Robot Dance based on Online Automatic Rhythmic Perception

Authors Deleted for Blind Review
Institutions Deleted for Blind Review

Abstract. Rhythmic perception is intimately related to the production of rhythm in the form of music or dance movements. In this paper we present an architecture and the development of a robotic system using humanoid robots, based on the Lego Mindstorms NXT, which tries to simulate the human rhythmic perception from audio signals, and its reactive behavior in the form of dance. To do so we used a rhythmic perception model based on Marsyas, an open source software framework for audio processing, from which we construct a MarSystem with the needed blocks to perform onset feature detection. The model's output is sent to the robot control application in real-time, via sockets, shaped in three rhythmic events, representing soft, medium and strong onsets in the music. The dance movements can be dynamically defined trough an interactive interface and are performed by the robot in a reactive manner to these rhythmic events' occurrence. These movements also depend on two kinds of sensorial events, namely the color stepped on the floor or the proximity to some kind of obstacle. This interactive robot control keeps the dynamism manifested by the human behavior, granting spontaneous and chaotic dance movements in synchronism to music, without any previous knowledge of it.

1 Introduction

Music is basically formed by a succession of sounds and silence organized in time, being analytically a temporal phenomenon of organized sound, [1]. Whatever is the method and the esthetic objective, the sonorous material being used by music is traditionally divided by analysts in four essential organizational elements: Rhythm, Melody, Harmony, and Timbre.

Rhythm is the organizational element, frequently associated to the horizontal dimension, and more directly related to time (duration) and intensity (magnitude), as it is the basic outline of music in time. Rhythm is, in this sense, formed by sounds and silence temporally succeeding, each sound with its own duration and intensity, and each silence (with zero intensity) with its own duration.

Like music, also dance is a temporal art structured in this domain. Both are composed by progressive patterns along time, and the dance is normally composed by music, as it offers vigorous cues of temporal discrimination and segmentation that are imposed to dance.

Therefore, the body movement, in the dance form, emerges as a natural response to the periodicities of musical rhythm, being impossible to disassociate the role of the body

and movement in the perception and production of this element [2], since it assumes a mandatory role in the establishment of the symbiotic relation between music and dance.

In this work we used a rhythmic perception model which induces a robot to reactively execute proper dance movements in a time-synchronous way, but individually spontaneous, trying to simulate the dynamic movement behavior typical from human beings.

The analyzed music data is composed by digital polyphonic audio signals (continuous audio data with a small abstraction level, implicitly presenting the timing and the fundamental rhythmic events structure), which are processed and reproduced by the music analysis module. To obtain the intended rhythmic events we focused our analysis in the detection of the music onset times (starting time of each musical note) through an onset function (a function whose peaks are intended to coincide with the times of note onsets) that attends to the energy variation along each musical frame (*frame wise features*).

The robot's body movement reacts, therefore, to a conjunction of stimulus formed by three rhythmic events, namely: *Low, Medium* or *Strong Onsets*; and two sensorial event groups defined by the detected color: *Blue, Yellow, Green, Red*; and by the proximity to an obstacle: *Low, Medium, High*. Based on the interchange of these inputs a user can, through a proper interface, dynamically define every intended dance movements.

Contrasting to other approaches, every movement pattern, as the interchange among them during the dance, is this way produced by the robot in a seemingly autonomous way, without former knowledge of the music.

The paper structure is as follows. The next section presents some recent related work on dancing robots. Section 3 discusses the system architecture principles presenting an overview of the Lego Mindstorms NXT hardware and explaining the software basis on the music analysis implementation and in the application interface. Section 4 presents an overview of the given experiment and results. Finally section 5 concludes this paper presenting the main conclusions and future work.

2 Related work

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

From recent events on this subject we can refer the "Robot Conductor Leads Detroit Symphony", on May 13th, 2008, where ASIMO robot conducted the Detroit Symphony Orchestra in a performance of Mitch Leigh's "The Impossible Dream" from the Man from La Mancha. We should also refer the RoboDance contest that takes place in RoboCup international and national competitions where school teams, for children aged eight to nineteen, from all over the world, put their robots (mainly Lego NXTs) in

action, performing dance to music in a display that emphasized creativity of costumes and movement.

Nakazawa, Nakaoka *et al.* [3, 4] presented an approach that lets a biped robot, HRP-2 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.

Weinberg *et al.* [5, 6], developed a humanoid robot, Haile, which plays percussion instruments in synchrony with a musician (percussionist). 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, one to detect the music beat onsets and another to collect pitch and timbre information from it, granting synchronous and sequential rhythmic performance.

Tanaka *et al.* from Sony, built a dancing robot, QRIO, to interact with children, presenting a *posture mirroring* dance mode [7, 8]. The 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.

More recently, in 2007, Aucoutuier et al. [9] developed a robot designed by ZMP, called MIURO, in which they built basic dynamics through a special type of chaos (specifically, *chaotic itinerancy* (CI)) to let the behavior 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.

Michalowski *et al.* [10, 11] 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 behaviors in non-verbal interactions through dance between Keepon, a small yellow creature-like robot, and children.

Burger and Bresin [12] also used the Lego Mindstorms NXT to design a robot, named M[ε]X, that 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.

Yoshii *et al.* [13] used Honda's ASIMO 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 behavior.

3 System Architecture

3.1 Hardware - Lego Mindstorms NXT[1]

Lego Mindstorms NXT is a programmable robotic kit designed by Lego (see fig. 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 I provided my robot with a color sensor, to detect and distinguish visible colors, and an ultrasonic sensor, capable of obstacle detection, retrieving the robot's distance to it in inches or centimeters.

Based on this technology we built a humanoid-like robot (see fig.2) using two NXT bricks that controls six servo motors (one for each leg and two for each arm with two degrees of freedom (DOFs)) and two sensors, already referred. This robot design grants 18 distinct dance movements: *R-Arm Left, R-Arm Right, R-Arm Rotate, R-Arm Up, R-Arm Down, L-Arm Left, L-Arm Right, L-Arm Rotate, L-Arm Up, L-Arm Down, 2-Arms Left, 2-Arms Right, 2-Arms Rotate, 2-Arms Up, 2-Arms Down, Legs Forward, Legs Backward, Legs Rotate*.

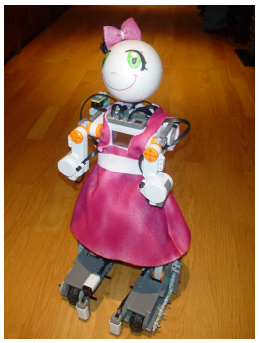

Fig.1. Lego NXT brick and some of its sensors and servo-motors. **Fig.2.** Our robot.

[1] For more information consult http://mindstorms.lego.com/eng/default.aspx

3.2 Software - Music analysis and Robot Control

The designed software application it's composed by two distinct modules, *Music Analysis* and *Robot Control*, that communicate with each other, via TCP sockets (see fig.3). The *Music Analysis* module uses a rhythm perception algorithm based on Marsyas to detect rhythmic events. These events are then sent in real-time to the Robot Control module which remotely controls the robot via Bluetooth.

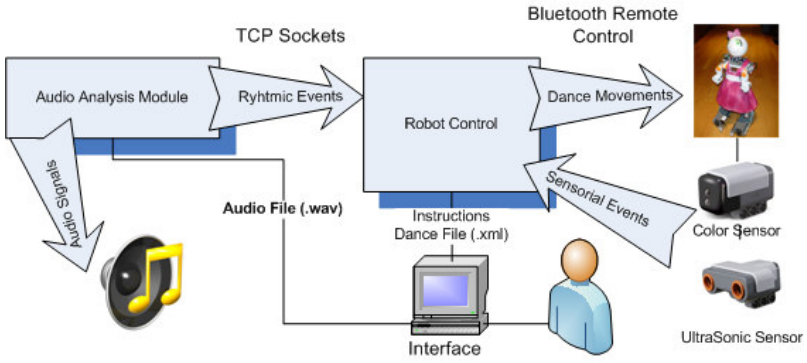

Fig.3. System architecture.

The application also presents an interface that grants user-robot interaction through a control panel, which achieves Bluetooth connection with both NXT bricks for any montage design, and a dance creation menu, which allows the user to dynamically define the robot choreography through dance movement in reaction to the cross-modulation of rhythmic and sensorial events, saving each dance in a proper .xml file (see fig.4 a) & b)).

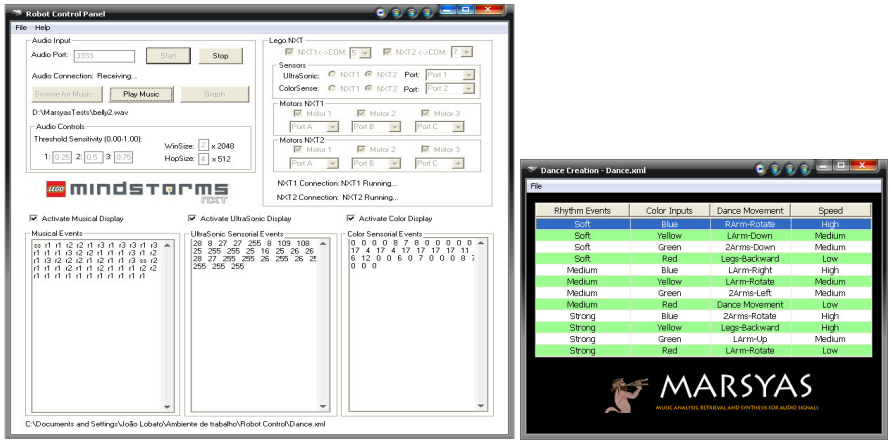

Fig. 4. *Application Interface.* ***a)*** *Robot Control Panel.* ***b)*** *Dance Creation.*

3.2.1 Music Analysis Module

Marsyas (Music Analysis, Retrieval and Synthesis for Audio Signals)[2]

Our rhythmic perception algorithm is designed under Marsyas. Marsyas 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.

Rhythmic Perception Considerations and Architecture

Under Marsyas we built a MarSystem (an aggregation of functional blocks) (see fig. 5) that performs onset feature detection from polyphonic audio signals, in real-time, based on frame energy variations along the music.

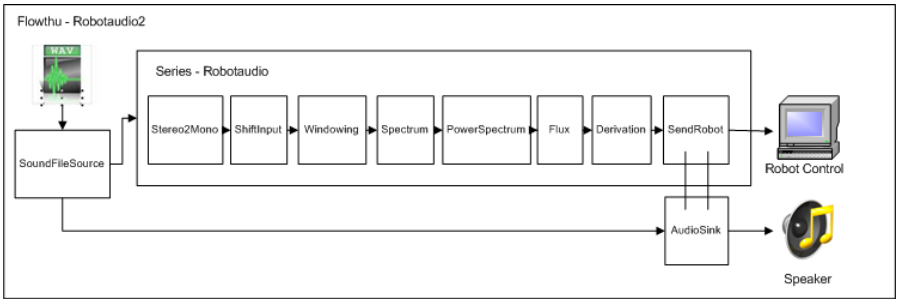

Fig. 5. MarSystem constitution with onset detection function blocks.

First the stereo input audio signal is converted to mono (with *Stereo2Mono*), and then consecutive frames are overlapped (with *ShiftInput*) to grant a more stable analysis. The analysis step is called hop size and equals the frame size minus the overlap (typically 10 ms).

To the Shifted signal is applied the FFT (Fast Fourier Transform) algorithm (with *Spectrum*) using a Hamming window (in *Windowing*) to obtain the music spectrum. To the *Spectrum* output is applied a *PowerSpectrum* function that retrieves the energy variation (magnitude – in dBs) along the music.

Then to this signal is added a Spectral *Flux* function that represents the actual onset detection method. This onset detection model is based on Dixon's (2006), [14], results

[2] For more information consult http://marsyas.sness.net/

which evince the Spectral Flux (SF) function as the one achieving the best results in the simplest and fastest way. SF measures the change in magnitude in each frequency bin (*k*) of each frame (*n*), restricted to the positive changes and summed across all *k*, with the given Onset Function (OF):

$$OF = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} H\left(|X\left(n,k\right)| - |X\left(n-1,k\right)|\right) . \quad \textbf{(1)}$$

where $H(x) = \frac{x + |x|}{2}$ **(2)**, is the half-wave rectifier function and X(x) the FFT.

The *Derivation* block retrieves only the crescent *Flux* output, by subtracting the *n* frame to the *n-1* one.

Finally the *SendRobot* block acts as our Peak Picking function and TCP client. It applies a peak adaptive thresholding algorithm to distinguish three rhythm events: *Strong, Medium* and *Soft* onsets, which then sends to the *Robot Control* module via TCP sockets.

3.2.2 Robot Control Module

The Robot Control Module represents the application GUI and uses a C++ NXT Remote API, designed by Anders Søborg[3], to remotely control the robot.

Each dance movement it's defined by six speed vectors and one time vector, representing each motor's speed and the time needed to fulfill the movement.

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. 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. This however incited a group delay that increased with the decrease of the normalized cutoff frequency (*Wn*), promulgating a minimum delay of 10 frames which is, in addition to the whole process natural delay, considerably high facing the requirements. In a way to bypass this issue we decided to slightly increase the window and hop size, which granted a lower sensitivity in onset detection focusing on the more relevant ones, with no delay imposed in the process.

In order to restrict and distinguish our three rhythmic events we designed a Peak Picking (PP) function with peak adaptive thresholding (always related to the highest onset detected so far) as follows:

[3] For more information consult http://www.norgesgade14.dk/index.php

$$PP(x) = \begin{cases} Strong, & if\ x > \delta_3 \\ Medium, & if\ \delta_2 < x < \delta_3 \\ Soft & if\ \delta_1 < x < \delta_2 \end{cases}. \quad \textbf{(3)} \qquad \text{where,} \begin{cases} \delta_1 = thres_1 \times peak \\ \delta_2 = thres_2 \times peak \\ \delta_3 = thres_3 \times peak \end{cases}, \quad \textbf{(4)}$$

$$0 < thres_i > 1$$

The values of $thres_1$, $thres_2$, $thres_3$, as the values of window size and hop size can be dynamically assigned in the application's interface. The function waits 35 frames to initialize the onset detection, starting with peak = (1/2) * - highest onset detected until then. This acts as the function normalization due to inconsistency in the beginning of any music data.

To check the adequate rhythm perception parameters to a large set of music data, we outfitted our application interface with a graph mode that uses the parameters inserted by the user to plot the respective output showing the three kinds of rhythm events detected along the music. This representative graph is plotted in MatLab due to the Marsyas' MatLab engine capabilities.

The set of tests were performed on the data used by Bello et al. [15], consisting of 4 sets of short excerpts (each with around 20s) from a range of instruments, classed into the following groups: NP — non-pitched percussion, such as drums; PP—pitched percussion, such as guitar; PN — pitched non-percussion, in this case some Tango violin; and CM — complex mixtures from popular and jazz music. Below we show some screenshots (fig. 6) and a table (table 1) with the tests results.

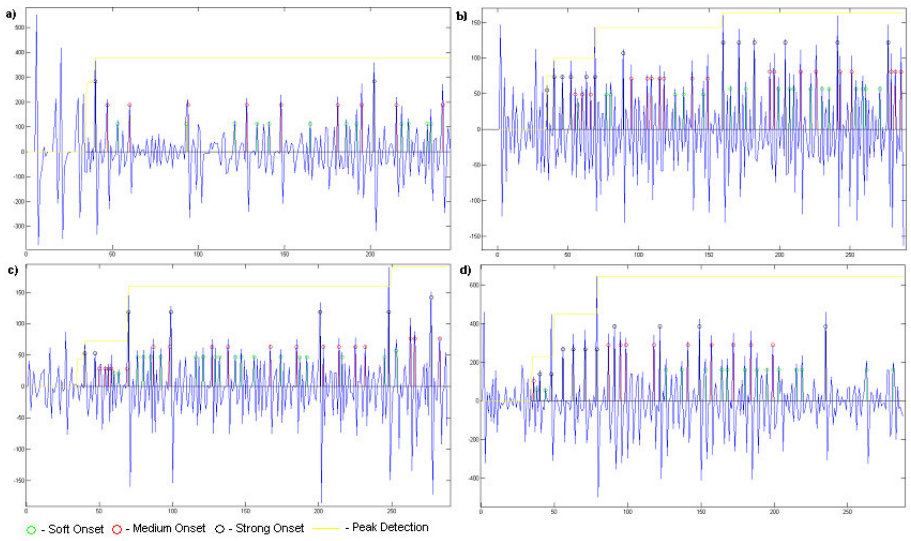

Fig. 6. Peak Picking and Onset Detection output. **a)** PN excerpt using $thres_1 = 0.30$; $thres_2 = 0.50$; $thres_3 = 0.75$. **b)** PP excerpt using $thres_1 = 0.35$; $thres_2 = 0.50$; $thres_3 = 0.75$. **c)** NP excerpt

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 1. Resultant onset counting for the performed tests (above).

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 inconsistency among the different music styles, as shown, we were compelled to define different parameters for each music data. To go around this issue we created a text file to each music file containing the respective parameters, from where the application imports them.

5 Conclusions and Future Work

We developed a biped humanoid robot that reacts to music in real-time, performing dance movements in synchronism to rhythm in a chaotic and seemingly autonomous way. This was achieved with a proper system architecture constituted by two modules (*Music Analysis* and *Robot Control*) that communicate via TCP sockets. The *Music Analysis* module is based on the Marsyas rhythm perception model based on onset feature detection, with peak picking and adaptive thresholding. The *Robot Control* reacts to the rhythm events sent by the former module, in real-time, and to the received sensorial events, promoting robotic dance movements, as defined in the *Dance Creation* interface.

This way our robot represents the significant first 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.

In future work, we will address the issue of automatic parameter estimation, with the aim of producing a fully automatic onset detection algorithm. We will also add some beat prediction capability applying a beat tracking algorithm to complement the onset detection, and this way design a more efficient and realistic rhythm perception module.

In our robotic system we will also address the issue of multi-robot dance, implementing a swarming system that allows robot-robot interaction while dancing, allowing the creation of synchronous and dynamic choreographies. We will also improve the robots sensitivity by adding other sensorial events, such as acceleration and orientation.

Finally we want to improve our application to be used as didactic software by children (and other people) to create their own robotic dances, and even to be used as a framework for creating fully functional systems for RoboDance competitions.

References

1. Gouyon, F., Dixon, S.: A Review of Automatic Rhythmic Description Systems. Computer Music Journal 29:1 (2005).
2. Guedes, C.: Mapping Movement to Musical Rhythm – A Study in Interactive Dance. PhD thesis, New York University, New York, USA (2005).
3. Nakazawa, A., Nakaoka, S., Ikeuchi, K., Yokoi, K.: Imitating Human Dance Motions through Motion Structure Analysis. IROS, pp. 2539–2544 (2002).
4. Nakaoka, S. et al.: Learning from Observation Paradigm: Leg Task Models for Enabling a Biped Humanoid Robot to Imitate Human Dance. Int'l J. Robotics Research, vol. 26, no. 8, pp. 829–844 (2007).
5. Weinberg, G., Driscoll, S., Parry, M.: Musical Interactions with a Perceptual Robotic Percussionist. Proceedings of IEEE International Workshop on Robot and Human Interactive Communication (RO-MAN) Nashville, TN (2005).
6. Weinberg, G., Driscoll, S.: The Perceptual Robotic Percussionist – New Developments in Form, Mechanics, Perception and Interaction Design. Proceeding of the ACM/IEEE International Conference on Human-Robot Interaction (2007).
7. Tanaka, F., Suzuki, H.: Dance Interaction with QRIO: A Case Study for Non-boring Interaction by using an Entertainment Ensemble Model. Proceedings of the 2004 IEEE International Workshop on Robot and Human Interactive Communication (RO-MAN), pp. 419-424, Kurashiki, Japan (2004).
8. Tanaka, F., Fortenberry, B., Aisaka, K., Movellan, J.: Plans for Developing Real-time Dance Interaction between QRIO and Toddlers in a Classroom Environment. Proceedings of 2005 4th IEEE International Conference on Development and Learning (ICDL), pp. 142-147, Osaka, Japan (2005).
9. Aucouturier, J.-J., Ogai, Y.: Making a Robot Dance to Music Using Chaotic Itinerancy in a Network of FitzHugh-Nagumo Neurons. Proceedings of the 14th International Conference on Neural Information Processing (ICONIP), Kitakyushu, Japan (2007).
10. Marek, P., Michalowski, Sabanovic, S., Kozima, H.: A Dancing Robot for Rhythmic Social Interaction. 16th IEEE International Conference on Robot & Human Interactive Communication, Jeju, Korea (2007 a).
11. Marek, P., Michalowski, Kozima, H.: Methodological Issues in Facilitating Rhythmic Play with Robots. 16th IEEE International Conference on Robot & Human Interactive Communication, Jeju, Korea (2007 b).
12. Burger, B., Bresin, R.: Displaying Expression in Musical Performance by Means of a Mobile Robot. In Paiva, A., Prada, R., & Picard, R. W. (Eds.), Affective Computing and Intelligent Interaction (pp. 753-754). Berlin / Heidelberg: Springer (2007).
13. Yoshii, K., Nakadai, K., Torii, T., Hasegawa, Y., Tsujino, H., Komatani, K., Ogata, T., Okuno, H.: A Biped Robot that Keeps Steps in Time with Musical Beats while Listening to Music with Its Own Ears. Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-2007), 1743-1750, IEEE, RSJ, San Diego (2007).
14. Dixon, S.: Onset detection revisited. In Proc. of the Int. Conf. on Digital Audio Effects (DAFx-06), pages 133–137, Montreal, Quebec, Canada, Sept. 18–20 (2006).
15. Bello, J.P., Daudet, L., Abdallah, S., Duxbury, C., Davies, M., Sandler, M.: A tutorial on onset detection in musical signals. IEEE Transactions on Speech and Audio Processing, vol. 13, no. 5, pp. 1035–1047 (2005).