Towards an Interactive Framework for Robot Dancing Applications

João Oliveira, Fabien Gouyon, and Luis Paulo Reis

FEUP – Faculty of Engineering of the University of Porto, Portugal, ee03123@fe.up.pt

INESC Porto– Systems and Computers Engineering National Institute, Porto, Portugal, fgouyon@inescporto.pt

LIACC – Artificial Intelligence and Computer Science Lab., University of Porto, Portugal, lpreis@fe.up.pt

Abstract — **In this paper we present the architecture for a robotic system using a humanoid robot, 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 implemented low-level aspects of rhythm perception based on Marsyas, an open source software framework for audio processing. 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 dynamic dance movements in synchronism to music, without any previous knowledge of it.**

Index Terms — **Acoustic applications, Acoustic signal analysis, Interactive systems, Multimedia systems, Music, Robot dynamics, Robots**

I. 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). 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 behaviors. This is where we focused our efforts by designing an interactive framework for robot dancing applications based on automatic music signal analysis.

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 [13]. 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.

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 behavior typical from human beings.

The robot's body movement reacts 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 body movement, as their sequence 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 musical robots. Section III 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 IV presents an overview of the given experiment and results. Finally section V concludes this paper presenting the main conclusions and future work.

II. RELATED WORK

Academically, "dancing robots" and "human-robot musical interaction" are 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 dancing robot that could interact with humans.

Nakazawa, Nakaoka et al. [1]-[2] 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. [3]-[4], 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 make this performance they used two Max/MSP objects, one to detect the music beats 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 [5]-[6]. This interaction 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. [7] 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. [8]-[9] 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 [10] 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. [11] 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 a significant 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.

In contrast to previous approaches, in this paper we propose a framework in which users have a deterministic role, by dynamically defining the robot choreography through selected individual dance movements.

III. SYSTEM ARCHITECTURE

A. Hardware - Lego Mindstorms NXT

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 we provided the robot with a color 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 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 each arm, one for a rotating hip and one for the head) and two sensors, already referred. 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*.

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

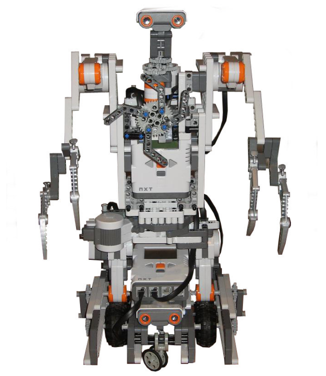


Fig. 2. Our robot.

B. Software - Music analysis and Robot Control

The designed software application is 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.

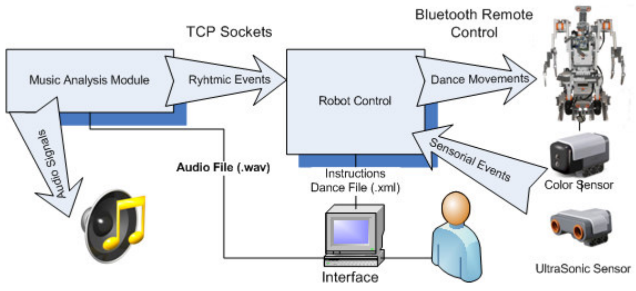


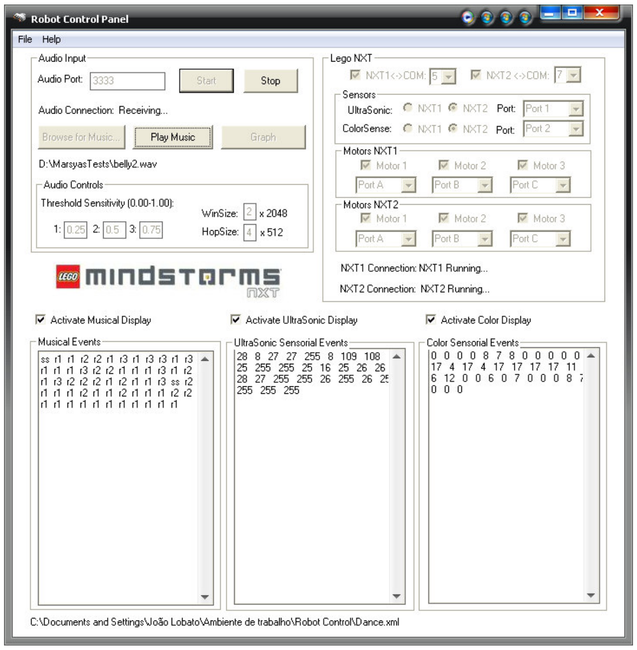
Fig. 3. System architecture.

The application also presents an interface that grants user-robot interaction through a control panel, which achieves Bluetooth connection with one or two NXT bricks, depending on the 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)).

B.1. Music Analysis Module

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

Our *Music Analysis* module is designed in 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 the help of students and researchers from all around the world. Marsyas has been used for a variety of projects in both universities and industry.



Robot Control Panel

Audio Input
Audio Port: 3333
Audio Connection: Receiving...
D:\MarsyasTests\belly2.wav
Audio Controls
Threshold Sensitivity (0.00-1.00):
1: 0.25 2: 0.5 3: 0.75
WinSize: 2 x 2048
HopSize: 4 x 512

Lego NXT
[✓] NXT1<->COM: 5 [✓] NXT2 <->COM: 7
Sensors
UltraSonic: NXT1 [✓ NXT2] Port: Port 1
ColorSense: NXT1 [✓ NXT2] Port: Port 2
Motors NXT1
[✓] Motor 1 Port A [✓] Motor 2 Port B [✓] Motor 3 Port C
Motors NXT2
[✓] Motor 1 Port A [✓] Motor 2 Port B [✓] Motor 3 Port C
NXT1 Connection: NXT1 Running...
NXT2 Connection: NXT2 Running...

[✓] Activate Musical Display
[✓] Activate UltraSonic Display
[✓] Activate Color Display

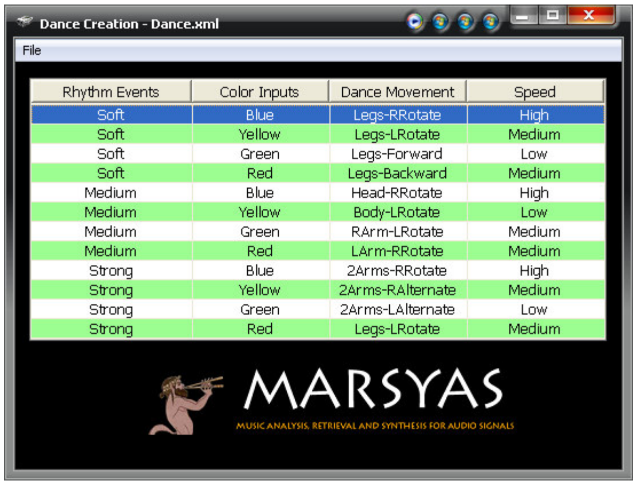
Musical Events
```
ss r1 r1 r2 r2 r1 r3 r1 r3 r3 r1 r3
r1 r1 r1 r3 r2 r2 r1 r1 r1 r3 r1 r2
r1 r3 r2 r2 r2 r1 r2 r1 r1 r3 ss r2
r1 r1 r1 r2 r1 r1 r2 r1 r1 r1 r2 r2
r1 r1 r1 r1 r1 r1 r1 r1 r1 r1 r1
```

UltraSonic Sensorial Events
```
28  8  27  27  255  8  109  108
25  255  255  25  16  25  26  26
28  27  255  255  26  255  26  25
255  255  255
```

Color Sensorial Events
```
0 0 0 0 8 7 8 0 0 0 0 0
17 4 17 4 17 17 17 17 11
6 12 0 0 6 0 7 0 0 0 8 7
0 0 0
```

C:\Documents and Settings\João Lobato\Ambiente de trabalho\Robot Control\Dance.xml

Fig. 4. Application Interface. a) Robot Control Panel.



Dance Creation - Dance.xml

Rhythm Events	Color Inputs	Dance Movement	Speed
Soft	Blue	Legs-RRotate	High
Soft	Yellow	Legs-LRotate	Medium
Soft	Green	Legs-Forward	Low
Soft	Red	Legs-Backward	Medium
Medium	Blue	Head-RRotate	High
Medium	Yellow	Body-LRotate	Low
Medium	Green	RArm-LRotate	Medium
Medium	Red	LArm-RRotate	Medium
Strong	Blue	2Arms-RRotate	High
Strong	Yellow	2Arms-RAlternate	Medium
Strong	Green	2Arms-LAlternate	Low
Strong	Red	Legs-LRotate	Medium

b) Dance Creation Interface

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.

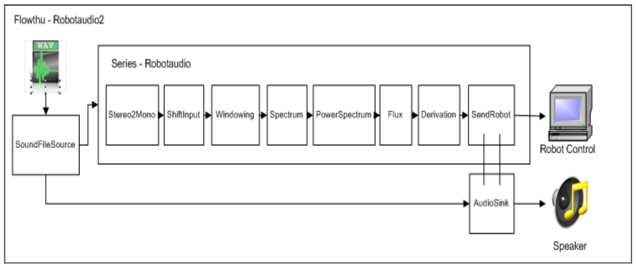


Fig. 5. MarSystem constitution with onset detection function blocks.

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 submitted to a Spectral *Flux* function that represents the actual onset detection method. This onset detection model is based on [12] results, 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 Detection Function (ODF):

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

where $H(x) = \dfrac{x + |x|}{2}$ is the half-wave rectifier function and $X(n,k)$ 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 are then sent to the *Robot Control* module via TCP sockets.

B.2. Robot Control Module

The Robot Control Module represents the application GUI and uses a C++ NXT Remote API, designed by Anders Søborg, to remotely control the robot.
Each dance movement is defined by six speed vectors and one time vector, representing each motor's speed and the time needed to fulfill the movement.

IV. 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, at first we applied 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 (aprox. 0.7s) 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 (to *WinSize* = 4096 and *HopSize* = 3072) 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:

$$PP(x) = \begin{cases} Strong, & if\ x > \delta_3 \\ Medium, & if\ \delta_2 < x < \delta_3 \\ Soft & if\ \delta_1 < x < \delta_2 \end{cases} \tag{2}$$

$$\text{where, } \begin{cases} \delta_1 = thres_1 \times peak \\ \delta_2 = thres_2 \times peak \\ \delta_3 = thres_3 \times peak \end{cases},\ 0 < thres_i > 1\ . \tag{4}$$

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 (equal to 2.43s due to fs_{Flux} = 14.36Hz) to initialize the onset detection, starting with peak = (1/2) * - highest onset detected until then. This acts as the function normalization due to potential inconsistency in the beginning of some music data.

To check the adequate rhythm perception parameters to a large set of music data, we embraced 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 diverse music styles, consisting of 4 sets of short excerpts (each with around 20s) from a range of instruments, classed into the following groups [13]: 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 (fig. 6) and a table (table 1) with the tests results.

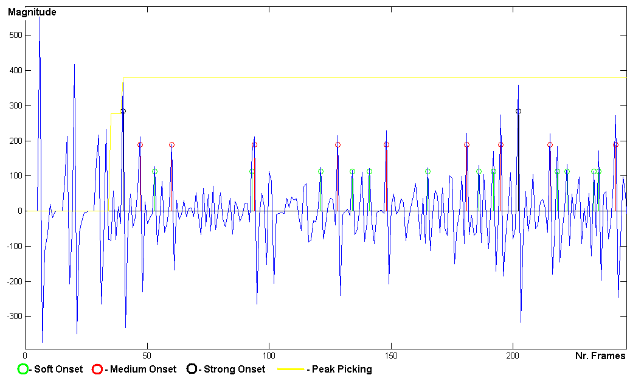
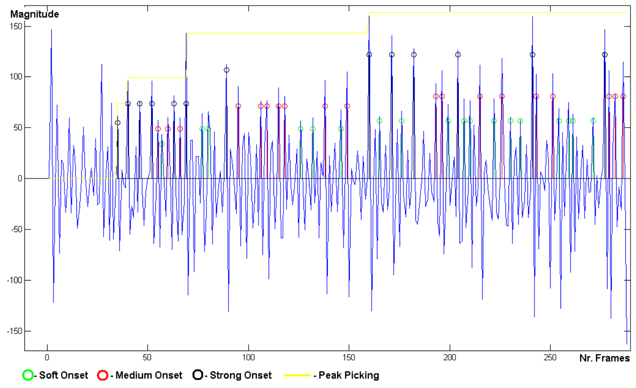
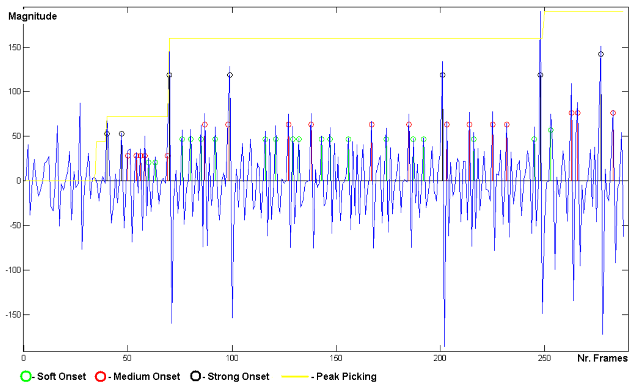


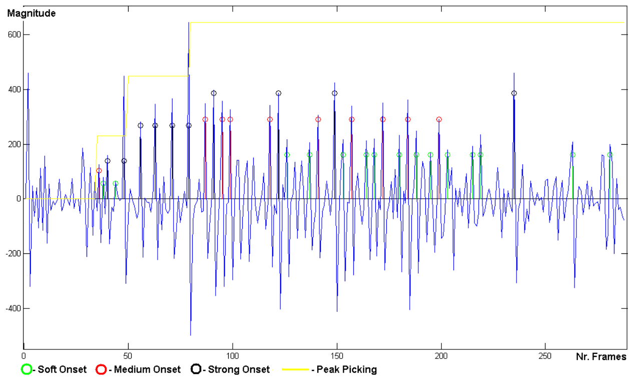
Fig. 6 Peak Picking and Onset Detection output. a) PN excerpt using $thres_1 = 0.30$; $thres_2 = 0.50$; $thres_3 = 0.75$.



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 I. 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.

V. 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 dynamic 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 an onset detection function, 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.

By doing this, our robot enforces the significant first step to create 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 apply an automatic music style definition that also addresses 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 a didactic software to children (and people in general) to create their own robotic dances, and even to be used as a framework for creating fully functional systems for RoboDance competitions.

References

[1] A. Nakazawa, S. Nakaoka, K. Ikeuchi, K. Yokoi, "Imitating Human Dance Motions through Motion Structure Analysis," IROS, pp. 2539–2544 (2002).

[2] S. Nakaoka 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).

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

[4] G. Weinberg, S. Driscoll, "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).

[5] F. Tanaka, H. Suzuki, "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).

[6] F. Tanaka, B. Fortenberry, K. Aisaka, J. Movellan, "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).

[7] J.-J. Aucouturier, Y. Ogai, "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).

[8] P. Marek, Michalowski, S. Sabanovic, H. Kozima, "A Dancing Robot for Rhythmic Social Interaction," 16th IEEE International Conference on Robot & Human Interactive Communication, Jeju, Korea (2007 a).

[9] P. Marek, Michalowski, H. Kozima, "Methodological Issues in Facilitating Rhythmic Play with Robots," 16th IEEE International Conference on Robot & Human Interactive Communication, Jeju, Korea (2007 b).

[10] B. Burger, R. Bresin, "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).

[11] K. Yoshii, K. Nakadai, T. Torii, Y. Hasegawa, H. Tsujino, K. Komatani, T. Ogata, H. Okuno, "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).

[12] S. Dixon, "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).

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