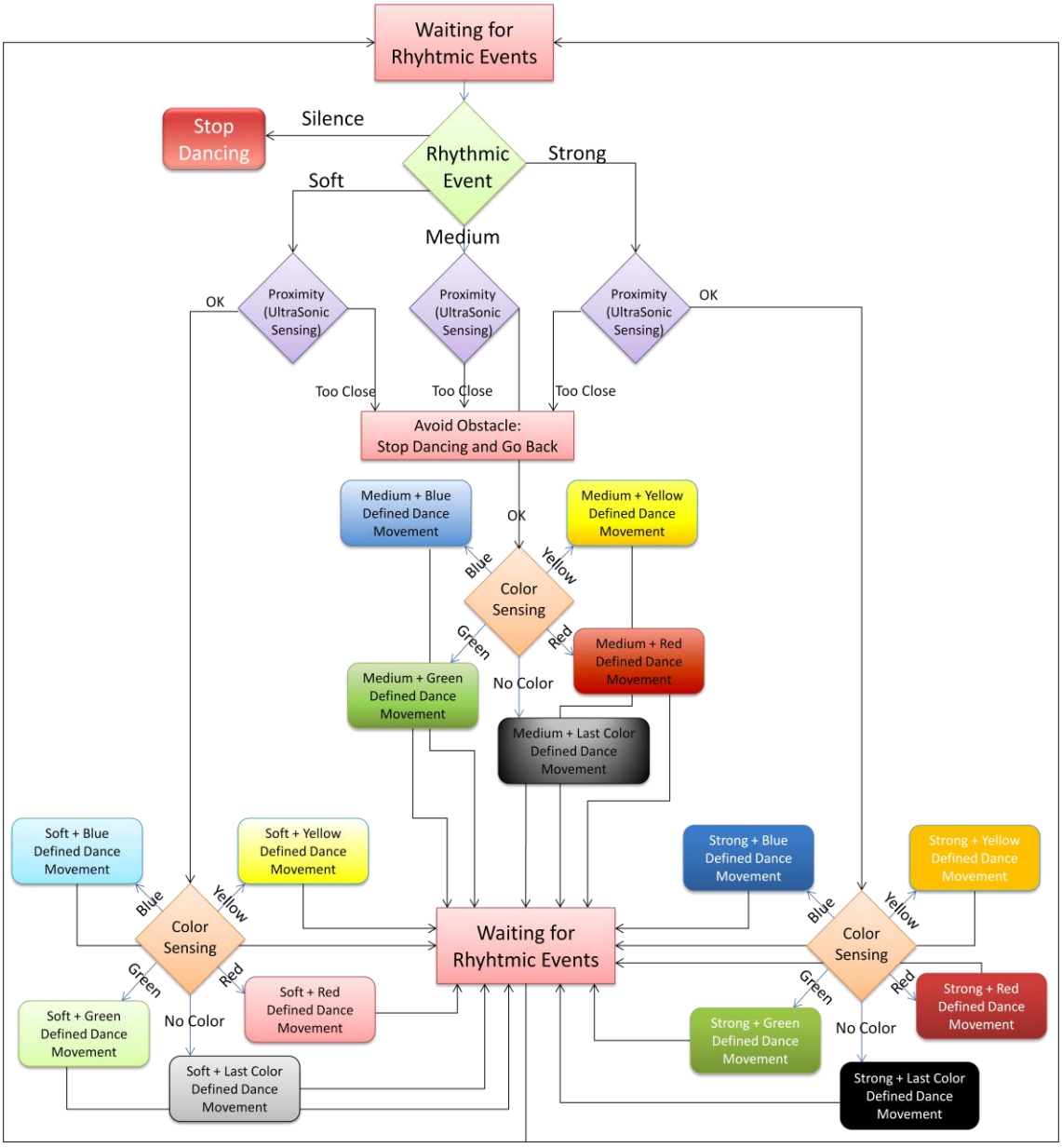


Figure 28 - *Robot Control Module* flow chart: processes and decisions involved in the robot control algorithm, inciting a different dance reaction to the occurrence of each rhythmic and sensorial event's conjunction.

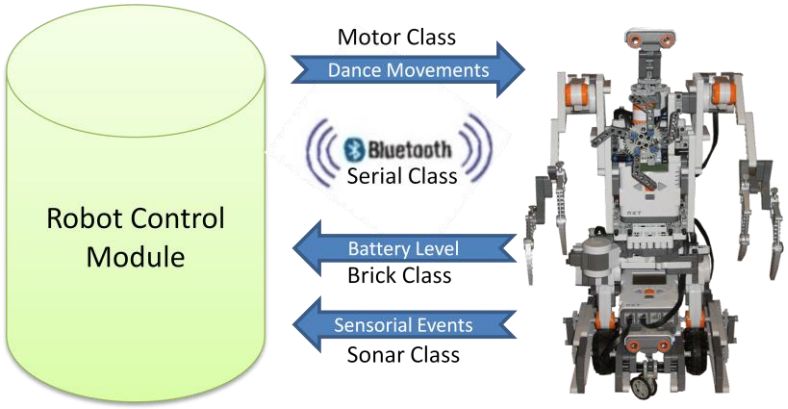


Figure 29 – Bi-directional interaction between this module and the robot, through four NXT Remote API classes: *Motor*, *Serial*, *Brick*, *Sonar*.

The presented architecture grants the aim of this module by controlling the robot in a behavioural reactive manner when submitted to a conjunction of rhythmic and sensorial events.

3.3.3 Human Control Module

To work with a system, users have to be able to control it and assess its state. A computational way to achieve this control is through a (graphical) user interface (GUI). The user interface (or human machine interface) is the aggregate of means by which people (the users) interact with the system: a particular machine, device, computer program or other complex tool. Hence, a user interface provides means of input, by allowing the users to manipulate a system; and output, by allowing the system to produce the effects of the users' manipulation.

The design of a user interface affects the amount of effort the user must expend to provide input for the system and to interpret the its output, and how much effort it takes to learn how to do this. Usability[18] is the degree to which the design of a particular user interface takes into account the human psychology and physiology of the users, and makes the process of using the system effective, efficient and satisfying.

Approaching these usability principles, we seek to assure the user interactivity by granting a flexible human control through our interface. Our GUI, as its interaction with the former modules, are presented in the figures below (Figure 30, Figure 31 and Figure 32). In consideration to them, Table 6 briefly describes each marked control component.

[18] Usability is mainly a characteristic of the user interface, but is also associated with the functionalities of the product and the process to design it. It describes how well a product can be used for its intended purpose by its target users with efficiency, effectiveness, and satisfaction, also taking into account the requirements from its context of use.

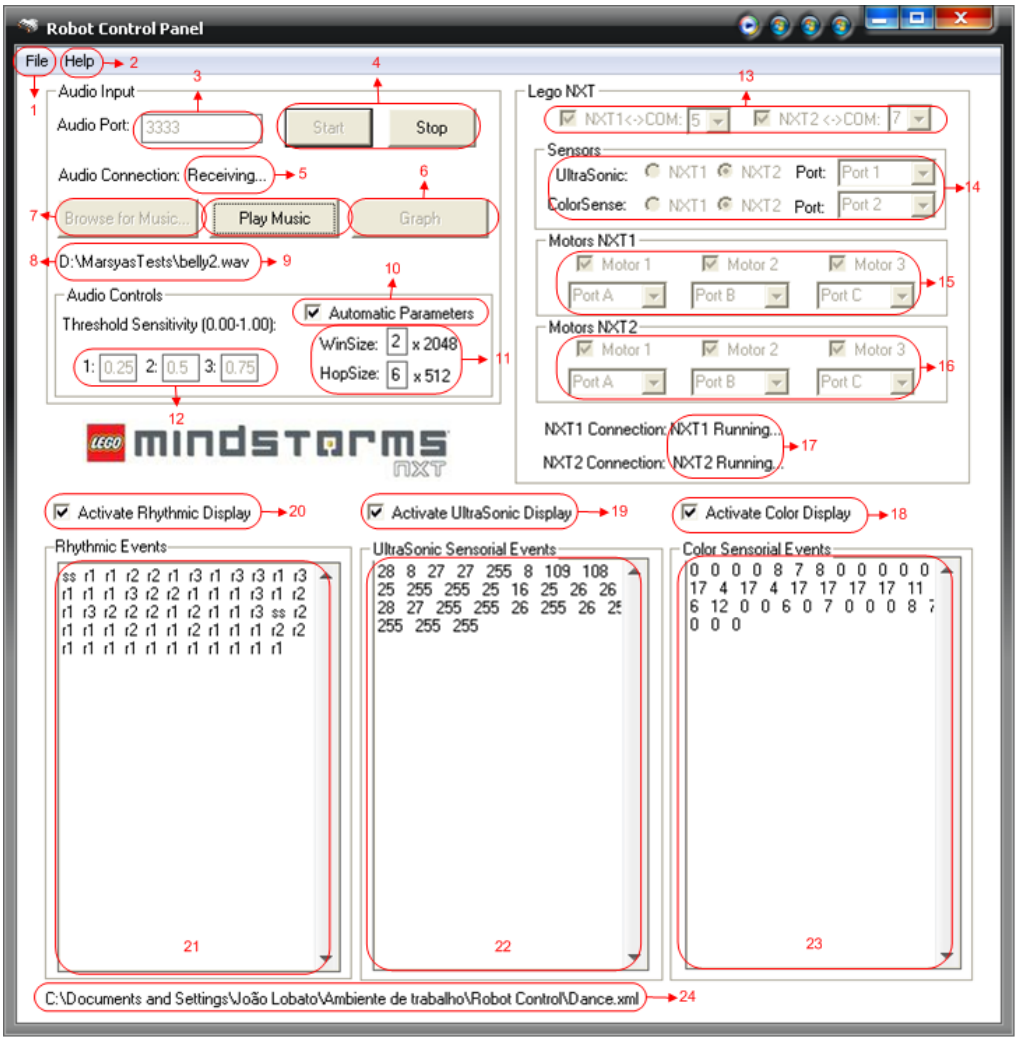


Figure 30 - Robot Control Panel GUI – this interface is responsible for all audio parameters and robot composition.

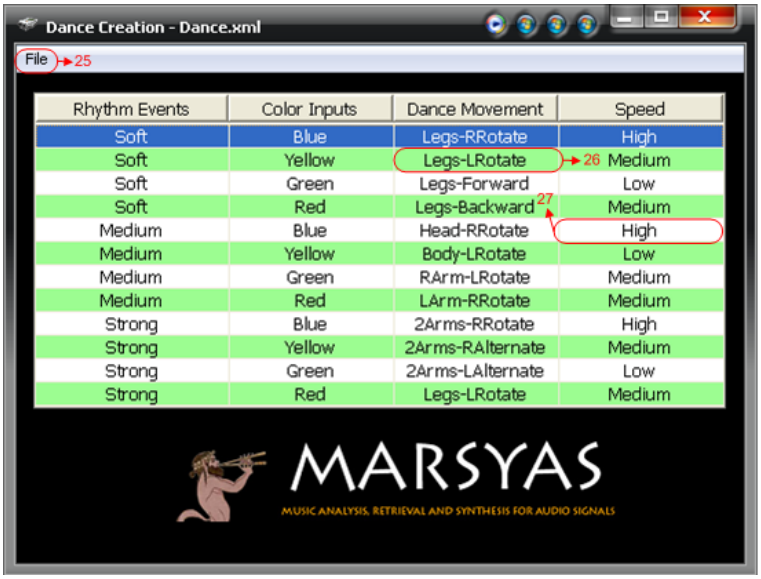


Figure 31 - Dance Creation GUI – this interface enables a dynamic full dance definition offering 16 distinct dance movements to be freely distributed through a conjunction of 12 rhythmic/sensorial events.

Table 6 - *Human Control Module* interface components description

Nr.	Control Component	Description
1	**Control Panel File Menu**	This menu is decomposed in three separators: *Load Dance File*, which opens the Windows explorer in order to choose and subsequently load a created XML dance file; *Dance Creation…*, which opens the Dance Creation interface; and *Exit*, to quit the application.
2	**GUI Help Menu**	This menu contains the *About…* separator which presents some information about the author and the software version.
3	**Audio Port**	In this text box the user can define the TCP client socket port opened by the correspondent socket server (*Music Analysis Module*), in order to transmit the audio parameters and receive the given rhythmic events. By default it was defined as 3333;
4	**Start/Stop Buttons**	These buttons are responsible for starting or stopping the robot control. The start button initializes the TCP connection with the *Music Analysis* Module and the Bluetooth connection with each checked NXT brick, loading every defined parameter. When ready the system then waits for simulation or performance (by pressing the respective button – see 6/7). The stop button stops the robot (if it was dancing), stop the connection with the *Music Analysis Module*, and clear all set parameters;
5	**Audio Connection Status**	This text control shows the status of the TCP connection to the *Music Analysis Module*.
6	**Graph Mode (Simulation)**	This button allows the simulation of the onset detection function, given the assigned parameters. This simulation is represented through a MATLAB plot (see *chapter 4*) which points out the peak-picking and each detected rhythmic event (soft, medium, or strong onset).
7	**Musical Mode (Performance)**	This button activates the actual performance of the robot, through its pre-defined dance movements in reaction to each individual conjunction of rhythmic and sensorial events.
8	**Audio (WAV) File Explorer**	This button opens the Windows explorer in order to choose the intended Audio WAV file, to be analyzed and parallelly reproduced.
9	**Audio (WAV) File Path**	This text control shows the path of the chosen audio file.
10	**Automatic Parameter Definition**	When checked this control grants that the audio parameters ($thres_1$, $thres_2$, $thres_3$, *WinSize*, *HopSize* – see *3.2.1*) are loaded from a properly defined parameter text (.txt) file. Each audio file shall have its own parameter file.
11	**WinSize and HopSize Parameters**	If the *Automatic Parameter Definition control* is unchecked the values of *WinSize* and *HopSize*, to be sent to the *Music Analysis Module*, shall be manually defined through these text boxes.
12	**Thresholding Parameters**	If the *Automatic Parameter Definition control* is unchecked the values of $thres_1$, $thres_2$, and $thres_3$, to be sent to the *Music Analysis Module*, shall be manually defined through these text boxes.
13	**Lego NXT Bricks (Check + BT COM Port)**	In this area the user can check the NXT bricks to be controlled, and define each correspondent Bluetooth serial COM port.

14	**Sensors Control (Brick + Port)**	In this area the user can define to each NXT brick and correspondent port the sensors are connected.
15	**Motors Control Brick 1 (Check + Ports)**	In this area the user can check which motors, from NXT brick 1, shall be controlled, and define the ports to which they are connected.
16	**Motors Control Brick 2 (Check + Ports)**	In this area the user can check which motors, from NXT brick 2, shall be controlled, and define the ports to which they are connected.
17	**Lego NXT Bricks Connection Status**	These text controls show the current connection status to each defined NXT brick.
18	**Colour Display Activation**	This checkbox control activates or deactivates the display of the colour sensorial events' input, received from the *Colour sensor*.
19	**UltraSonic Display Activation**	This checkbox control activates or deactivates the display of the ultrasonic sensorial events' input, received from the *UltraSonic sensor*.
20	**Rhythmic Display Activation**	This checkbox control activates or deactivates the display of the rhythmic events' input, received from the *Music Analysis Module*.
21	**Rhythmic Display**	If the *Rhythmic Display Activation checkbox* is checked, this frame displays the received rhythmic events.
22	**UltraSonic Display**	If the *UltraSonic Display Activation checkbox* is checked, this frame displays the received ultrasonic events.
23	**Colour Display**	If the *Colour Display Activation checkbox* is checked, this frame displays the received colour events.
24	**Dance File Path**	This text control shows the path of the chosen XML dance file.
25	**Dance Creation File Menu**	This menu is decomposed in three separators: *New*, which cleans the *Dance Creation* interface to the default values; *Load Dance File…*, which opens the Windows explorer in order to choose and subsequently load a created XML dance file to this interface, to be further analyzed or altered; *Save* and *Save As…*, to save the created dance in the opened dance file or in a new one.
26	**Dance Movement ComboBox**	To choose the intended dance movement from the complete list (with 16 distinct movements – see *3.1*).
27	**Movement Speed ComboBox**	To define the correspondent dance movement speed, from a three values' list: *High*, *Medium*, or *Low* speed.

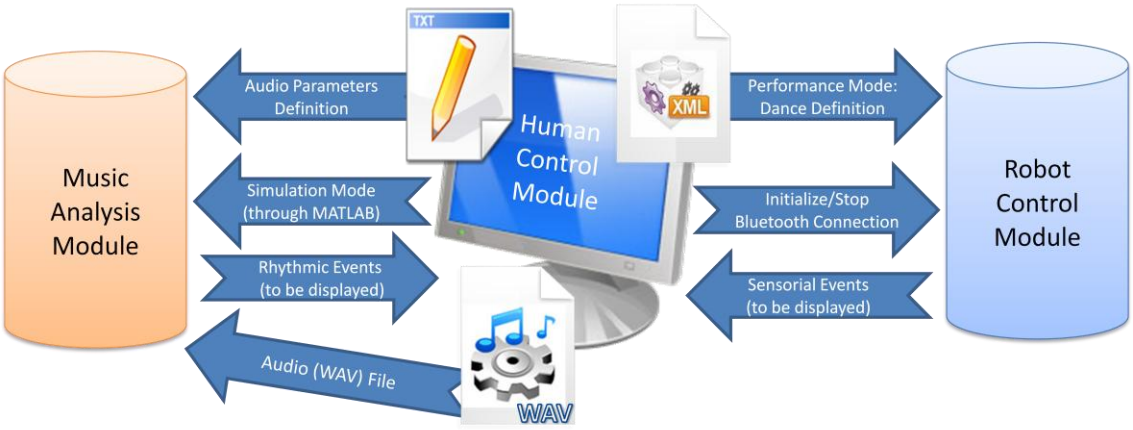


Figure 32 – *Human Control Module* (GUI) bi-directional interaction with the *Music Analysis Module* and the *Robot Control Module*.

By now analyzing Figure 32 we shall note the high-level position of this module in the (human) control of the whole framework. The user has then a deterministic role in the system behaviour, by dynamically defining the robot choreography through selected individual dance movements, and by defining the polyphonic audio data (WAV file) to be analyzed and the audio parameters which shall highly influence the resultant rhythmic perception.

3.4 Conclusions

This chapter described our robotic system, the involving dancing environment and presented an overview of the system framework decomposed in its constituent modules. Regarding this framework, it explained the role of every module in the final composition and their interconnection to the achievement of a common goal: robot dancing performance in a synchronous, dynamic and realistic manner.

Given this system architecture, in the next chapter we present some experiments and its correspondent results, in order to find the audio parameters and dance movements which best represent a real human performance.

Chapter 4

Experiments and Results

Our experiments focused on efficiency and synchronism issues related to the music onset detection and to the robot performance with proper and clear dance movements. This two experiment factors are presented in this chapter through two distinct sections: *Rhythmic Perception Regulation and Parameters Testing* and *Robot Dancing Performance*.

4.1 Rhythmic Perception Model Regulation

In this section we expose the experimentation made on the rhythmic perception algorithm through the proposed onset detection model (see *3.3.1*). We decomposed these experiments in two sub-sections: *Onset Detection Scheme Adjustments* and *Parameters Testing and Definition*. All testing was done via *Simulation Mode* (see *3.3.3*), thanks to Marsyas' MATLAB engine capabilities.

4.1.1 Onset Detection Scheme Adjustments

In order to reduce the sensitivity of our onset function to the main onsets, we started to apply a Butterworth low-pass filter to the *Flux* output, using many different coefficient values (see Figure 33 – *Filter block*).

Digital Butterworth are FIR (Finite Impulse Response) filters characterized by a magnitude response that is maximally flat in the pass-band and monotonic[19] overall. These filters sacrifice roll-off steepness for monotonicity in the pass- and stop-bands, being essentially smooth.

[19] Monotonicity is a property of certain types of digital-to-analog converter (DAC) circuits. In a monotonic DAC, the analog output always increases or remains constant as the digital input increases. Monotonicity is an important characteristic in many communications applications where DACs are used.

The gain G(ω) of an n-order Butterworth low pass filter is given in terms of the transfer function H(s) (output-input ratio ($\frac{V_o(s)}{V_i(s)}$), where s=jω) as:

$$G^2(\omega) = |H(j\omega)|^2 = \left|\frac{V_o(j\omega)}{V_i(j\omega)}\right|^2 = \frac{G_0^2}{1+\left(\frac{\omega}{\omega_c}\right)^{2n}} \quad (11.1)$$

where *n* is the order of the filter; ω_c the cutoff frequency[20], which must be a number between 0 and 1 (1 corresponds to the Nyquist frequency, π radians per sample); and G_0 the DC gain (gain at zero frequency).

Marsyas processes this filtering through a generic filter transfer function defined by the coefficients in length *n+1* row vectors *b* and *a*, with coefficients in descending powers of z, as follows:

$$H(z) = \frac{V_o(x)}{V_i(x)} = \frac{b(1)+b(2)z^{-1}+\cdots+b(n+1)z^{-n}}{1+a(2)z^{-1}+\cdots+a(n+1)z^{-n}}. \quad (11.2)$$

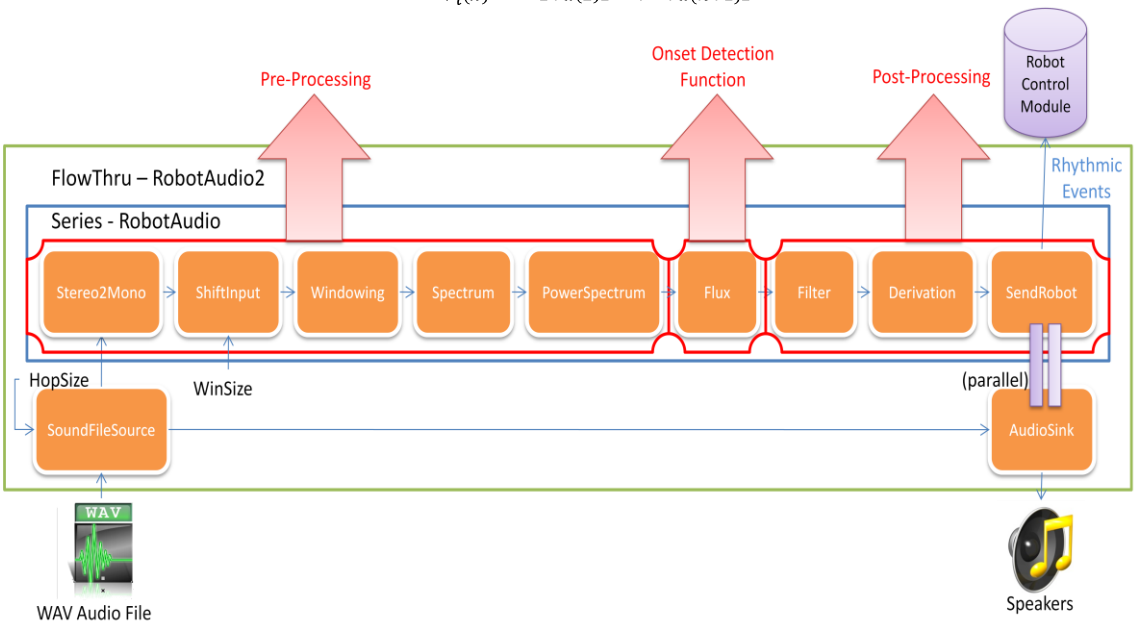


Figure 33 - Onset Detection Model with filtering (*Filter block*).

In order to retrieve the correspondent Butterworth coefficients (a_i, b_i), to each chosen *n* and ω_c values, we used the MATLAB *butter(n,ω$_c$)* function.

It can be seen, as shown in Figure 34a, that as *n* approaches infinity, the gain becomes a rectangle function and frequencies below ω_c will be passed with gain G_0, while frequencies above ω_c will be suppressed. For smaller values of *n*, the cutoff will be less sharp.

[20] Cutoff frequency is that frequency where the magnitude response of the filter is $\sqrt{1/2}$.

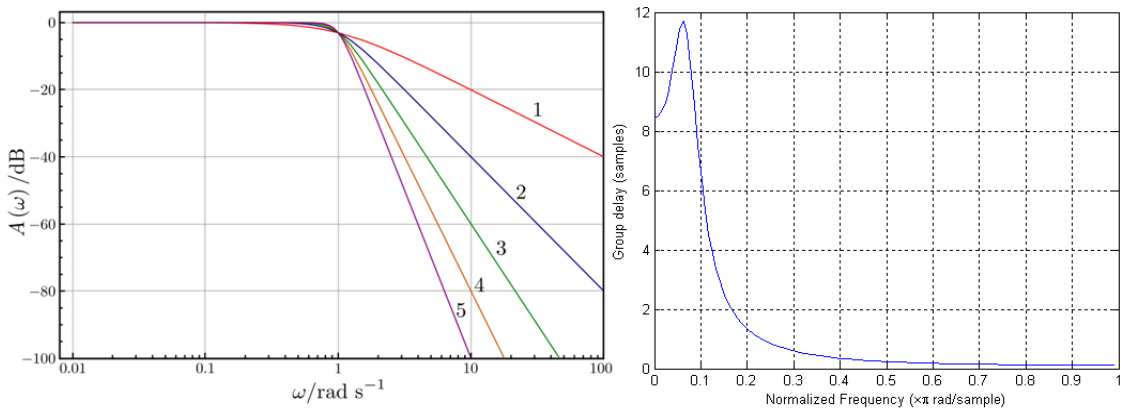


Figure 34 - Butterworth low-pass filter characteristics: a) plot of its gain for *n* = *1* to *n* = *5*. Note that the slope is 20n dB/decade where *n* is the filter order [www.wikipedia.org]. b) group delay of the filter third order (*n* = 3) with ω_c = *0.075*.

Considering these definitions, in Figure 35 we present the *Filter block* output results achieved for a different group of coefficient values, to the same music input.

As observable by increasing *n* and decreasing the cutoff frequency, ω_n, the signal became smoother, starting to isolate only the main onsets (signal peaks). This however incited a group delay[21] (see Figure 34b) that increased with this smoothing. The minimum acceptable coefficient values (ω_c = *0.075* and *n* =*3* - Figure 35b), promulgated a delay of almost 12 frames, which corresponds to approximately 0.8 seconds due to fs_{Flux} = 14.36Hz, representing, in addition to the whole process natural time consumption, a considerably high delay, facing the requirements.

In a way to bypass this issue we decided to substitute the filter with a slight increase of the window and hop size.

By experimenting different pairs of values (to the same music), as shown in Figure 36, we agreed to set these to *WinSize* = 4096 and *HopSize* = 3072. These can however be changed manually through the user interface (see *3.3.3*).

This option granted a lower sensitivity in onset detection, focusing on the more relevant ones, with no delay imposed in the process.

[21] The group delay is defined as the derivative of the phase with respect to angular frequency and is a measure of the distortion in the signal introduced by phase differences for different frequencies. It can be seen that there are no ripples in the gain curve in either the pass-band or the stop-band.

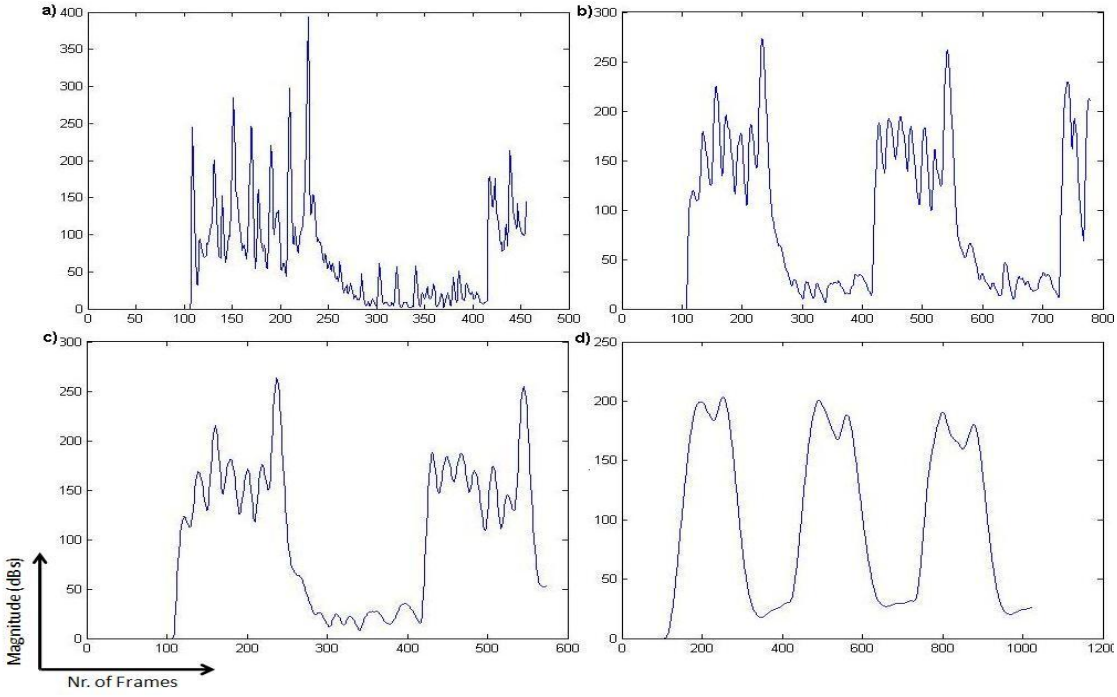


Figure 35 - Butterworth low-pass filter output for different coefficient values: a) ω_c = *0.28* and *n* =*2*. b) ω_c = *0.075* and *n* =*3*. c) ω_c = *0.075* and *n* =*4*. d) ω_c = *0.02* and *n* =*4*.

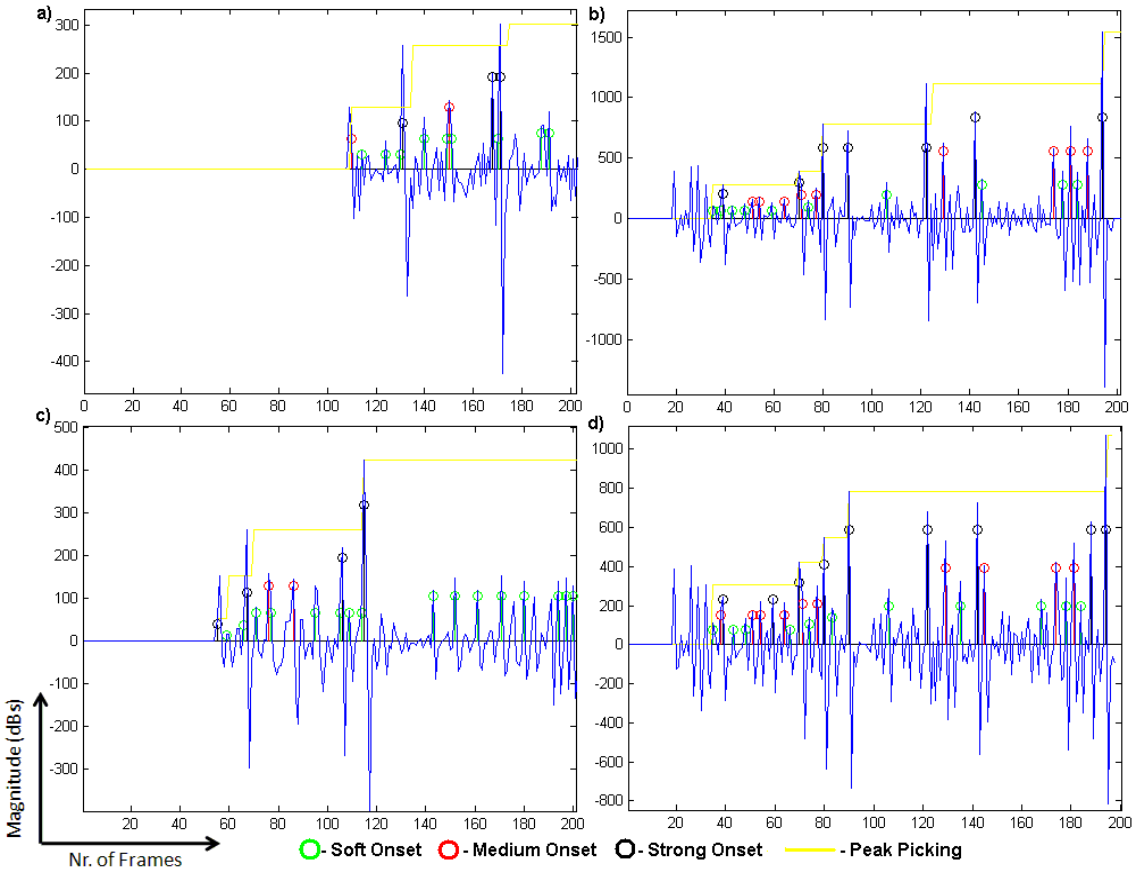


Figure 36 - Music Analysis Module output for different pairs of win size and hop size values: a) *WinSize* = *2048*and *HopSize* = *512*. b) *WinSize* = *2048* and *HopSize* = *3072*. c) *WinSize* = *4096* and *HopSize* = *1024*. d) *WinSize* = *4096* and *HopSize 3072*.

4.1.2 Parameters Testing and Definition

In this section we present our attempts to check the adequate thresholding parameters to a large set of music data.

The set of tests were performed on diverse music styles, consisting of 4 sets of short excerpts (each with around 20s) from a range of instruments, classed into the following groups [6]: NP – non-pitched percussion, such as drums; PP–pitched percussion, such as guitar; PN – pitched non-percussion, in this case some violin; and CM – complex mixtures from popular and jazz music. Below we show some screenshots, in Figure 37, and a table (Table 7) with the correspondent results.

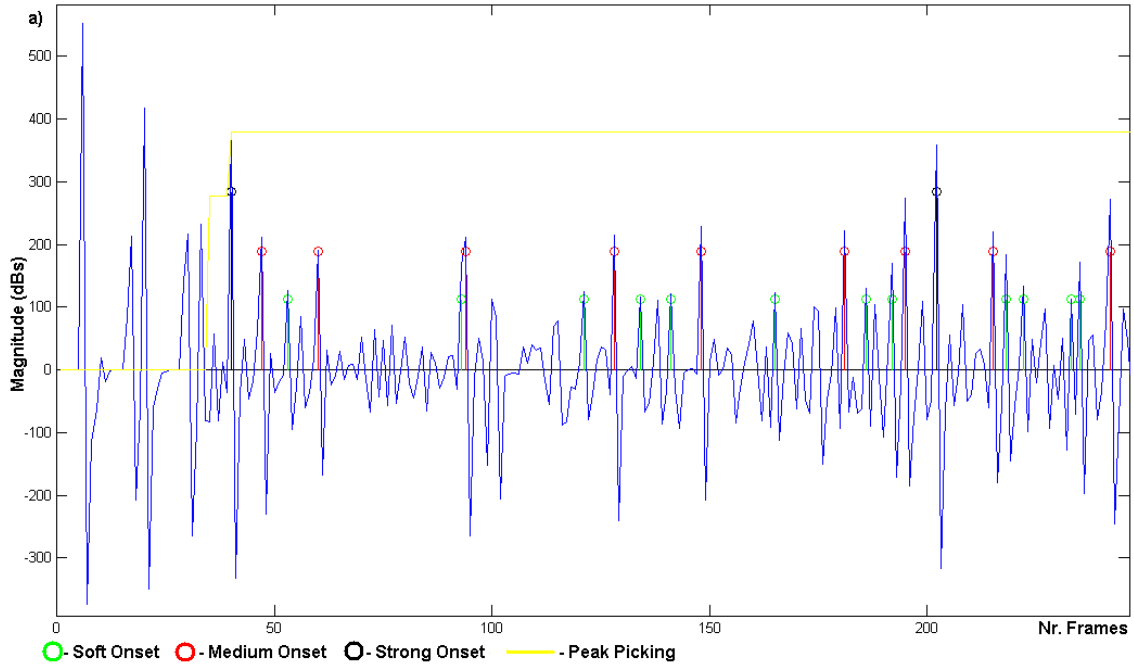
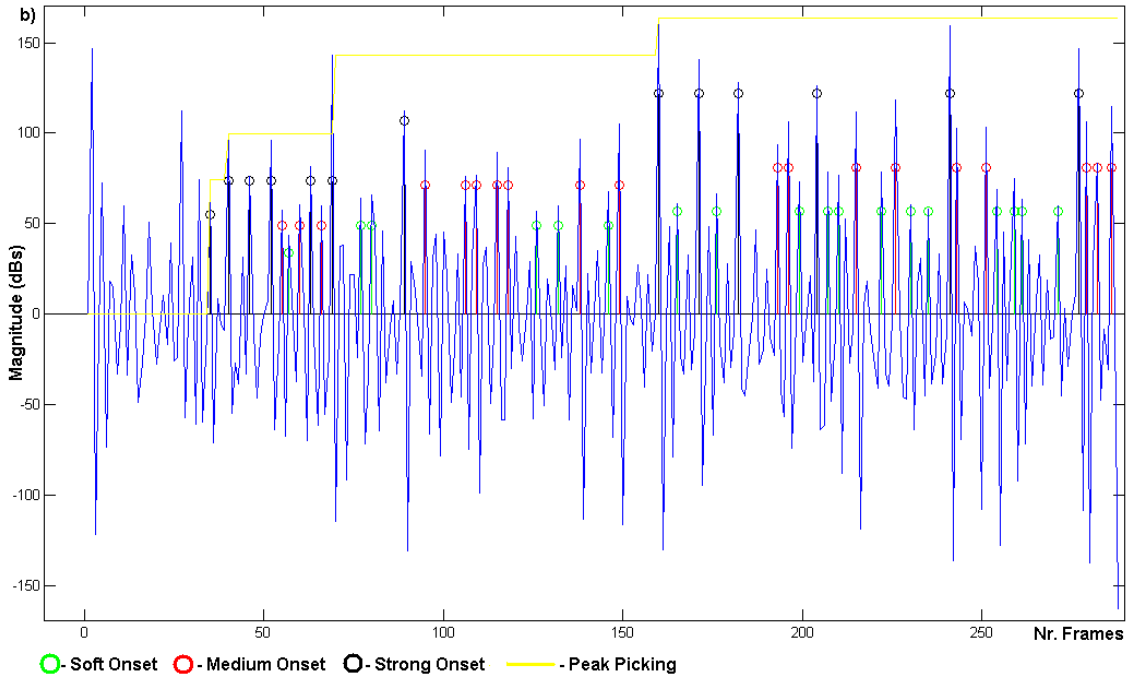
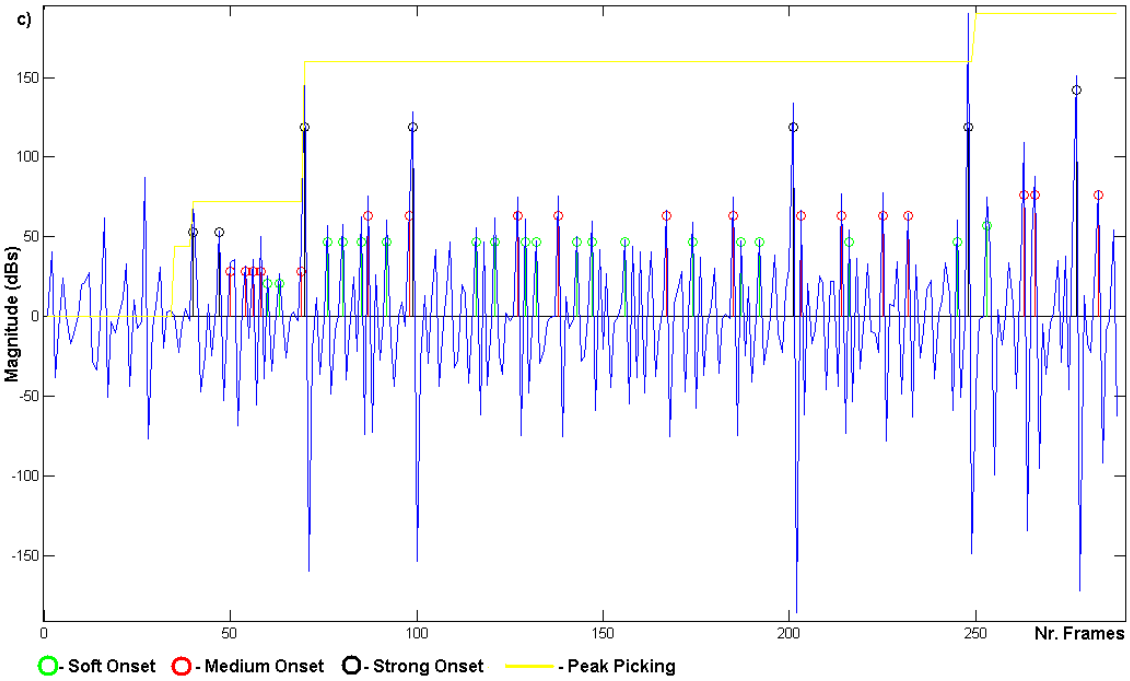


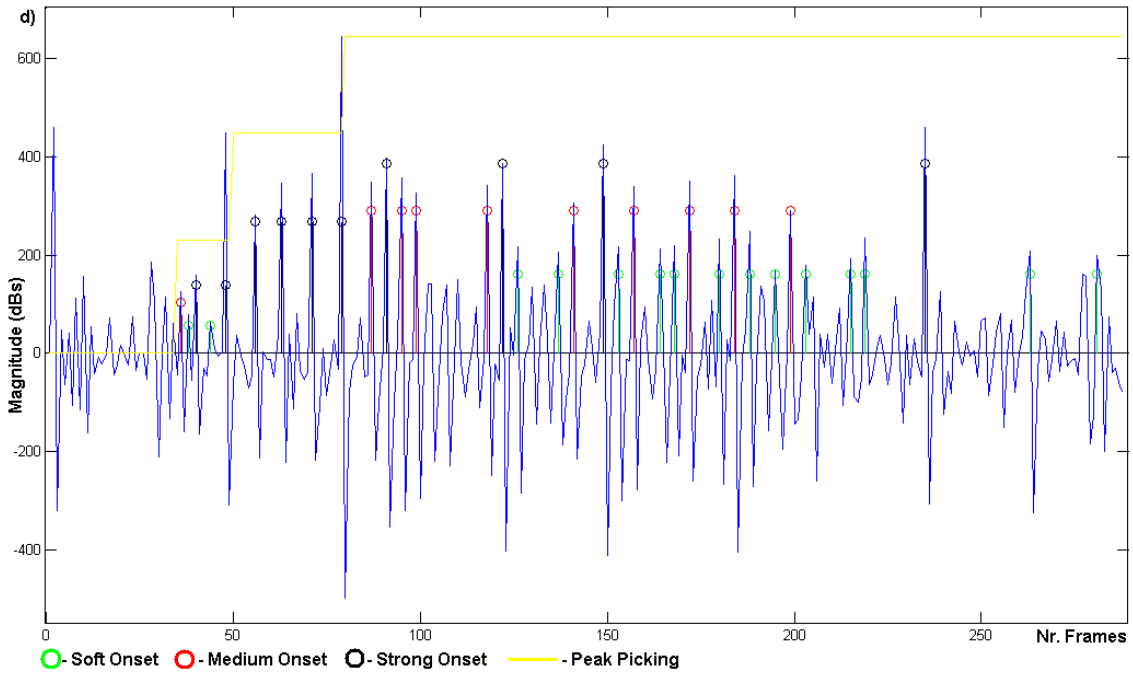
Figure 37 - *Music Analysis Module* (Onset Detection model) output: **a)** PN excerpt using $thres_1 = 0.30$; $thres_2 = 0.50$; $thres_3 = 0.75$.



b) PP excerpt using $thres_1 = 0.35$; $thres_2 = 0.50$; $thres_3 = 0.75$.



c) NP excerpt using $thres_1 = 0.30$; $thres_2 = 0.40$; $thres_3 = 0.75$.



d) CM excerpt using $thres_1 = 0.25$; $thres_2 = 0.45$; $thres_3 = 0.60$.

Table 7 - Resultant onset counting for the performed tests (in Figure 37).

Music Style	Soft Onsets	Medium Onsets	Strong Onsets	Total
PN	12	9	2	23
PP	19	18	7	44
NP	15	10	10	35
CM	18	19	13	50

Due to the observable inconsistency among the different music styles we were compelled to define different parameters for each music data. To go around this issue we created a text (.txt) file to each music file containing the respective parameters (see *3.3.3*), from where the application imports them.

4.2 Robot Dancing Performance

The analysis of the robot dancing performance was essentially based on live observation, in comparison to the meaningful data of human behaviour in a real world dance environment. We focused our analysis on *synchronism*, *dynamism*, and *realism* factors:

- **Synchronism:** due to the complexity of the algorithm and the processing and Bluetooth limitative capabilities we've verified some lacks in synchronism. This was essentially caused by the use of multithreading on a single processor (see Figure 25 for this architecture composition). The use of this architecture granted the required

simultaneity between the modules' processing, but caused some synchrony flaws due to *race condition* (dependence in a certain sequence of threads processing in order to complete a certain function) issues instigated by the complex (highly time-consumptive) task of dance movement decision and the robot Bluetooth communication overflow (as it can only receive/send data via BT in time-intervals of approximately 50-100ms and it takes around 30ms to transition from receive mode – motors, to transmit mode - sensors). In order to solve this problem we could use some multithreading synchronization objects (*Critical Sections*, *Events*, *Semaphores*, or *Mutexes*), which are used to protect memory from being modified by multiple threads at the same time (which might make the data incorrect); by assuring that each thread waits for the others' transmissions, respecting the data dependence. Yet this solution is impracticable due to the real-time requirements, which imposes that every action shall occur in a reactive manner, through a cause effect behaviour (so music can't wait for a dance decision, which on its hand cannot wait to communicate with the robot).

In future work the threading (execution of each task - *Module*) shall be replaced by a multiprocessing architecture executed in a multiprocessor or multi-core system, wherein the three modules and their recurrent processes can run literally simultaneously without suffering from synchronization issues. This shall be better exposed in *5.2 – Future Work*.

- **Dynamism:** the dynamism of our work is granted by the enormous variety of possible dance style definitions (in a total of 17^{12}-1), formed by 16 distinct individual dance movements (plus "*do nothing*") distributed through 12 (3 rhythmic events x 4 colour events) different event's conjunctions (being possible to repeat the same movement in two or more conjunctions). This dynamic behaviour is, so, transposed to the versatility of human decision, which has the power to adapt the robot performance to its own image through an interactive conduction.

  Ultimately this dynamism and interactive behaviour, in compromise with synchronism, assures an interesting relationship between a human and an artificial agent in the long-term.

  Some considerations to enhance the dynamism of our robotic system are discussed in *5.2*.

- **Realism:** realism can be defined as the fidelity to nature or to real life through representation, in adherence to the actual facts.

  Representation is a cornerstone topic in traditional AI [1]. *When intelligence is approached in an incremental manner, with strict reliance on interfacing to the real*

world through perception and action, reliance on representation disappears…due to no clean division between perception (abstraction) and reasoning [2].

Our robotic system approached this incremental intelligence through the parallel processing of different modules in the production of a reactive behaviour-based performance, which tries to replicate the human behaviour. The robot, through its dance motion, experience the world (dance environment) directly (*embodiment*), and this world directly influences its behaviour, through sensation (colour and ultrasonic sensing) (*situatedness*).

The resulting dance alternates in a seemingly autonomous manner between a diversity of motion styles coupled to the musical rhythm, and varying in consonance with the colour stepped on the dance environment.

So, despite some synchrony issues, referred above, the robot seems to react dynamically in real-time, showing a notable sense of realism.

To conclude this dissertation in the next chapter we substantiated these results in an overall review, finalizing it with some proposals to their enhancement.

Chapter 5

Conclusions and Future Work

Designing entertainment systems that exhibit such dynamic compromise between short-term synchronization and long-term autonomous behaviour is the key to maintain an interesting relationship between a human and an artificial agent, while sustaining long-term interest.

Being this the key of our work, we focused our efforts in the development of a human-interactive dancing robotic system which essentially rely on *synchronism*, *dynamism* and *realism* factors, as described in *4.2*.

Giving so, in this chapter, in *5.1 – Work Revision and Summary of Contributions*, we summarize our approach and its main results, presenting our contribution and a list of possible applications. Following, in *5.2 – Future Work* we present our proposal to enhance this framework, and motivate further research by the interactive robot dancing community.

5.1 Work Revision and Summary of Contributions

It seemed a reasonable requirement that intelligence be reactive to dynamic aspects of the environment, that a mobile robot operate on time scales similar to those of animals and humans, and that intelligence be able to generate robust behaviour in the face of uncertain sensors, an unpredictable environment, and a changing world [1].

Our idea was then to let a reactive behaviour-based system take care of the real time issues involved with interacting with the world while a more traditional AI system sits on top, making longer term executive decisions that affect the policies executed by the lower level.

In order to summarize our approach and present some global contributions of our work in contrast to the existing applications in the area, we decomposed this section on the two following sub-sections.

5.1.1 Work Summary

In this research project, we have developed a framework to make a biped humanoid Lego NXT-based robot react to music in real-time, performing dance movements in synchronism to rhythm in a dynamic and seemingly realistic autonomous way, when situated in a real world dance environment. This was achieved with a proper system architecture constituted by three modules (*Music Analysis*, *Human Control* and *Robot Control*) that communicate with each other through a multithreading architecture or via TCP sockets. The *Music Analysis Module* performs a low-level rhythmic perception based on the Marsyas onset detection function, with peak picking and adaptive thresholding. The *Human Control Module* consists on an interactive graphical interface for dance and audio parameters definitions, which grants the user deterministic role in the behaviour of this system while assuring its dynamic reaction.

The *Robot Control Module* incites the robot to react, in real-time, to the received rhythm and sensorial events, embodying the previously defined dance.

This way our robot enforces a significant step in creating an intelligent robot dancer that can generate rich and appropriate dancing movements in correspondence to the rhythm of musical pieces, and supporting human-machine interaction through dynamic dance definitions. It can also act as the motivation for other robot dancing applications or even to real human performances.

5.1.2 Summary of Contributions

The proposed research and work is enforced by its various recurrent fields of application which can be decomposed in six main areas:

- **Robotic Entertainment:** it's undeniable the increasing role of robotics in multidisciplinary entertainment areas and the great investment that is constantly done to improve the robots entertainment capabilities, due to their potential help in people's life, being further enjoyable. In this context we intended to enhance robot dancing to a new level of expressiveness and captivation by focusing on high-level human control and human-robot interaction.

  As an overall purpose we must refer some robot dancing events, namely Robot Dance contests (for example one of RoboCup's competitions), where the software can be applied as a plausible framework.

- **Rhythmic Perception and Dance Definition:** This work shall assist these study fields as a playground for professional dancers and choreographers in complete dance movement definitions and for researchers in the testing of embodied auditory perception theories.

- **Education:** from an educational point of view our goal was to create an intuitive environment for learners and children being able to experiment the creation of perceptual dancing behaviours, by defining behavioural-based responses through dance movements. The proposed interdisciplinary pedagogy shall address aspects such as rhythm, beat, dance, embodied movements, programming, and interaction, among others. We hope that through the medium of music and dance we can attract students from diverse backgrounds, who are not regularly drawn to fields such as mathematics, physics, computation, and robotics, allowing them to move across modalities and media, and between action and symbolic expression.

- **Research:** In the research community our work can assume a starting point as an interactive framework for robot dancing applications, and motivate others to follow this step. Some thoughts on further work are presented in *5.2*.

- **Open-Source:** It's wise to publish our application as an open-source software to better support the referred above applications, and as a collaboration to the development of a more robust and efficient framework for robot dancing applications.

5.2 Future Work

By keeping the incremental intelligence, as approached, we presented our future work through six modules that should be achieved subsequently, while maintaining the reactive behavioural-based architecture. Each module should consist on an activity producing subsystem which incorporate its own perceptual, modelling, and planning requirements, individually connecting sensing to action, and parallelly processed, in a multi-processing (with one processor per model) architecture, in order to produce a global behaviour.

These ideas shall enhance our actual framework to ultimately achieve a flexible framework which grants full human control through a cooperative multi-robot dancing system with multi-level cross-modal rhythmic perception capabilities.

- **Module 1 - Individual Robot Dancing Using a Low-Level Rhythmic Audio Perception Model:** This module corresponds to the development of a low-level rhythmic perception computational model, to detect fundamental musical aspects such as note onsets, beats, pitch, amplitude and timbre, from complex music data composed by polyphonic audio signals; and the correspondent actuators to the embodiment of synchronous dance movements. The perception model may consist on functions to perform onset feature detection, and ascertain pitch, timbre and volume information, as well as an intelligent function to perform beat prediction, based on beat induction and tracking algorithms. The robot motion actuators (motors) shall be directly coupled to this model (without recurring to a central system), consummating a local cognition module.

  This module constitutes the framework basis and it's a natural extent to the current developed work.

- **Module 2 - Implementation of a High-Level Rhythmic Audio Perception Model:** This module is constituted by music analysis algorithms for high-level rhythmic aspects retrieval, such as similarity, genre and mood, and a correspondent increase in the robot movement capabilities. This high-level analysis correlates pre-detected (in Module 1) low-level rhythmic features.

  Besides improving the robot rhythmic perceptive capabilities, this module might grant automatic music style identification.

- **Module 3 - Implementation of a Spatial Rhythmic Perception Model:** This module corresponds to the development of a rhythmic perception computational model for extracting rhythmic patterns from dance movement performances, through beat interval and pattern length retrieval, based on spatial and spectral algorithms.

  The extraction of these relevant motion (spatial) data can be obtained through visual or acceleration sensing. The second shall be the chosen approach as it presents both finer spatial granularity and higher temporal resolution.

- **Module 4 - Cross-Modulate Audio and Spatial Rhythmic Perception:** The cross-modulation of audio and spatial perception should enhance the robotic system's rhythmic perception, manifesting a human-like sensing to multi-modal stimuli present in a real world environment.

- **Module 5 - Implementation of Multi-Robot Cooperative Dancing:** Cooperative dancing between robots fulfils the issue on robot dancing, replicating the human interaction through dance. This module may be developed using a proper entrainment

model, based on pattern and imitation principles, and founded on swarming AI methodologies and techniques.

- **Module 6 – Development of a Flexible Interactive User Interface:** This module may consist on the development of a proper user interface (GUI), based on the one already developed, to allow human control in the robotic system through each dance definition. This shall be the application's highest level, managing all the implemented modules.

In the following diagram (Figure 38) we can visualize the whole general idea, as a clarification of this proposed framework. It is observable the incremental capabilities of this system, achieved through the subsequent incorporation of each module and their interconnection, while keeping the parallel processing in a multi-processor architecture.

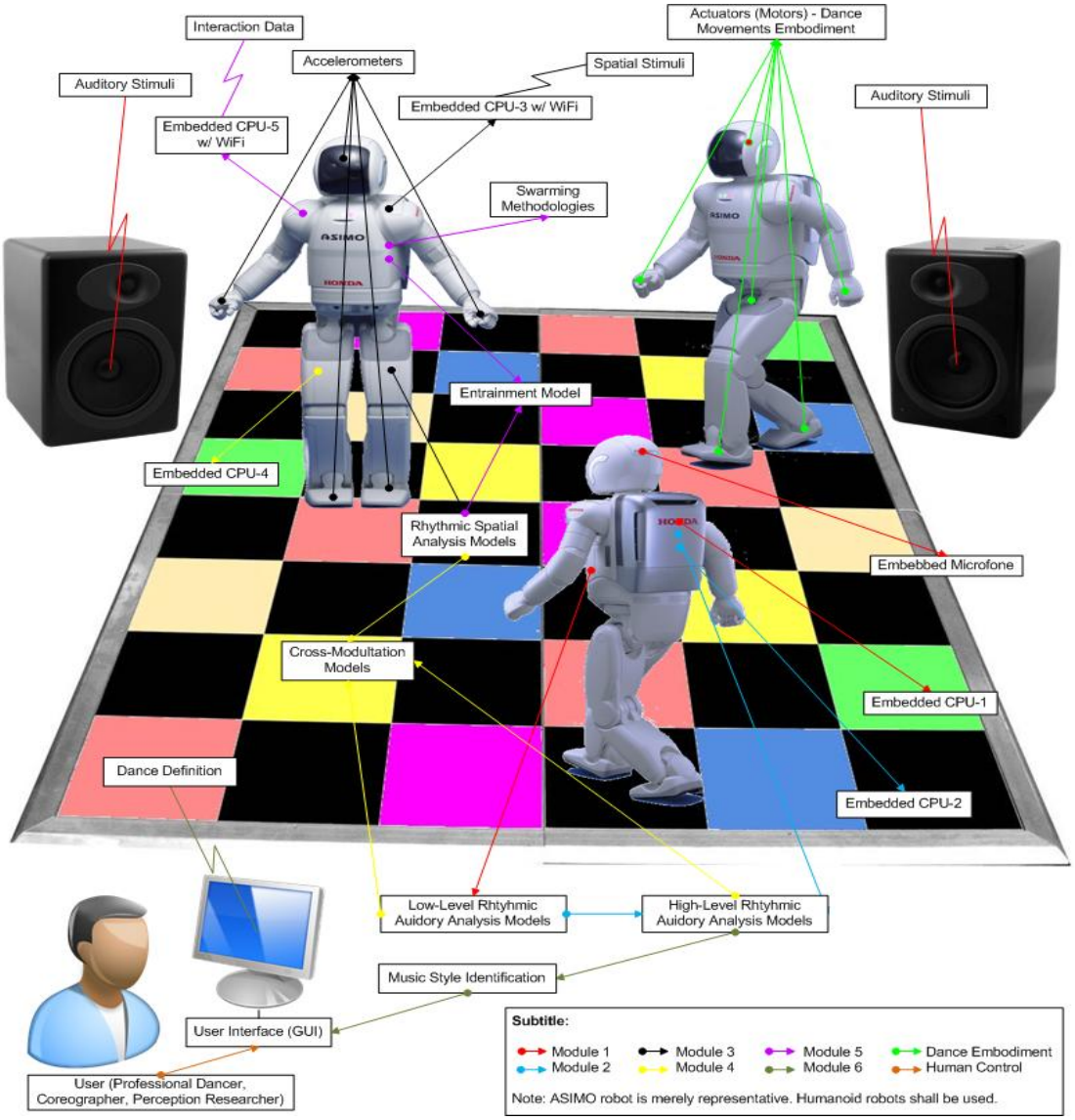


Figure 38 - Future work global idea, incorporating all the proposed modules and their interconnection.

Appendix A

Related Publications

In this annex, we provide a list of publications of relevance to this dissertation in which the author has participated. Abstracts and electronic versions of these publications are available at http://paginas.fe.up.pt/~ee03123/

[1] Oliveira, J. Gouyon, F. Reis, L. P. Robot Dance based on Online Automatic Rhythmic Perception : Submitted to Iberamia 2008, Lisbon, Portugal, 2008.

[2] Oliveira, J. Gouyon, F. Reis, L. P. Towards an Interactive Framework for Robot Dancing Applications : Submitted to Artech 2008, Porto, Portugal, 2008.

Appendix B

Colour Sensor

The NXT Colour Sensor, designed by HiTechnic[22], operates by using three different colour light emitting diodes (LED) to illuminate the target surface and measures the intensity of each colour reflected by the surface. Using the relative intensity of each colour reflection, the colour sensor calculates a Colour Number that is returned to the NXT program.

The Colour Sensor connects to an NXT sensor port using a standard NXT wire and uses the digital I2C communications protocol. The Colour Number calculated by the sensor is refreshed approximately 100 times per second.

Bellow we present the colour number chart which shows the relationship between the target colour and the colour number returned by the Colour Sensor.

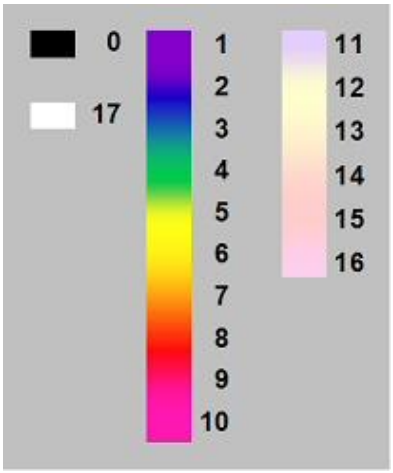


Figure 39 – HiTechnic Colour Sensor number chart.

[22] For more information visit http://www.hitechnic.com/.

Appendix C

XML Dance File Structure

In this annex we present the structure of our XML dance file, which saves the dance choreographies created by the user, through the *Human Control Module*:

```
<?xml version="1.0" encoding="utf-8" ?>
<Dance>
    <Rhythmic_Event colour="Colour">
        <movement>dance_movement</movement>
        <speed>speed</speed>
    </Rhythmic_Event >
</Dance>
```

References

[1] **Brooks, R.** *New Approaches to Robotics.* s.l. : Science, Volume 253, Issue 5025, pp. 1227-1232, 1991.

[2] **Brooks, R..** *Intelligence Without Representation.* s.l. : Artificial Intelligence, 47(1-3), pp. 139-159, 1991.

[3] **Scheirer, E.** *Music-Listening Systems.* s.l. : PhD Thesis, MIT, Cambridge, 2000.

[4] **Desain, P and Honing, H.** *Music, Mind, and Machine: Studies in Computer Music, Music Cognition, and Artificial Intelligence.* s.l. : Amsterdam: Thesis Publishers, 1992.

[5] **Dixon, S.** *Onset Detection Revisited.* s.l. : In Proc. of the Int. Conf. on Digital Audio Effects, Septembre 18–20, Montreal, Quebec, Canada, pp. 133–137, 2006.

[6] **Bello, J. P., Daudet, L., Abdallah, S., Duxbury, C., Davies, M. and Sandler, M.** *A Tutorial On Onset Detection in Musical Signals.* s.l. : IEEE Transactions on Speech and Audio Processing, vol. 13, no. 5, pp. 1035–1047, 2005.

[7] **Duxbury, C., Bello, J. P., Davies, M. and Sandler, M.** *Complex Domain Onset Detection for Musical Signals.* London, UK : In Proceedings of 6th International Conference on Digital Audio Effects (DAFx '03), 2003.

[8] **Masri, P.** *Computer Modeling of Sound for Transformation and Synthesis of Musical Signal.* s.l. : University of Bristol, Bristol, UK, 1996. Ph.D. thesis.

[9] **Bello, J.P. and Sandler, M.** *Phase-Based Note Onset Detection for Music Signals.* s.l. : Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP-03), Hong Kong. April 6-10, 2003.

[10] **Bello, J.P., Duxbury, C., Davies, M. and Sandler, M.** *On The Use of Phase and Energy for Musical Onset Detection In The Complex Domain.* s.l. : IEEE Signal Processing Letters. 11(6), pp. 553-556, June, 2004.

11] **Daudet, L.** *Transients Modeling by Pruned Wavelet Trees.* s.l. : in Proc. International Computer Music Conference (ICMC'01), Havana, Cuba, 2001.

[12] **Jehan, T.** *Musical Signal Parameter Estimation.* s.l. : M.S. thesis, Univ. of California, Berkeley, CA, 1997.

[13] **Abdallah, S. and Plumbley, M.** *Probability as Metadata: Event Detection in Music Using ICA as a Conditional Density Model.* s.l. : in Proc. 4th Int. Symp. Independent Component Analysis and Signal Separation (ICA2003), Nara, Japan, pp. 233–238, 2003.

[14] **Kauppinen, I.** *Methods for Detecting Impulsive Noise in Speech and Audio Signals.* s.l. : in Proc. 14th Int. Conf. Digit. Signal Process. (DSP2002), vol. 2, Santorini, Greece, Jul, pp. 967–970, 2002.

[15] **Duxbury, C., Sandler,M. and Davies, M.** *A Hybrid Approach to Musical Note Onset Detection.* s.l. : in Proc. Digital Audio Effects Conf. (DAFX,'02), Hamburg, Germany, pp. 33–38., 2002. pp. 33–38.

[16] **Scheirer, E.** *Tempo and Beat Analysis of Acoustic Musical Signals.* s.l. : Journal of the Acoustical Society of America, 103(1):588–601, 1998.

[17] **Klapuri, A.** *Sound Onset Detection by Applying Psychoacoustic Knowledge.* s.l. : In Proc. IEEE Int. Conf. Acoust., Speech, and Signal Proc. (ICASSP), pages 3089– 92, 1999.

[18] **Klapuri, A., Eronen, A. and Astola, J.** *Analysis of the Meter of Acoustic Musical Signals.* s.l. : IEEE Transactions on Audio, Speech and Language Processing, vol. 14, no. 1, pp. 342–355, 2006.

[19] **Duxbury, C., Bello, J. P., Davies, M. and Sandler, M.** *A Combined Phase and Amplitude Based Approach to Onset Detection for Audio Segmentation.* s.l. : Proceedings of the 4th European Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS-03), London, UK. April 9-11, 2003.

[20] **Davy, M. and Godsill, S.** *Detection of Abrupt Spectral Changes Using Support Vector Machines: An Application to Audio Signal Segmentation.* Orlando, Fla, USA : in Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '02), vol. 2, pp. 1313–1316, Orlando, Fla, USA, May, 2002.

[21] **Kapanci, E. and Pfeffer, A.** *A Hierarchical Approach to Onset Detection.* s.l. : in Proceedings of the International Computer Music Conference (ICMC '04), Miami, Fla, USA, October, 2004.

[22] **Marolt, M., Kavcic, A. and Privosnik, M.** *Neural Networks for Note Onset Detection in Piano Music.* s.l. : in Proceedings of the International Computer Music Conference (ICMC '02), Gotenborg, Sweden, September , 2002.

[23] **Lacoste, A. and Eck, D.** *Onset Detection With Artificial Neural Networks.* s.l. : MIREX2005 note onset detection contest, 2005.

[24] **Lacoste, A. and Eck, D..** *A Supervised Classification Algorithm for Note Onset Detection.* s.l. : EURASIP Journal on Applied Signal Processing, 2007(ID 43745):1-13, 2007.

[25] **Collins, N.** *A Comparison of Sound Onset Detection Algorithms with Emphasis on Psychoacoustically Motivated Detection Functions.* s.l. : in 118th Convention of the Audio Engineering Society, Barcelona, Spain, 2005.

[26] **Downie, J.S., West, K., Ehmann, A. and Vincent, E.** *The 2005 Music Information Retrieval Evaluation Exchange (MIREX 2005): Preliminary Overview.* s.l. : in 6th International Conference on Music Information Retrieval, pp. 320–323, 2005.

[27] **Leveau, P., Daudet,L. and Richard, G.** *Methodology and Tools for The Evaluation of Automatic Onset Detection Algorithms in Music.* s.l. : in 5th International Conference on Music Information Retrieval, 2004, pp. 72–75, 2004.

[28] **Nakazawa, A., Nakaoka, S., Ikeuchi, K. and Yokoi, K.** *Imitating Human Dance Motions through Motion Structure Analysis.* s.l. : IROS, pp. 2539–2544, 2002.

[29] **Nakaoka, S., et al.** *Learning from Observation Paradigm: Leg Task Models for Enabling a Biped Humanoid Robot to Imitate Human Dance.* s.l. : Int'l J. Robotics Research, vol. 26, no. 8, pp. 829–844, 2007.

[30] **Aucouturier, J.-J.** *Cheek to Chip: Dancing Robots and AI's Future.* s.l. : In IEEE Intelligent Systems, March/April 2008 (Vol. 23, No. 2), pp. 74-84 , 2008.

[31] **Tanaka, F., Suzuki, H.** *Dance Interaction with QRIO: A Case Study for Non-boring Interaction by using an Entertainment Ensemble Model.* s.l. : Proceedings of the 2004 IEEE International Workshop on Robot and Human Interactive Communication (RO-MAN), pp. 419-424, Kurashiki, Japan, 2004.

[32] **Tanaka, F., Fortenberry, B., Aisaka, K., Movellan, J.** *Plans for Developing Real-time Dance Interaction between QRIO and Toddlers in a Classroom Environment.* s.l. : Proceedings of 2005 4th IEEE International Conference on Development and Learning (ICDL), pp. 142-147, Osaka, Japan, 2005.

[33] **Aucouturier, J.-J. and Ogai, Y.** *Making a Robot Dance to Music Using Chaotic Itinerancy in a Network of FitzHugh-Nagumo Neurons.* s.l. : Proceedings of the 14th International Conference on Neural Information Processing (ICONIP), Kitakyushu, Japan , 2007.

[34] **Burger, B. and Bresin, R.** *Displaying Expression in Musical Performance by Means of a Mobile Robot.* s.l. : In Paiva, A., Prada, R., & Picard, R. W. (Eds.), Affective Computing and Intelligent Interaction, pp. 753-754, Berlin / Heidelberg: Springer, 2007.

[35] **Yoshii, K., Nakadai, K., Torii, T., Hasegawa, Y., Tsujino, H., Komatani, K., Ogata, T. and Okuno, H.** *A Biped Robot that Keeps Steps in Time with Musical Beats while Listening to Music with Its Own Ears.* s.l. : Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-2007), 1743-1750, IEEE, RSJ, San Diego, 2007.

[36] **Takeda, T. Hirata, Y. and Kosuge, K.** *Dance Step Estimation Method Based on HMM for Dance Partner Robot.* s.l. : IEEE Trans. Industrial Electronics, vol. 54, no. 2, pp. 699–706, 2007.

[37] **Michalowski, Marek P., H., Kozima and H., Sabanovic.** *A Dancing Robot for Rhythmic Social Interaction.* s.l. : 16th IEEE International Conference on Robot & Human Interactive Communication, Jeju, Korea, 2007.

[38] **Michalowski, Marek P. and Kozima, H.** *Methodological Issues in Facilitating Rhythmic Play with Robots.* s.l. : 16th IEEE International Conference on Robot & Human Interactive Communication, Jeju, Korea, 2007.

[39] **Michalowski, M., Sabanovic, S. and Michel, P.** *Roillo: Creating a Social Robot for Playrooms.* s.l. : In Proceedings of the 15th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN 2006), University of Hertfordshire, United Kingdom, September, 2006.

[40] **Weinberg, G., Driscoll, S. and Parry, M.** *Musical Interactions with a Perceptual Robotic Percussionist.* s.l. : Proceedings of IEEE International Workshop on Robot and Human Interactive Communication (RO-MAN) Nashville, TN , 2005.

[41] **Weinberg, G., Driscoll, S.** *The Perceptual Robotic Percussionist – New Developments in Form, Mechanics, Perception and Interaction Design.* s.l. : Proceeding of the ACM/IEEE International Conference on Human-Robot Interaction, 2007.

[42] **Weinberg, G., Aimi, R. and Jennings, K.** *The Beatbug Network: A Rhythmic System for Interdependent Group Collaboration.* s.l. : NIME 2002: 106-111, 2002.

[43] **Arsenio, A. and Fitzpatrick, P.** *Exploiting Cross-Modal Rhythm for Robot Perception of Objects.* s.l. : Proceedings of the 2nd International Conference on Computational Intelligence, Robotics, and Autonomous Systems, Singapore , 2003.

[44] **Arsenio, A. and Fitzpatrick, P..** *Exploiting Amodal Cues for Robot Perception.* s.l. : International Journal of Humanoid Robotics, 2:2, pp. 125-143 , 2005.

[45] **Lee, E. Enke, U., Borchers, J. and L. de Jong.** *Towards Rhythmic Analysis of Human Motion using Acceleration-Onset Times.* s.l. : In Proceedings of the 2007 Conference on New Interfaces for Musical expression (NIME07), New York, USA, 2007.