UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

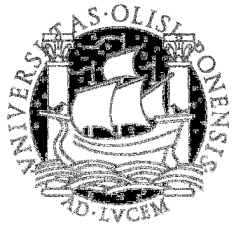
CAN CAN ROBÓTICO

Marco António Morais Lourenço

MESTRADO EM ENGENHARIA INFORMÁTICA
Interacção e Conhecimento

2011

UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática



CAN CAN ROBÓTICO

Marco António Morais Lourenço

DISSERTAÇÃO

Trabalho orientado pelo Prof. Doutor Paulo Jorge Cunha Vaz Dias Urbano
e co-orientado por Prof. Doutor Carlos Jorge da Conceição Teixeira

MESTRADO EM ENGENHARIA INFORMÁTICA
Interacção e Conhecimento

2011

Agradecimentos

Se há coisa que não posso fazer é agradecer apenas aqueles que me acompanharam ao longo do ano que decorreu este projecto, pois se aqui cheguei devo-o em particular a cada uma das pessoas que me apoiou e me auxiliou ao longo dos meus anos académicos. Corro o risco de me alongar em agradecimentos ou mesmo de me esquecer de alguém, mas aqui deixo os meus agradecimentos do fundo do coração a todos aqueles que me ajudaram e que me ajudam ainda hoje.

Começo então, como não podia deixar de ser por agradecer aos meus pais e ao meu irmão por terem tido paciência comigo todos estes anos. Só eles sabem o quanto os aborreci com os problemas que me foram aparecendo.

Quero também agradecer ao resto da minha família, deixando um especial obrigado à minha avó Joana que me criou com tanto carinho e dedicação. Sei que estás algures a olhar por mim e que hoje ficas um pouco mais orgulhosa. Obrigado avó.

À minha namorada Rita, o grande amor da minha vida que me foi segurando nos momentos difíceis e que esperava por mim de braços abertos depois das longas horas que eu gastava a trabalhar nos projectos que me guiaram ao que aqui apresento hoje.

Como é óbvio, não me esqueço também dos meus amigos que estiveram ao meu lado durante este percurso. Segue então um especial obrigado para o Daniel Policarpo e o Ricardo Mascarenhas, que foram os meus grandes companheiros. Não querendo esquecer ninguém e apesar de saber que não os posso apontar a todos, atrevo-me a apontar alguns dos nomes que me acompanharam desde o inicio. Assim, um obrigado também para “Chimy”, “Copinho”, Craveiro, Dias, “Fufinha”, “Insta”, Lopes e “Mano”.

Tenho de agradecer ainda a todos os meus colegas de LabMAg, em especial ao André, Christian, Davide, Fernando, Gil, João, “Jomi”, Jorge, Miguel e Phil que foram vítimas do meu mau humor mas também companheiros de brincadeiras.

Falta-me ainda agradecer aos meus orientadores, Paulo Urbano e Carlos Teixeira pelo apoio dado durante o projecto e na escrita deste documento.

À vida académica, e aos momentos bons que me fez viver

Resumo

Cada vez mais a área de robótica tem vindo a despertar maior interesse devido à sua aplicação em diversas áreas que vão desde aplicações militares até aplicações domésticas. Uma das áreas que tem sido bastante desenvolvida é a área de entretenimento. Nesta área existem robôs que fazem o papel de animais de estimação e robôs que funcionam como parceiros de dança ou de conversa.

O objectivo deste projecto consiste em explorar a área de entretenimento através da intersecção do mundo da robótica e da dança, criando um robô dançarino que seja capaz de reagir ao ritmo musical em tempo real e de dançar de acordo com a música que está a tocar.

A maioria dos robôs bailarinos dança utilizando a imitação. Com algumas excepções como seja o caso do robô da Lego de João Manuel Oliveira que reage à música que está a ouvir em tempo real. No entanto este robô não tem como objectivo principal a dança mas sim o sistema de análise de som.

Neste documento serão apresentados os passos dados com vista a construir um robô capaz de dançar cancan em tempo real.

Palavras-chave: robôs; robôs dançarinos; análise de som; cancan

Abstract

More and more robotics is an area that has come to awake a greater interest due to its application in diverse areas that go from military applications to domestic ones. One of the areas that has been greatly developed is entertainment. In this area there are robots that do the part of pet animals and those who work as dance or talk partners.

The objective of this project consists in exploring the area of entertainment through the intersection of robotic and dance world, creating a dancing robot that is capable of reacting to musical rhythm in real time and dance according to a song that is playing.

Most dancing robots use imitation. With a few exception like the case of the Lego robot of João Manuel Oliveira that reacts to the music that it's listening in real time. However this robot prime goal isn't dance but a system of sound analysis.

In this document will be presented the steps towards building a robot capable of dancing "cancan" in real time.

Keywords: robots; dancing robot; sound analysis; cancan

Conteúdo

Capítulo 1 Introdução 1

1.1 Motivação 1

1.2 Objectivos 2

1.3 A dança cancan 2

1.4 Trabalho realizado 3

1.5 Material utilizado 4

1.5.1 Marsyas 5

1.5.2 Kit Lego Mindstorms NXT 5

1.5.3 iCommand 6

1.6 Contribuições científicas 7

1.7 Organização do documento 7

Capítulo 2 Estado da Arte 9

2.1 Dança Robótica 9

2.2 Análise de som 14

2.2.1 Detecção de *onsets* 15

2.2.2 Detecção de *beats* 17

Capítulo 3 Modelo Inicial (Catatua Dançarina) 19

3.1 Reagindo a *onsets* 20

3.1.1 Arquitectura do sistema 20

3.1.2 Módulo de análise de som 20

3.1.3 Construção do robô 21

3.1.4 Módulo de controlo do robô 22

3.2 Reagindo a *onsets* com conhecimento prévio 22

3.2.1 Arquitectura do sistema 23

3.2.2 Módulo de análise de som 23

3.2.3 Módulo de controlo do robô 24

3.3 Discussão de resultados 25

Capítulo 4 Modelo Final (Pernas Cancan) 27
4.1 Arquitectura do sistema 27
4.2 Módulo de análise de som com Beatroot 28
4.3 Construção do robô 31
4.4 Coreografia 32
4.5 Módulo de controlo do robô 35
4.5.1 Controlador geral 36
4.5.2 Controlador da perna esquerda e cintura 39
4.5.3 Controlador da perna direita 40
4.6 Discussão de resultados 41
Capítulo 5 Conclusões e trabalho futuro 43
Referências 45

Lista de Figuras

1.1 *Brick* ligado a três servo-motores e quatro sensores 6

2.1 Robô humanóide utilizado por Riley et al 9

2.2 Robô HRP-2 10

2.3 Sony Dream Robot 'QRIO' 11

2.4 Robô Honda's ASIMO a bater os pés ao ritmo da música 12

2.5 Robô Keepon 13

2.6 Robô Lego NXT de Oliveira et al 14

2.7 Humanoide RoboNova 14

2.8 *Onset*, *attack*, *transient* e *decay* de uma nota 15

2.9 Gráfico que demonstra o algoritmo de detecção de onsets 16

3.1 Arquitectura do sistema da versão inicial da Catatua Dançarina 20

3.2 Catatua Dançarina.....21

3.3 Arquitectura do sistema da segunda versão da Catatua Dançarina 23

4.1 Arquitectura do sistema do modelo final.....28

4.2 Diagrama com a representação da arquitectura do módulo de análise de som 29

4.3 Interface gráfica do Beatroot 30

4.4 Exemplo de um ficheiro com os primeiros *beats* da música de Offenbach 30

4.5 Pernas Cancan 31

4.6 Diagrama com a representação da construção do robô 32

4.7 Localização dos motores 32

4.8 Exemplo de um pedaço do ficheiro de coreografia 35

4.9 Arquitectura do sistema com foco para o módulo de controlo do robô 36

4.10 Ilustração dos valores de *rotation count* para os joelhos e coxas.....38

4.11 Funcionamento do sistema usando dois computadores.....41

4.3 Pernas Cancan vestido (de frente) 42

4.4 Pernas Cancan vestido (de lado).....42

Lista de Tabelas

4.1 Movimentos reconhecidos pelo módulo de controlo do robô e sua composição a partir de movimentos simples.....34

4.2 Conversão dos movimentos lidos do ficheiro de coreografia em *rotation count*.....37

Capítulo 1

Introdução

Este capítulo está organizado da seguinte forma: Na secção 1.1 é descrita a motivação para a realização deste trabalho; seguidamente na secção 1.2 são descritos os objectivos deste trabalho; na secção 1.3 é descrito o software e hardware utilizado neste trabalho; na secção 1.4 são apresentadas as contribuições científicas deste trabalho; finalmente, na secção 1.5 é descrita a organização deste documento.

1.1 Motivação

A área de robótica é a cada dia que passa uma área de maior interesse. Desde que foram criados os primeiros robôs que os investigadores têm tentado variar as áreas de aplicação da robótica. Hoje em dia, temos robôs capazes de conduzir veículos, realizar missões de exploração, realizar tarefas domésticas, dar auxilio a operações cirúrgicas… No entanto, uma das áreas com grande interesse é uma área bem simples e com bastante visibilidade, o entretenimento.

Num mundo em que a diversão é cada vez mais descurada devido ao stress causado pelo dia a dia, o entretenimento é a salvação para os nossos problemas. Uma salvação que também os investigadores da robótica vêm como área de acção.

Então, com vista a trazer mais e melhor entretenimento à população do mundo, os investigadores da robótica têm vindo a desenvolver robôs capazes de interagir com humanos. São diversas as formas de interacção, desde robôs que conseguem conversar com humanos até robôs capazes de dançar para os entreter.

Foi esta última forma de interacção que despertou interesse. Robôs capazes de dançar para entreter os humanos.

Assim, este documento descreve os procedimentos utilizados para criar um robô capaz de realizar uma dança específica chamada cancan.

1.2 Objectivos

Apesar de já existir muita investigação na área da dança robótica, nem todos os sistemas permitem aos robôs combinar capacidades de análise de som com capacidades de realizar movimentos ao ritmo do som analisado, pois a maioria dos trabalhos nesta área não têm um funcionamento em tempo real. O que é proposto nesta abordagem, é um sistema que permita realizar movimentos que se assemelhem aos de um humano ao ritmo da música com uma análise de som em tempo real. As abordagens anteriores a esta pecavam por não terem movimentos semelhantes aos de um humano, ou quando os tinham, dançavam coreografias baseadas em aprendizagem por observação e não davam atenção ao ritmo da música enquanto esta era reproduzida.

Então, o objectivo principal deste trabalho é criar um sistema para um robô modelado com um kit educacional Lego NXT que seja capaz de analisar os eventos sonoros de uma música em tempo real e com base nesses eventos interpretar uma coreografia que possa ser adaptável às instruções dos utilizadores.

Para que este objectivo seja cumprido, foram criadas algumas metas no percurso, sendo elas:

- Escolha do sistema de análise de som
- Construção do robô
- Definição dos movimentos
- Elaboração da coreografia.

1.3 A dança cancan

Cancan é uma dança em ritmo acelerado que nasce em Paris por volta de 1830. Apesar de ter nascido em Paris, o nome de cancan foi pela primeira vez usado em Londres, pois antes, a dança era conhecida pelo nome de quadrilha. A quadrilha é uma dança de oito minutos de cortar a respiração que era coreografada ao ritmo da musica de Offenbach.

Quando o cancan surgiu, foi considerado uma provocação à sociedade, visto que era dançada por bailarinas que usavam roupas coloridas e esvoaçantes, que utilizavam movimentos extremos de sensualidade misturados com ousadas acrobacias.

Com o passar dos tempos, o cancan dividiu-se em duas existências paralelas, uma delas em França onde continuou a ser uma dança individual e a outra em Inglaterra onde passou a ser uma dança de grupo, na qual as bailarinas se colocavam em linha para desempenhar uma coreografia elaborada.

Mais tarde, o estilo de cancan dançado em Inglaterra é adoptado também pelos franceses, originando o famoso cancan francês.

O cancan francês, já durava mais tempo do que a quadrilha, era uma dança que durava pelo menos 10 minutos na qual as bailarinas tinham um tempo para mostrar as suas especialidades.

Alguns dos movimentos mais famosos do cancan, são o "*high kick*", que é o nome dado ao movimento no qual as bailarinas elevam a perna até à cabeça, o "*rond de jambe*", que é um movimento de rotação da parte inferior da perna enquanto o joelho está levantado e o "*port d'armes*", que é o movimento no qual as bailarinas fazem parte de uma rotação enquanto seguram o tornozelo no ar.

1.4 Trabalho realizado

Na sua génese, esta dissertação teve como inspiração principal o trabalho de João Oliveira [1]. Nesse trabalho, João Oliveira desenvolveu um robô com um Kit Lego Mindstorms NXT, que dançava adaptando-se ao ritmo da música e ao ambiente físico. Para fazer a adaptação ao ritmo da música, o robô detecta os tempos de *onset,* que correspondem tempo de inicio de cada nota musical. Esta detecção é feita recorrendo a uma função da Framework *open source* Marsyas [2][3], dedicada ao processamento áudio. O tipo de movimentos do robô apresentado por João Oliveira, consistia apenas em movimentos de rotação horária e anti-horária dos braços, cintura e cabeça, sendo o movimento da parte inferior do robô realizado por rodas.

Inicialmente adoptei como sistema de análise de som, o Marsyas utilizando a detecção de *onsets*, mas ao contrário do trabalho de João Oliveira tentei dar ao robô movimentos mais precisos e mais semelhantes aos de um humano.

Comecei por desenvolver um modelo simples de robô com um único movimento, baseado na dança de uma catatua [4], que tem por objectivo integrar os diversos componentes necessários para o desenvolvimento do robô. Este modelo simples tem o formato físico da cabeça de uma catatua que reage aos eventos musicais (*onsets*), realizando movimentos com a cabeça para cima e para baixo. Não havendo conhecimento de quando iria ocorrer o *onset* seguinte, o robô realizava os movimentos a uma velocidade constante. Isto levava a que não houvesse uma adaptação ao ritmo da música. Assim, por vezes o movimento termina antes de ser identificado um *onset* e o robô fica em espera, e outras vezes o movimento não chega a terminar e acaba por ser interrompido.

Para tornar o movimento mais adaptado à música seria necessário utilizar um sistema que permitisse prever a ocorrência de eventos musicais. No entanto, não tendo

sido possível utilizar um sistema capaz de fazer essa previsão foi necessário tentar manter o conceito de tempo real adaptando o sistema para ter uma capacidade simular a previsão. Para isso realizei um pré-processamento ao áudio, que proporciona o conhecimento relativo a quanto tempo irá existir entre dois *onsets* consecutivos. Assim, foi possível desenvolver uma segunda versão da catatua, com um comportamento mais adaptável ao ritmo a música. Esta nova versão modifica a velocidade do motor, adaptando-a para que a catatua use todo o tempo, que existe entre dois *onsets,* para desempenhar um movimento.

Depois do modelo de testes baseado na catatua, avancei para o modelo final de robô que tem por objectivo realizar alguns dos movimentos do cancan. O robô construído tem a forma de duas pernas com flexibilidade nos joelhos, coxas e cintura, necessitando para isso de utilizar cinco motores. Este modelo realizava um movimento de uma coreografia sempre que era identificado um *onset*. No entanto, durante o desenvolvimento, notei que os *onsets* não seriam o evento musical ideal para um robô que tinha por objectivo fazer movimentos precisos. Assim, avancei para um novo método de análise de som que permitia fazer detecção de *beats*, que são a unidade básica de tempo numa música.

A integração de *beats* no sistema, permitiu que o robô realizasse movimentos no ritmo correcto da música. Assim, a versão final do robô tem um mecanismo com duas pernas e uma cintura que desempenha movimentos de uma coreografia ao ritmo da música que é dado pela detecção de *onsets*.

1.5 Material utilizado

Desde o inicio que partimos para a escolha do sistema de análise de som e para a definição dos mecanismos que iam ser utilizados para construção do robô. A escolha inicial recaiu sobre o sistema Marsyas para análise de som e sobre o Kit Lego Mindstorms NXT para a construção do robô.

A escolha do Marsyas, deveu-se às capacidades descritas na documentação do sistema que davam conta da possibilidade de analisar uma música em tempo real identificando um conjunto de eventos sonoros úteis para a colocar um robô a dançar.

O Kit Lego Mindstorms NXT, não terá sido a melhor escolha possível, pois as capacidades do Kit são bastante limitadas, no entanto, um robô melhor para por em prática esta abordagem teria um custo elevado. Estes Kits da Lego têm tido bastantes aplicações em vários laboratórios de robótica e ainda que a sua precisão seja baixa, a relação capacidade/custo é bastante atractiva.

Nas subsecções 1.3.1 e 1.3.2 será feita uma pequena descrição do Marsyas e do Kit Lego Mindstorms NXT para que seja possível compreender melhor o trabalho descrito neste documento.

1.5.1 Marsyas

É uma framework que teve a origem do seu nome numa figura mitológica grega. O nome vem de "*MusicAl Research SYstem for Analysis and Synthesis*", MARSYAS. Foi implementado numa arquitectura de cliente-servidor, sendo o servidor totalmente desenvolvido em C++ enquanto o cliente foi desenvolvido em Java. A função do cliente é apenas fornecer uma interface gráfica de utilização. No caso do nosso sistema a parte do cliente não foi utilizada, sendo substituída pelo módulo de controlo do robô.

O Marsyas foi desenvolvido por George Tzanetakis com a ajuda de estudantes e investigadores de vários locais do mundo tendo sido integrado em vários projectos. A Framework permite realizar experiências com sinais de áudio e fazer a extracção de informação de músicas. Tem disponíveis, várias funcionalidades das quais se destacam para esta dissertação, a detecção de *onsets*, a detecção de *beats* e a previsão de *beats*, todas elas feitas em tempo real.

1.5.2 Kit Lego Mindstorms NXT

Lançado em Agosto de 2006, o Lego Mindstorms NXT tinha por objectivo proporcionar um robô composto por estruturas Lego que fosse facilmente utilizado por crianças. Deste então os kits Lego Mindstorms NXT transformaram-se num fenómeno de popularidade. Este fenómeno deve-se em grande parte à enorme flexibilidade de conceitos em que o kit se pode aplicar. Apesar de ser considerado por alguns um brinquedo de crianças, parece que os limites dos projectos desenvolvidos com estes kits são definidos apenas pela imaginação.

Um kit destes é composto por uma maleta com peças Lego Technic, um bloco programável NXT (*brick*), sensores, servo-motores, bateria recarregável, conversor de energia e software de programação NXT. O *brick* contém um processador Atmel 32-bit ARM, três portas de saída digital que são ligadas aos servo-motores, quatro portas de entrada onde são ligados os sensores, um *display* tipo matriz, um altifalante, uma porta de comunicação USB 2.0 e uma porta de comunicação Bluetooth. Os sensores disponíveis no kit são um sensor de ultra-som, um sensor de som, um sensor de luz e um sensor de contacto. Na figura 1.1 pode ver-se parte do kit.

Para o desenvolvimento dos modelos de robô apresentados neste documento, não foram utilizados sensores físicos, apenas sensores internos ao robô que permitem obter o posicionamento dos motores. A não utilização de sensores como o de som deve-se ao

facto da sua baixa qualidade de percepção do sinal. Logo de inicio percebi que não seria possível identificar os eventos musicais com os dados recolhidos por esse sensor e parti para uma abordagem em que a análise de som passou a ser feita no computador como é descrito na secção do módulo de análise de som.

Figura 1.1: A figura mostra um *brick* ligado a três servo-motores e quatro sensores.

1.5.3 iCommand

O iCommand foi lançado em 2006 e é uma API Java, desenhada para controlar um *brick* através de uma conexão Bluetooth. Esse *brick* precisa de ter o *firmware* padrão da Lego NXT de forma a poder receber comandos Java.

Esta API permite:

- Estabelecer ligações Bluetooth com o *brick*;
- Ler os valores dos sensores do robô;
- Controlar os motores do robô;
- Controlar a coluna de som do robô;
- Ler informação do estado da bateria e do *brick*.

Das funcionalidades acima descritas torna-se importante destacar no âmbito deste trabalho, a forma como são controlados os motores do robô. As funções de controlo dos motores permitem aceder a informações como a posição relativa, a velocidade ou o ID do motor. No entanto, a posição relativa, é dada numa unidade com o nome de *rotation count*, pois o que esta devolve é a posição do tacómetro em termos de quanto ele rodou em relação à posição inicial. Este *rotation count* será utilizado no trabalho para definir que posição os motores devem assumir para desempenhar um movimento.

1.6 Contribuições científicas

A contribuição científica desde documento foi escrita sobre a forma de artigo.

Esse artigo foi submetido e aceite no "*Workshop on Intelligent Systems and Applications*" (WISA 2010), que estava inserido na quinta Conferencia Ibérica Sistemas e Tecnologias de Informação (CISTI). O artigo [5], com o título "*The First Steps of Robotic Cancan*" foi apresentado a 19 de Junho de 2010, em Santiago de Compostela, Espanha.

1.7 Organização do documento

Este documento está organizado da seguinte forma:

- Capítulo 2 – Estado da Arte
- Capítulo 3 – Modelo Inicial (Catatua)
- Capítulo 4 – Modelo Final (Pernas Cancan)
- Capítulo 5 – Conclusões e trabalho futuro

Capítulo 2

Estado da Arte

2.1 Dança Robótica

Ao longo do tempo, têm vindo a ser desenvolvidos vários tipos de robôs dançarinos. Os investigadores que os vêm a desenvolver tentam cada vez mais aperfeiçoar as técnicas antigas de dança robótica para que os seus robôs dancem de forma semelhante à dos humanos.

Nesta secção são apresentadas algumas das soluções atingidas por estes investigadores da robótica na dança.

Em 2000, Riley et al. [6] utilizam um robô humanóide (figura 2.1) com capacidade de 30 graus de movimento para reproduzir movimentos semelhantes aos de um humano. Este robô, apesar da elevada quantidade de graus de movimento necessitava de estar fixo pois não tinha noções de equilíbrio. O objectivo do trabalho desenvolvido por Riley et al. era conseguir capturar movimentos de dança através de *motion capture,* que seriam aplicados no robô humanóide que eles utilizaram. No entanto, esta abordagem não utiliza análise de som e identifica problemas no tempo de reacção do robô devido à enorme quantidade de movimentos que era necessário fazer em simultâneo.

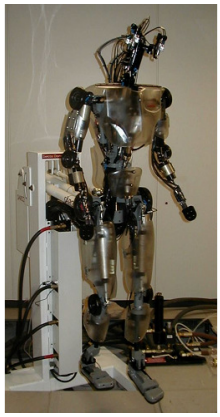


Figura 2.1: Robô humanóide utilizado por Riley et al. (retirado de [6])

Em 2002, Nakazawa, Nakaoka et al. [7] utilizam o robô bípede HRP-2 (Figura 2.2), dançar músicas tradicionais japonesas. Para isso, eles desenvolveram um método de treino de aprendizagem por observação. Neste método o robô começa por aplicar um sistema de captura de movimento para aprender a imitar os movimentos de um humano, sendo que a posteriori imita esses movimentos, neste caso para dançar as referidas músicas tradicionais japonesas. Ainda que este método ofereça uma grande flexibilidade de movimentos, o robô não consegue determinar de forma autónoma quando executar cada movimento, ou seja, o robô limita-se a imitar o que aprendeu e não interpreta a música.

Figura 2.2: Robô HRP-2 (retirado de [7])

Em 2003, Kosuge et al [8] criam uma abordagem na qual um robô é capaz de acompanhar um humano numa dança de salão. Esta abordagem foca-se na interacção com um humano que guia o robô através de movimentos corporais. Esta abordagem foi mais tarde desenvolvida para prever intenções do humano que interage com o robô. Ao robô foi dado o nome de Ms DanceR (*Mobile Smart Dance Robot*), no entanto, este robô apenas seguia o humano, não fazendo qualquer tipo de análise de som.

Em 2005, Tanaka et al. [9][10] desenvolveram um software para o robô, Sony Dream Robot QRIO (Figura 2.3), que permitia que o robô interagisse com crianças de forma a imitar os seus movimentos. Tal como a abordagem acima descrita também esta utiliza a imitação, no entanto, em vez de utilizar aprendizagem por observação utiliza uma politica de "*rough but robust imitation*", ou seja, ele não aprende com os movimentos, ele simplesmente imita os movimentos da pessoa com a qual está a interagir. Esta abordagem também não oferece a possibilidade de colocar o robô a dançar de forma autónoma ao ritmo da música.

Figura 2.3: Sony Dream Robot 'QRIO' (retirado de [9])

Em 2007, Shinozaki et al. [11] elaboraram um estudo sobre o entretenimento da dança utilizando robôs. Para o estudo, eles fizeram uma pesquisa relativa aos robôs que dançam e tentaram clarificar em termos de entretenimento como funcionava a interacção entre humanos e robôs.

Neste estudo, foram feitas comparações entre sistemas que funcionam em tempo real e sistemas que não funcionam em tempo real. Ambas as abordagens tinham as suas vantagens e desvantagens. A principal vantagem do tempo real é a sensação de que o robô está a agir intuitivamente, mas isto provoca a desvantagem de que os erros na dança se tornam mais comuns. Já os sistemas que não funcionam em tempo real trazem como principal vantagem o perfeccionismo no movimento, mas por outro lado provoca a perda daquilo que o tempo real dá como vantagem, a sensação de acções intuitivas.

Shinozaki et al. terminam o estudo, apresentando como trabalho futuro um caso de uso da dança robótica aplicada ao robô RoboNova onde propõem a aplicação de um sistema de identificação de tempo e beats para gerar uma sequência de movimentos num determinado ritmo. Os movimentos para o robô são adquiridos através de *motion capture* aplicada sobre os movimentos de um dançarino profissional. Os movimentos capturados são bastante suaves, permitindo assim que o robô os desempenhe garantindo o equilíbrio.

No nosso trabalho, a musica do Cancan exige um ritmo bem mais acelerado do que o aplicado nesta versão e como tal trabalhar o equilíbrio seria uma tarefa árdua na qual seria necessária uma intensiva pesquisa na área da física.

Ainda em 2007, é apresentada uma continuação do estudo de Shinozaki et al, pelos mesmos autores [12]. Esta continuação do estudo já tem por objectivo desenvolver um sistema que permita a um robô dançar. Tal como eles haviam definido no trabalho anterior, o sistema foi aplicado a um robô RoboNova, que devido à sua enorme quantidade de motores (seis motores em cada perna, quatro motores em cada braço um motor na cintura e um motor na cabeça) tem uma vasta capacidade de movimentos.

Shinozaki et al, optaram por tentar um ritmo de música um pouco mais acelerado tendo escolhido o *hip hop* por ter movimentos básicos simples. No entanto depararam-

se com o problema do equilíbrio que foi resolvido adicionando uma postura de equilíbrio para o robô. A dança ficou um pouco limitada, pois entre dois movimentos tornava-se obrigatório que o robô passasse pela sua postura de equilibro de forma a garantir, que de um movimento para outro ele não caía. Esta abordagem abstraiu-se da análise de som focando apenas a dança e os movimentos do robô.

Também em 2007, Yoshii et al. [13] usaram um robô Honda ASIMO (Figura 2.4) para desenvolver um sistema que faz o robô bater os pés ao som das batidas (*beats*) da música. Este robô tem a capacidade de receber som e em tempo real prever quando vai receber um *beat*. Esta capacidade de percepção representa um grande avanço para a criação de robôs dançarinos inteligentes, pois para que seja possível dançar ao ritmo da música é necessário saber prever quando vão dar-se certos elementos sonoros.

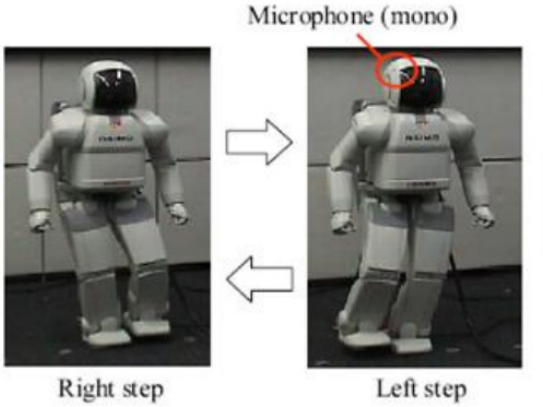


Figura 2.4: Robô Honda ASIMO (retirado de [13]) a bater os pés ao ritmo da música

Ainda em 2007, Aucoutuier et al. [14] criaram um robô chamado MIURO, que ao contrário dos até aqui apresentados não é bípede. Eles criaram neste robô uma dinâmica básica através de um tipo de caos para deixar que os comportamentos emergissem de forma autónoma.

Em 2008, Kozima et al. [15], utilizam um robô Keepon (figura 2.5), para uma pesquisa de terapia e entretenimento. O Keepon é um robô bastante simples com um corpo deformável e com a forma de um pinto. Este robô é muito utilizado para interacção com crianças permitindo que estas tenham melhor desenvolvimento psicológico. Um dos testes que foi feito para interacção foi precisamente na área da dança na qual o robô dançava ao ritmo da música atraindo as crianças para que estas desempenhem movimentos semelhantes aos dele. Contudo, devido a imperfeições do robô, este não consegue manter-se sincronizado com a música.

Figura: 2.5: Robô Keepon (retirado de [15])

Em 2008, Oliveira et al. [16][17] apresentam uma arquitectura para um sistema robótico que reage à música em tempo real. O robô é construído usando um Kit Lego Mindstorms NXT (Figura 2.6) e simula percepção rítmica através de uma aplicação chamada Marsyas [3]. A aplicação Marsyas consegue detectar o início das notas musicais (*onsets*) e realizar a comunicação com o módulo de controlo do robô para o informar dos eventos detectados. Contudo, o robô não reage apenas aos eventos musicais, reage a conjunção de estímulos formada por eventos rítmicos e sensoriais. Os eventos rítmicos têm a ver com a quantidade de *onsets* que existem num determinado intervalo de tempo e os sensoriais são relacionados com a cor do chão e a proximidade a obstáculos.

Respondendo aos eventos rítmicos e sensoriais, o robô realiza movimentos da coreografia. A coreografia é bastante limitada pois todos os motores têm uma rotação total possibilitando que o robô realizasse movimentos que não se assemelham a movimentos de um humano, como por exemplo várias rotações de 360º na cintura, várias rotações de 360º do pescoço… A juntar a isto há ainda a questão da análise de som. Nesta abordagem, Oliveira et al. propõem um sistema em que o robô dança identificando a quantidade de *onsets* que existe num determinado tempo. Isto permite que o robô se mova mais rápido ou mais devagar conforme o ritmo da música, mas não necessariamente ao ritmo da música. Para movimentos como os que proponho no Cancan Robótico, a abordagem de Oliveira et al. é incompleta pois o Cancan Robótico precisa de movimentos coreográficos bem definidos e desempenhados no momento em que se identifica um evento musical.

Figura 2.6: Robô Lego NXT de Oliveira et al (retirado de [16])

Em 2009, Grunberg, Ellenberg et al. [18], desenvolveram uma abordagem capaz de fazer um robô dançar de forma autónoma. O objectivo desta abordagem era criar um protótipo para auxiliar coreógrafos. Assim, eles desenvolveram um algoritmo que consegue identificar algumas características do som como *beats*, tempo e estilo, e aplicaram-no a um RoboNova (Figura 2.7). O RoboNova é um robô não só com uma grande liberdade de movimentos quer a nível dos membros do robô quer a nível do próprio tronco como também um robô com boas características para trabalhar em termos de equilíbrio. Isto faz deste robô um forte candidato a robô bailarino.

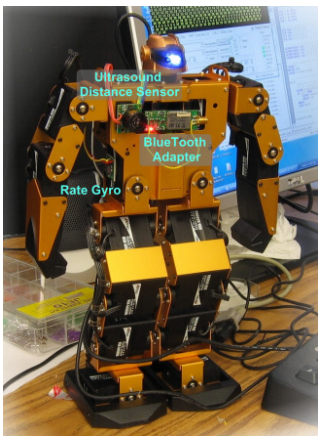


Figura 2.7: Humanóide RoboNova (retirado de [18])

2.2 Análise de som

Uma música tem muita informação disponível que pode ser extraída através da análise do som. Alguns investigadores têm trabalhado nesta área tentado obter vários tipos de informação dos sinais sonoros. No contexto deste documento, é importante referirmos alguns dos sistemas que têm vindo a ser utilizados por outros investigadores

para conseguir colocar robôs a dançar. Sendo assim, esta secção está organizada em 2.2.1 detecção de *onsets* e 2.2.2 detecção de *beats*.

2.2.1 Detecção de *onsets*

Detecção de *onsets* é uma forma de análise de som desenvolvida para conseguir detectar quando existe uma alteração no sinal sonoro. Esta alteração no sinal representa a entrada de uma nota musical.

Em 2005, Bello et al [19], apresentaram um tutorial para detecção de *onsets*. Nesse trabalho eles explicaram como é possível extrair informação de uma música através da identificação dos tempos de *attack, transient, decay* e *onset* (Figura 2.8).

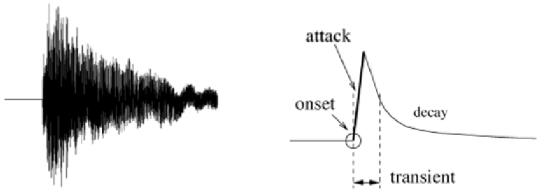


Figura 2.8: *Onset*, *attack*, *transient* e *decay* de uma nota (retirado de [19])

Cada um dos elementos identificados tem diferentes importâncias apesar de todos serem utilizados na detecção dos *onsets*.

• O *attack* de uma nota é o intervalo durante o qual a amplitude da nota cresce criando excitação no sinal.

• O *transient* é um pouco mais difícil de descrever, mas de uma forma simplificada, podemos dizer que o *transient* é o intervalo de tempo durante o qual o sinal evolui de uma forma imprevisível.

• O *decay*, é o intervalo de tempo durante o qual a amplitude reduz e o sinal perde a excitação criada durante o ataque.

• O *onset* de uma nota corresponde ao momento inicial do *transient*. Desta forma podemos dizer que o *onset* marca o tempo em que ocorre um evento sinal, o início de uma nota musical. Obtendo o conjunto de *onsets* de uma musica podemos mais tarde identificar quais destes *onsets* correspondem a, por exemplo, notas graves tornando possível através destas notas identificar o tempo dos *beats*.

Para obter um conjunto de *onsets* de uma música, existem vários algoritmos. Apesar de não haver um algoritmo que possa ser apontado como perfeito, Bello et al descrevem os melhores como uma combinação equilibrada entre três processos: Pré-processamento, redução e selecção de picos (Figura 2.9).

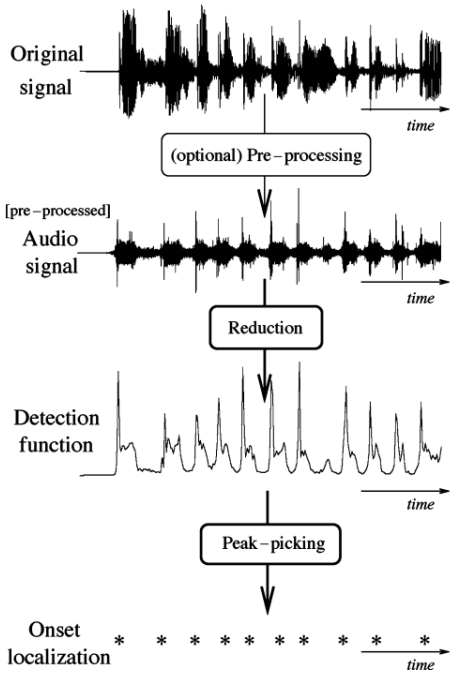


Figura 2.9: Gráfico que demonstra o algoritmo de detecção de *onsets* (retirado de [19])

O pré-processamento é um processo durante o qual a transformação do sinal serve para acentuar ou atenuar determinadas características do sinal. É normalmente apontado como um processo opcional, mas para um bom sistema de detecção de *onsets* é necessário.

Já a operação de redução, no contexto de detecção de *onsets*, serve para transformar o sinal proveniente do pré-processamento, numa função de detecção (*detection function*) onde se torna possível identificar o *transient*.

Por fim temos a selecção de picos (*peak-picking*) que consegue identificar na função de detecção o conjunto de picos. Após este processo conseguimos obter a localização temporal de cada *onset*.

O método de detecção de *onsets* é muito útil pois permite perceber quando ocorrem os eventos durante a música tornando possível utilizar os tempos dos *onsets* para elaborar uma coreografia.

2.2.2 Detecção de *beats*

A detecção de *beats* consiste na identificação das batidas que são normalmente descritas como "a pulsação da música". Normalmente durante a dança esta detecção de *beats* é realizada pelos bailarinos de forma instintiva. De uma forma natural, as pessoas que ouvem música tendem a bater palmas ou abanar a cabeça ao ritmo da música. Na verdade, as palmas e o abanar da cabeça são reacções intuitivas sincronizadas com os *beats*.

A música é geralmente caracterizada pela repetição de uma sequência de *beats* através dos quais se pode identificar o tempo e ritmo da música.

Em 1990, Allen e Dannenberg [20] apresentam uma forma de identificar os *beats* de uma música em tempo real. Para isso eles analisam uma sequência de sons à qual aplicam várias interpretações. Assim, eles conseguem prever o tempo do próximo *beat* utilizando a interpretação que lhes ofereceu melhores resultados.

Em 1995, Goto e Muraoka [21] apresentam uma nova abordagem para detecção de *beats* com maior fiabilidade. Goto e Muraoka fazem o sinal sonoro passar por um processo de detecção de *onsets* seguido de um processo de detecção do som de percussão. Seguidamente, os dados analisados passam por um conjunto de agentes que através do cruzamento dos dados dos *onsets* com os dados da percussão, conseguem não só identificar o tempo em que ocorre um *beat* como também fazer uma previsão do tempo em que se espera que ocorra o próximo.

Com a introdução da detecção de *beats* no meu sistema será possível identificar quais os tempos em que o robô deve executar um passo coreográfico para que um observador consiga visualizar a dança sincronizada com a música.

Capítulo 3

Modelo Inicial (Catatua Dançarina)

O modelo inicial com o nome de Catatua Dançarina serviu como base de testes para o desenvolvimento do robô final. Este modelo tinha apenas por objectivo fazer testes ao sistema escolhido inicialmente para integrar o módulo de análise de som e estudar quais as melhores formas de reagir aos eventos por ele identificados.

Este modelo é baseado nos movimentos de uma catatua que reage instintivamente a eventos musicais [22]. Esta catatua ouve uma música e expressa-se através de um conjunto de movimentos de entre os quais se distingue maioritariamente o movimento de *headbang* e o movimento de levantar a pata ao alto semelhante ao *high kick* da dança Cancan.

Como este modelo é apenas experimental e o seu objectivo é testar a capacidade do robô de responder ao módulo de análise de som, a única preocupação foi reproduzir um único movimento, o *headbang*.

O *headbang* é um movimento de cabeça decomposto em duas acções, a descida da cabeça e a subida da cabeça. Ambas as acções devem ser desempenhadas consecutivamente em resposta a um evento musical. Para isso, este modelo utiliza um sistema de análise de som que lhe permite identificar *onsets.* Assim, sempre que um *onset* é identificado, o robô inicia o movimento de *headbang* que deve ser realizado a uma velocidade que permita que o robô esteja pronto para realizar um novo movimento, momentos antes da ocorrência de um novo *onset*.

Para a Catatua Dançarina foram realizadas duas versões semelhantes, que diferem na forma como é feita a reacção aos *onsets*. Como referido, inicialmente o propósito deste modelo era verificar como seria o comportamento do sistema de análise de som quando em comunicação com um módulo feito em Java para controlo do robô, e quais as capacidades do Kit Lego para responder às necessidades do Cancan Robótico. No entanto, devido a dificuldades encontradas na precisão dos motores ficou perceptível que seria necessário atribuir algum conhecimento ao robô para garantir que a coreografia seria desempenhada de uma forma mais correcta. Esta necessidade levou a

que fosse desenvolvida uma segunda versão da Catatua Dançarina que inclui conhecimento dos intervalos de tempo entre eventos musicais que simulam uma espécie de previsão de eventos.

Assim, as duas versões desenvolvidas foram a Catatua Dançarina que reage a *onsets* e a que reage a *onsets* com conhecimento prévio. Nas secções seguintes será explicado como foi desenvolvida cada uma das versões.

3.1 Reagindo a *onsets*

Nesta versão, o robô não sabe quanto tempo tem de intervalo entre dois *onsets* consecutivos. Isto significa que o robô não sabe quanto tempo tem para completar um determinado movimento e por consequência o movimento é sempre desempenhado à mesma velocidade. O comportamento do robô nesta versão é totalmente baseado na reacção aos eventos detectados.

3.1.1 Arquitectura do sistema

A arquitectura do sistema está dividida em três partes principais, o módulo de análise de som, o módulo de controlo do robô e o robô.

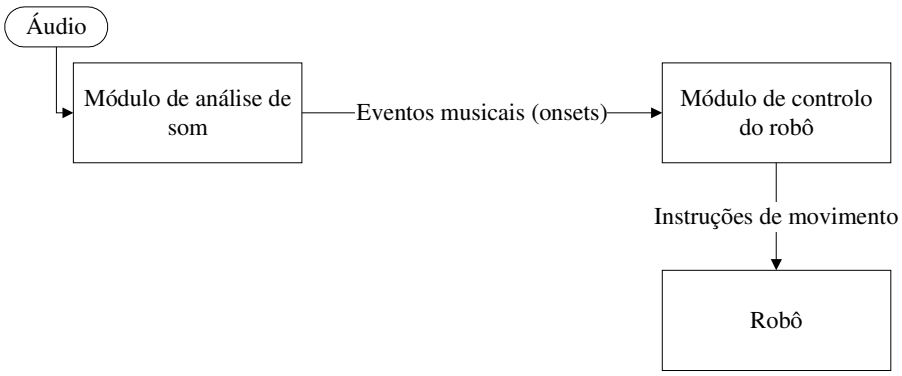


Figura 3.1: Arquitectura do sistema da versão inicial da Catatua Dançarina.

Nas secções seguintes irei explicar como foram elaboradas cada uma das partes.

3.1.2 Módulo de análise de som

A Catatua Dançarina utiliza o módulo de análise de som para reprodução da música e detecção de *onsets* em tempo real. Como tal, este módulo é um sistema com o nome de Marsyas que permite fazer a detecção dos *onsets* que são necessários para identificar o ritmo da música.

Para conseguir colocar este sistema a funcionar foram encontradas muitas dificuldades e despendido muito tempo, pois a documentação do sistema estava mal

elaborada e muitas das funcionalidades que se diziam implementadas não funcionavam de forma correcta. Felizmente a funcionalidade de detecção de *onsets* estava operacional e praticamente preparada para operar como módulo de análise de som. Para colocar a funcionalidade totalmente apta para cumprir a função de comunicar ao módulo de controlo do robô quando ocorria um evento musical (*onset*), foi necessário colocar a aplicação a fazer comunicação por *sockets* com o módulo de controlo do robô. Foi ainda necessário criar um mecanismo através do qual fosse possível informar o módulo de controlo do robô quando a música terminasse. Desta forma, ao terminar a música, o módulo de análise de som podia terminar a sua execução e informar o módulo de controlo do robô que pode terminar também a actividade de dança.

Ao ser iniciado, o sistema de análise de som fica em espera até que lhe seja feito um pedido de estabelecimento de ligação por *socket TCP*. Ao receber o pedido de ligação, o sistema envia um aviso informando que a ligação foi aceite e que vai ser iniciada a reprodução e análise da música. É iniciada a reprodução e análise que resulta na detecção de *onsets*. Sempre que é detectado um *onset* durante a reprodução da música, é enviada uma informação por *socket* informando a ocorrência desse evento. Quando a música termina é enviado o último aviso por *socket* que dá conta que o módulo de análise de som vai deixar de estar em execução por ter terminado a reprodução da música. Todos os dados enviados pela ligação *socket* devem ser recebidos pelo módulo de controlo de robô para que este decida os instantes em que deve pedir ao robô para actuar.

3.1.3 Construção do robô

Devido à escassa quantidade de movimentos necessária para realizar os testes a que este modelo se propunha, este robô foi construído utilizando um Kit Lego Mindstorms NXT. Desse Kit foram utilizadas peças Lego Technic para construir o formato da cabeça de uma catatua, um motor, que permitia mexer as peças que formavam a cabeça, e um *brick* que é o responsável por receber instruções vindas do módulo de controlo do robô e coordenar o movimento do motor. Para uma melhor percepção pode ver-se uma imagem que representa a estrutura física do robô na figura 3.2.

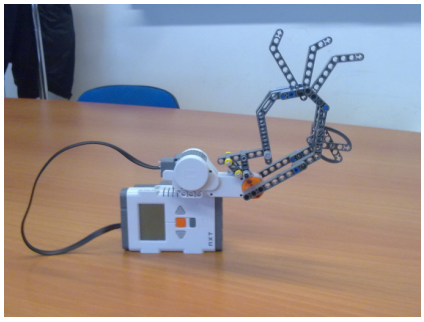


Figura 3.2: Catatua Dançarina

Na construção do robô, tive o cuidado de colocar algumas limitações físicas de forma a evitar que movimentos muito rápidos do motor conduzissem a situações que fisicamente seriam impossíveis para uma catatua. Por isso mesmo existem algumas peças colocadas ao nível do pescoço que impossibilitam o movimento da cabeça demasiado para trás e demasiado para a frente.

3.1.4 Módulo de controlo do robô

O módulo de controlo do robô é responsável por enviar instruções de movimento para o robô sempre que recebe informação de que ocorreu um evento musical identificado pelo módulo de análise de som.

Este módulo estabelece um canal de comunicação por Bluetooth com o *brick* do robô, por onde vai enviar as instruções de movimento e cria ainda outro canal de comunicação por *socket TCP* com o módulo de análise de som, de onde irá receber informação sobre o instante em que ocorre um *onset*.

Visto que o único objectivo era testar a comunicação e o módulo de análise de som, o funcionamento do módulo de controlo do robô é relativamente simples. Sempre que é recebido um *onset*, o robô desempenha um movimento de *headbang*. Esse movimento de *headbang* corresponde a duas acções do motor do robô. Uma das acções é uma rotação de aproximadamente 70º no sentido positivo (descer a cabeça), e a outra é uma rotação de aproximadamente 70º no sentido negativo (subir a cabeça). Em termos da API iCommand 70º correspondem a cerca de 62 *rotation counts.*

Com os testes feitos a esta versão, foi detectado um obstáculo que era necessário resolver. Esse obstáculo vinha do tempo disponível para executar um movimento. Quando uma pessoa dança, tem uma intuição que lhe permite saber quanto tempo decorre desde o inicio de um passo até ao fim desse passo. Essa intuição funciona como uma previsão que garante que a pessoa dança ao ritmo da música. No caso do robô, tudo aquilo que o módulo de análise de som devolve é a informação de que estava a ocorrer um *onset*. Assim, sendo a análise feita em tempo real e não havendo um algoritmo disponível para fazer a previsão de *onsets*, não havia uma forma de garantir que o robô completasse um movimento antes de ser enviada a instrução de começar o movimento seguinte. Isto fazia com que o robô perdesse o ritmo.

Para ultrapassar este obstáculo avancei para uma nova versão do modelo Catatua Dançarina que usava conhecimento prévio para simular a previsão de *onsets*.

3.2 Reagindo a *onsets* com conhecimento prévio

O objectivo desta nova versão da Catatua Dançarina é resolver o problema dos movimentos incompletos. Para uma dança como o cancan, é importante que os

movimentos tenham a maior precisão possível e como tal, tentei que este modelo da Catatua Dançarina realizasse os movimentos aproveitando todo o tempo que tinha para os completar. Para isso era necessário conseguir fazer a previsão do tempo em que um *onset* iria ocorrer. Visto não existir nenhuma funcionalidade que permitisse realizar essa previsão, parti para uma abordagem em que essa previsão fosse simulada através de uma análise prévia da música.

3.2.1 Arquitectura do sistema

A arquitectura do sistema é semelhante à da versão anterior, no entanto, o módulo de análise de som faz uma análise prévia para fornecer dados relativos ao intervalo de tempo entre cada par de *onsets* consecutivos.

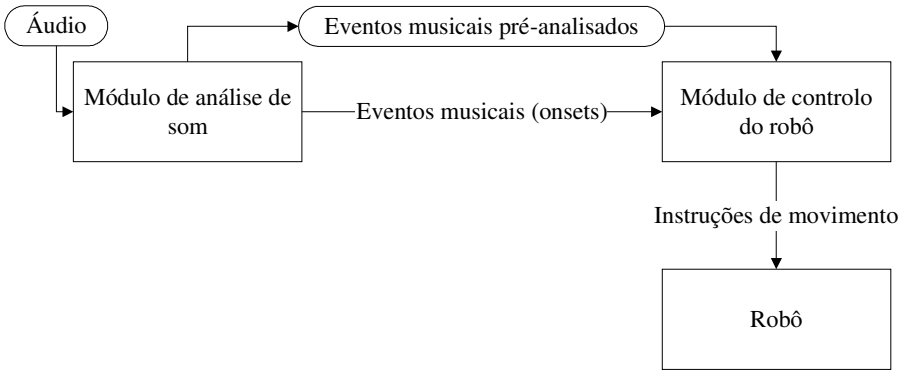


Figura 3.3: Arquitectura do sistema da segunda versão da Catatua Dançarina.

Nas secções seguintes irei explicar as alterações que foram feitas ao módulo de análise de som e ao módulo de controlo do robô. A construção do robô não será explicada novamente, pois não sofreu alterações.

3.2.2 Módulo de análise de som

À semelhança do que foi usado anteriormente na catatua, também nesta versão é utilizado o sistema de análise de som Marsyas. No entanto, foi feita uma alteração à forma como o módulo de análise de som funcionava, para que fosse possível fazer uma pré-análise à música.

Nesta nova versão, o módulo de análise de som começa por fazer uma análise a toda a música, produzindo um ficheiro com os intervalos de tempo entre cada par de *onsets* consecutivos. Após criar este ficheiro, o módulo inicia um funcionamento igual ao descrito na versão anterior, informando o módulo de controlo do robô sempre que um *onset* ocorre.

3.2.3 Módulo de controlo do robô

A nova versão da Catatua Dançarina, foi elaborada para resolver o problema da falta de conhecimento que originava uma falta de adaptação da velocidade dos movimentos ao ritmo da música.

A novidade neste módulo foi a possibilidade de fazer variar a velocidade do motor para que a acção deste fosse realizada ao ritmo da música. Para isso, foi criado um parâmetro com o nome de velocidade que assume valores conforme o intervalo de tempo que existia até que ocorresse o próximo *onset*. Tendo em conta que o movimento de *headbang* é decomposto em duas acções (subida e descida), durante o *headbang* o motor assume duas velocidades distintas, velocidade de subida e velocidade de descida. Cada uma dessas velocidades é calculada para ser desempenhada em parte do tempo de intervalo entre dois *onsets*. Pela observação da catatua que serviu de base para todo o modelo, reparei que o movimento de descida da cabeça era realizado mais rapidamente do que o movimento de subida, como tal reservei 4/10 do intervalo de tempo para executar a descida da cabeça e os restantes 6/10 para executar a subida.

Tendo em conta que a função que permite ao iCommand definir a velocidade, atribui ao motor uma velocidade em graus por segundo, a fórmula aplicada para a descida da cabeça é dada por:

$$v = \frac{d}{\frac{4}{10} \times \Delta t}$$

Já a fórmula aplicada para a subida da cabeça é dada por:

$$v = \frac{d}{\frac{6}{10} \times \Delta t}$$

Em ambas as fórmulas apresentadas acima, o ‘d’ corresponde a distância a percorrer em rotação, neste caso 62 *rotation count* (aproximadamente 70º). O ‘v’ corresponde à velocidade que vai ser assumida pelo motor e o Δt corresponde ao intervalo de tempo que é lido do ficheiro fornecido pelo módulo de análise de som. Assim, quanto maior é o intervalo de tempo menor vai ser a velocidade que o motor assume para completar um movimento.

Esta melhoria provocou outro obstáculo que proveio das velocidades variáveis do motor. O problema é que quanto mais rápido o motor se move, maior é a distância de travagem. Assim, como o robô demorava mais tempo a travar por vezes os motores entravam em situações de bloqueio por embaterem nas limitações físicas. Para evitar que o motor ficasse bloqueado e em esforço, por não conseguir parar na posição para onde se estava a mover, foi desenvolvida uma função que realiza a verificação da

posição do motor e do deslocamento que este estava a realizar. Essa função permitiu evitar situações de bloqueio.

Nesta nova versão consegui observar um comportamento bastante satisfatório no modelo da Catatua Dançarina que proporcionou uma boa base de trabalho para o modelo final do robô.

3.3 Discussão de resultados

Com as experiências feitas com a Catatua Dançarina, foi possível verificar principalmente que o sistema de análise de som estava incompleto. Compreendi então, que para atingir os resultados esperados, não deviam ser usados os *onsets*, mas sim dos *beats*. Antes de partir para o modelo seguinte tentei utilizar as funções de detecção e previsão de beats do Marsyas, contudo estas funções encontravam-se incompletas e mal documentadas não sendo possível utilizá-las.

Para terminar, notou-se que a utilização de apenas um Kit Lego Mindstorms NXT iria ser uma limitação pois para um modelo de pernas era necessário utilizar pelo menos dois motores por perna. No entanto, devido à limitação de um *brick*, que apenas pode ter ligados a si três motores, foi necessário introduzir a utilização de mais que um *brick* e como tal passei a utilizar dois Kits Lego.

As conclusões tiradas deste modelo permitiram que o modelo final tivesse uma base de partida mais sólida e assim fosse planeado de uma forma mais elaborada.

Capítulo 4

Modelo Final (Pernas Cancan)

O cancan é uma dança composta uma grande variedade de movimentos envolvendo todas as partes do corpo humano: pernas, braços, tronco, cabeça... Sendo impossível com o material disponível elaborar um modelo capaz de reproduzir todo o repertório, seleccionei um subconjunto de movimentos que apenas envolvem as pernas e o tronco, e que são paradigmáticos do cancan como os *kicks* e *highkicks.* Por esta razão chamei ao modelo final do robô, Pernas Cancan.

Apesar da Catatua Dançarina ter oferecido uma boa base de trabalho, tiveram de ser feitas algumas alterações. As alterações foram feitas ao nível:

- Da construção do robô, na qual foram acrescentados mais motores para possibilitar uma maior variedade de movimentos, obrigando a uma reformulação no módulo de controlo de robô;
- Do módulo de análise de som, pois o robô passou a reagir aos *beats* e não aos *onsets*, dançando em tempo real de acordo com uma coreografia, conhecendo o intervalo de tempo entre dois *beats* consecutivos para realizar cada movimento, através de uma análise previa dos *beats*;
- E finalmente, devido a limitações dos motores foi necessário atrasar o próprio ritmo da música.

Para descrever este modelo, este capítulo está dividido nas subsecções: Arquitectura do sistema, Módulo de análise de som, Construção do robô, Coreografia, Módulo de controlo do robô e Discussão dos resultados.

4.1 Arquitectura do sistema

A arquitectura do sistema está dividida em três partes principais, o módulo de análise de som, o módulo de controlo do robô e o robô. No entanto, apesar das semelhanças com o modelo da Catatua Dançarina, as principais diferenças estão no

novo conteúdo do ficheiro de eventos musicais pré analisados e na integração de um ficheiro coreográfico (figura 4.1).

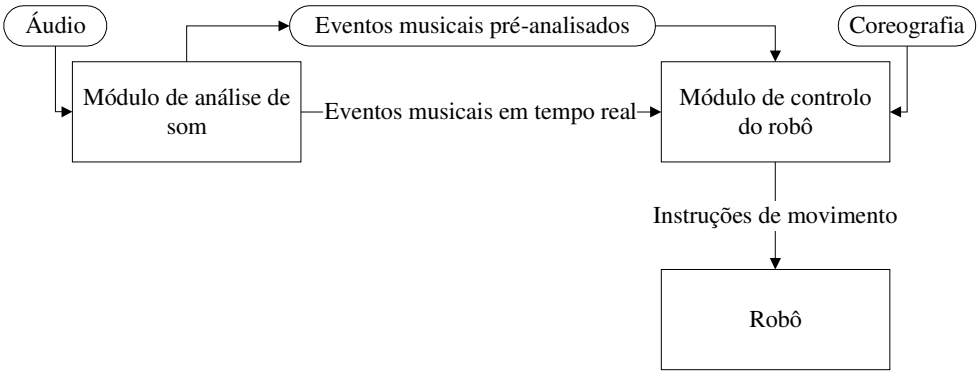


Figura 4.1: Arquitectura do sistema do modelo final

Nas secções seguintes irei explicar como foram elaboradas cada uma das partes, e como são integrados os ficheiros de coreografia e eventos musicais pré analisados.

4.2 Módulo de análise de som com Beatroot

A resposta aos *onsets* não era adequada ao objectivo que pretendia atingir e, como tal, foi necessário elaborar um novo módulo de análise de som baseado na detecção de *beats* (Figura 4.2). Infelizmente, as funcionalidades de detecção e previsão de *beats* do Marsyas, não estavam operacionais. Após uma pesquisa, consegui identificar outros sistemas de análise de som que conseguiam fazer detecção de *beats*, contudo, estes não permitiam fazer uma análise do sinal sonoro em tempo real. O que seleccionei foi o BeatRoot [23], que foi desenvolvido em Java por Simon Dixon. Para dançar em tempo real e tendo em conta que cada *beat* é também um *onset*, recorri de novo ao detector de *onsets* do Marsyas que é utilizado após a detecção de *beats* feita a priori pelo BeatRoot. Desta forma, o robô não vai reagir a todos os *onsets*, mas apenas àqueles que correspondem a *beats*. Para verificar quais dos onsets são *beats*, recorre-se aos dados produzidos previamente pelo BeatRoot, que são guardados num ficheiro representado no diagrama da figura 4.2 como "eventos musicais pré-analisados" e onde estão expressos os instantes exactos em que os *beats* ocorrem.

O ficheiro com os dados pré analisados é usado pelo módulo de controlo do robô para identificação dos *beats* em tempo real. O ficheiro áudio a ser analisado pelo Marsyas é o mesmo que foi pré-processado pelo BeatRoot.

O Marsyas comunica com o módulo de controlo de robô utilizando *sockets TCP*. Tal como no modelo Catatua Dançarina, o Marsyas é executado em tempo real e é o responsável por reproduzir a música, detectar os *onsets* e enviá-los para o módulo de

controlo do robô. A diferença em relação aos modelos anteriores é que o módulo de controlo do robô faz a comparação do tempo do *onset* com o tempo de um *beat* para escolher quais dos *onsets* enviados pelo Marsyas devem ser interpretados através de um movimento da coreografia.

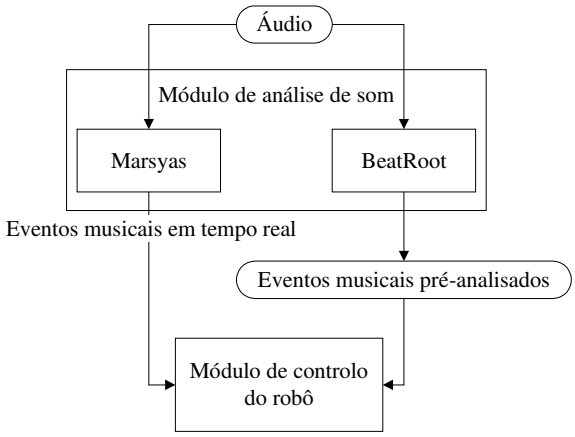


Figura 4.2: Diagrama com a representação da arquitectura do módulo de análise de som.

O funcionamento do BeatRoot, é explicado na subsecção seguinte

BeatRoot

O sistema de detecção de *beats* desenvolvido por Simon Dixon, permite analisar uma música, detectando os instantes em que ocorrem *beats*. Este sistema pode ser decomposto em duas partes principais, *Tempo Indution* e *Beat Tracking,* que são precedidas por uma detecção de *onsets*. A lista de *onsets* passa por um algoritmo de *Tempo Induction* que permite o respectivo agrupamento. Segue-se a detecção de *beats* (*beat tracking*), esta é a parte mais complexa do sistema de Simon Dixon, pois utiliza um conjunto de agentes para determinar os instantes em que ocorrem *beats*. Identificados os *beats*, podemos ver o resultado da análise numa interface gráfica (figura 4.3). Esta interface inclui uma funcionalidade que produz um ficheiro de texto contendo informação sobre os tempos em que ocorrem os *beats* identificados.

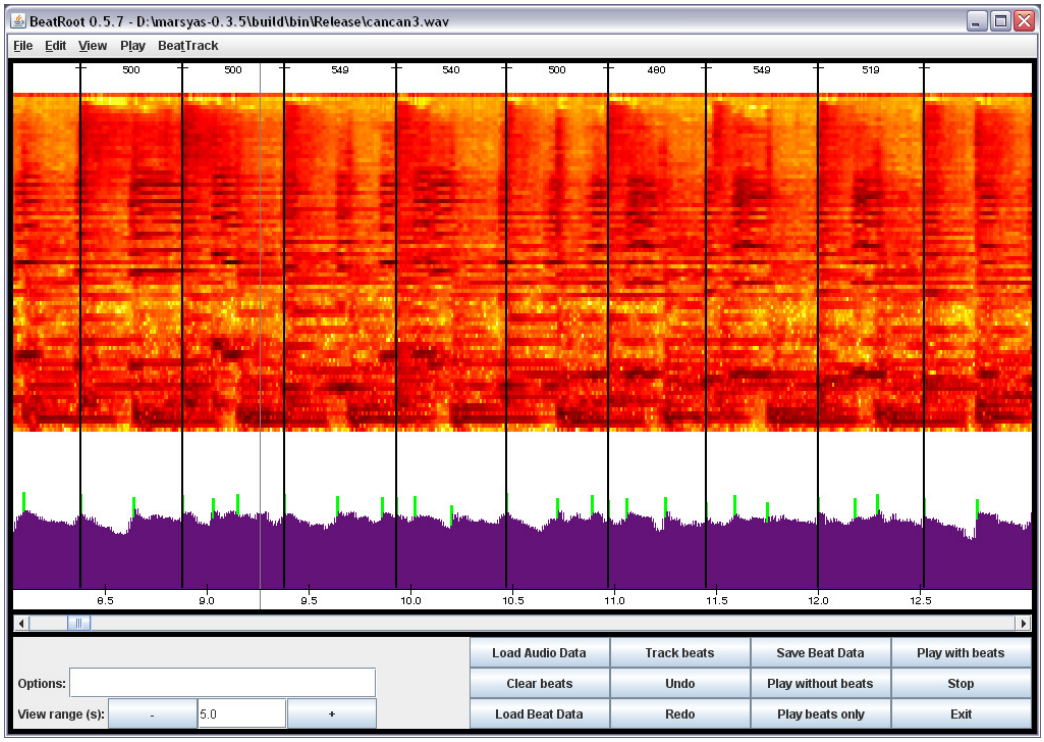


Figura 4.3: Interface gráfica mostrando o excerto de uma música de Offenbach, na qual podemos ver o intervalo entre os *beats* em milissegundos (em cima), tempos dos *beats* (linhas pretas verticais), espectrograma (ao centro), amplitude (em baixo), *onsets* detectados (pequenas linhas verdes sobre o roxo) e o painel de controlo (em baixo).

O ficheiro produzido pelo BeatRoot contem muita informação que não é relevante para o Pernas Cancan. Foi então criado um filtro que limpa os dados não relevantes e deixa apenas os valores correspondentes aos tempos em que ocorrem os *beats* (Figura 4.4).

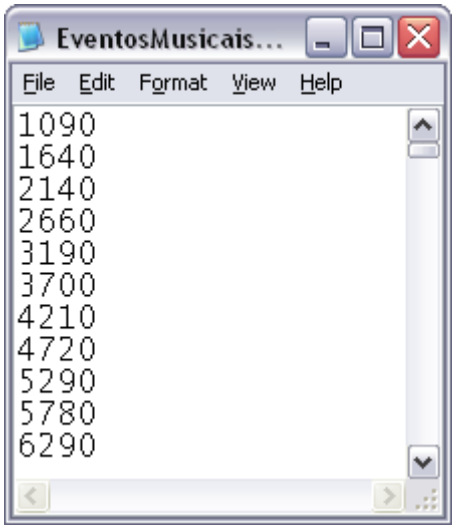


Figura 4.4: Exemplo de um ficheiro com os primeiros *beats* da música de Offenbach.

4.3 Construção do robô

Como referido anteriormente, foi necessário reconstruir o robô para introduzir no robô a capacidade de desempenhar movimentos variados, nomeadamente movimentos que requerem que os joelhos dobrem, as pernas levantem ou a cintura rode.

Este modelo do robô foi construído com dois kits Lego NXT dos quais são utilizados dois *bricks*, cinco motores e algumas peças Lego Technic. Devido à problemática do equilíbrio do robô durante a dança, que exigia estudos sobre física e que seria difícil de aplicar a um robô construído com Lego NXT, optei pela criação de uma base que sustenta o robô para que este não caia. Inicialmente tentei construir a base com as peças do kit, mas rapidamente ficou claro que o peso do robô seria demasiado para a resistência das peças. Optei então por conceber uma base em madeira com capacidade de sustentar todo o robô sem que este caía ou se desequilibre (Figura 4.5).

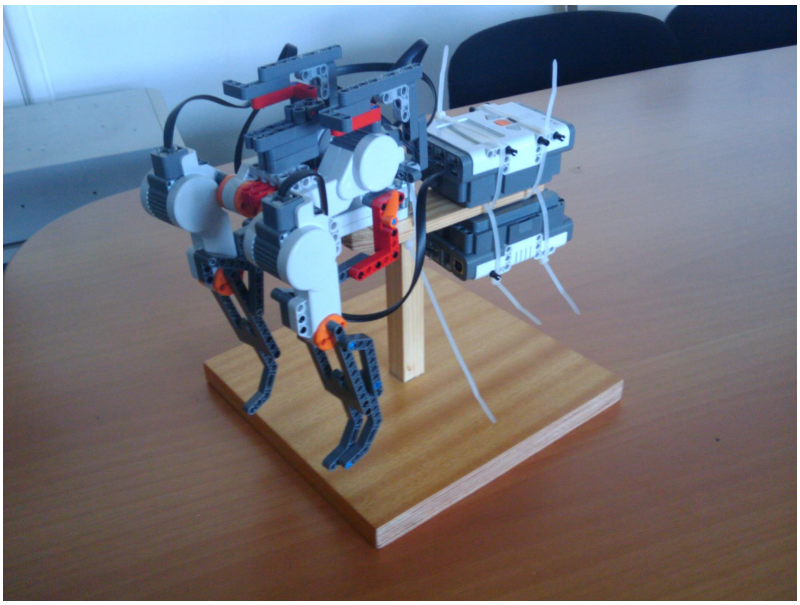


Figura 4.5: Pernas Cancan

Cada uma das pernas do robô está associada a um *brick* diferente e é composta por dois motores. Um motor que possibilita a rotação da perna em torno da anca, e o outro que possibilita a rotação da parte inferior da perna em torno do joelho. Para evitar situações que num humano seriam impossíveis, foram criadas algumas limitações físicas a nível do joelho e das ancas, evitando por exemplo situações em que o joelho dobrasse ao contrário do habitual. Existe ainda um motor que atribui ao robô a capacidade realizar parte de uma rotação em torno do seu eixo vertical. Este motor é controlado pelo mesmo *brick* que controla a perna esquerda (figura 4.6).

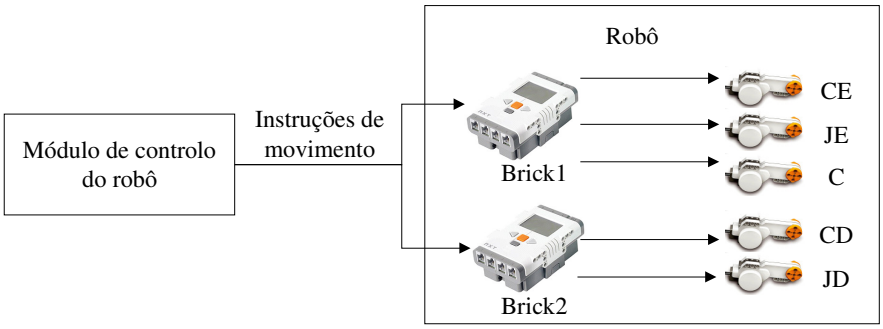


Figura 4.6: Diagrama com a representação da construção do robô.

A base de madeira, apesar de oferecer um bom suporte, retira um pouco de liberdade de movimento ao robô, especialmente no que diz respeito à capacidade de rotação em torno do eixo vertical, pois os cabos que ligam os motores ao *brick* e as próprias pernas batem na madeira se as instruções exigirem uma rotação muito grande. Contudo, esta é a melhor solução que encontrei para o problema do equilíbrio sem que tenha de ser trabalhada em detalhe a questão da física.

4.4 Coreografia

A nova construção do robô tornou possível criar o conjunto de movimentos simples necessário para a coreografia do cancan. Cada um dos movimentos simples é executado por apenas um motor e pode ser utilizado individualmente ou combinado com outros movimentos para criar movimentos compostos. Um movimento composto aparece a partir da combinação de dois ou mais movimentos simples, que devem ser executados por motores diferentes.

Para uma melhor percepção denominei os motores de Coxa Esquerda (CE), Coxa Direita (CD), Joelho Esquerdo (JE), Joelho Direito (JD) e Cintura (C) (Figura 4.7).

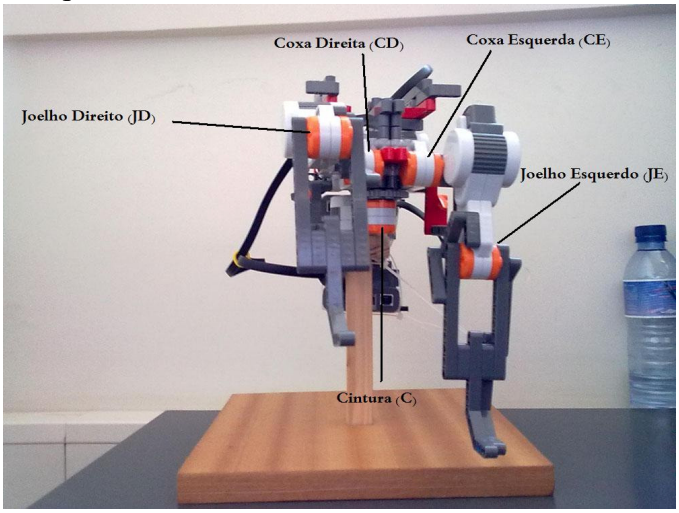


Figura 4.7: Localização dos motores.

Utilizando a nomenclatura dos motores definimos como movimentos simples:

- Dobrar joelho: Movimento aplicado aos motores JE ou JD que faz com que o motor realize uma rotação de aproximadamente 45º. Este movimento simula uma bailarina a dobrar o joelho.
- Esticar joelho: Movimento aplicado aos motores JE ou JD que faz com que o motor realize uma rotação de aproximadamente 45º no sentido inverso ao do movimento "dobrar joelho". Este movimento simula uma acção na qual uma bailarina estende a perna deixando de ter o joelho flectido.
- Levantar perna: Movimento aplicado aos motores CD ou CE que faz com que o motor realize uma rotação de aproximadamente 90º. Este movimento faz com que toda a perna se levante até ao nível da cintura. Este movimento simula um tipo de movimento conhecido, na dança cancan, como *kick*.
- Descer perna: Movimento aplicado aos motores CD ou CE que faz com que a perna desça até que o pé do robô fique próximo do chão. Este movimento simula o momento em que uma bailarina de cancan desce a perna para colocar o pé no chão.
- Levantar perna ao alto: Movimento aplicado aos motores CD ou CE. Este movimento assemelha-se ao movimento de "levantar perna". A diferença é que em vez de levantar a perna a aproximadamente 90º, levanta-a a aproximadamente 180º.
- Rotação de cintura: Movimento aplicado ao motor C que permite ao robô fazer pequenas rotações de cerca de 22,5º. Este movimento serve para simular acções nas quais uma bailarina de cancan se vira de lado para o público. Este movimento pode ser aplicado com um valor positivo ou negativo, isto faz com que a rotação seja realizada para a esquerda ou para a direita respectivamente.
- Centrar cintura: Movimento aplicado ao motor C que permite ao robô executar uma rotação para ficar virado para a posição inicial. Por norma estas rotações são de cerca de 22,5º mas poderiam variar caso o movimento de rotação de cintura realizasse uma rotação maior. Este movimento serve para simular acções nas quais uma bailarina de cancan roda para ficar de frente para o público.

A partir da lista de movimentos simples acima apresentados é possível construir diversos movimentos compostos. É exemplo disto, um movimento de "KickDobrado" que consiste na composição de um movimento de "levantar perna" com um movimento de "dobrar joelho".

Tendo em conta que o módulo de controlo de robô necessita de uma lista definida dos movimentos que pode realizar, resolvi fazer uma lista de movimentos possíveis que simulam acções coreográficas de uma bailarina de cancan. Podemos assim ver na tabela 4.1, uma lista dos movimentos actualmente reconhecidos pelo módulo de controlo de robô e qual é a sua composição em termos de movimentos simples desempenhados por cada motor.

Nome do movimento	Motor CE ou CD	Motor JE ou JD	Motor C
Kick	Levantar perna	-	-
DescePerna	Descer perna	-	-
KickDobrado	Levantar perna	Dobrar joelho	-
DesceDobrado	Descer perna	Esticar joelho	-
RodaEKick	Levantar perna	-	Rotação de cintura
RodaEKickDobrado	Levantar perna	Dobrar joelho	Rotação de cintura
RodaDKick	Levantar perna	-	-Rotação de cintura
RodaDKickDobrado	Levantar perna	Dobrar joelho	-Rotação de cintura
DesceRoda	Descer perna	-	Centrar cintura
DesceRodaKickDobrado	Descer perna	Esticar joelho	Centrar cintura
Highkick	Levantar perna ao alto	-	-
RodaEHighkick	Levantar perna ao alto	-	Rotação de cintura
RodaDHighkick	Levantar perna ao alto	-	-Rotação de cintura

Tabela 4.1: Lista de movimentos reconhecidos pelo módulo de controlo do robô e sua composição a partir dos movimentos simples.

Os movimentos definidos na tabela 4.1 podem ser aplicados a ambas as pernas, sendo apenas necessário que o módulo de controlo do robô decida se deve aplicar o movimento aos motores CD e JD ou aos motores CE e JE.

Com base nos movimentos definidos na tabela, foi criado um ficheiro de coreografia (figura 4.8). Esse ficheiro é mais tarde lido pelo módulo de controlo de robô para que este possa interpretar a informação recebida do ficheiro para transformar essa informação em acções para o robô. Como podemos ver no exemplo de ficheiro presente na figura 4.8, cada linha do ficheiro corresponde a um movimento composto seguido de um carácter, ‘E’ ou ‘D’, que serve para informar o módulo de controlo do robô sobre qual das pernas deve desempenhar a acção a que corresponde o movimento lido. Assim, quando é lido um ‘E’ o movimento será executado pela perna esquerda, e quando é lido um ‘D’ o movimento será executado pela perna direita.

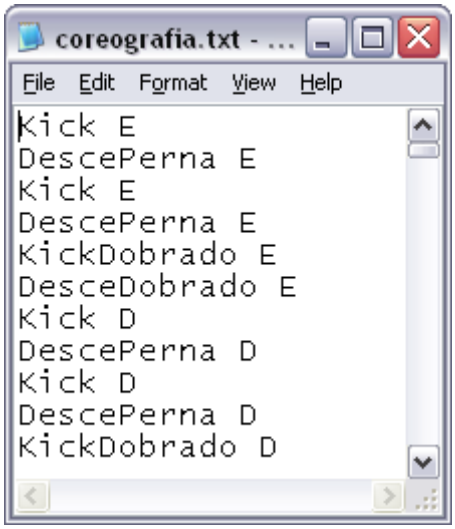


Figura 4.8: Exemplo de um pedaço do ficheiro de coreografia

4.5 Módulo de controlo do robô

O novo módulo de controlo do robô foi decomposto em três partes sendo que cada uma dessas partes pode ser executada em diferentes computadores. São essas três partes descritas nas subsecções abaixo com os nomes de Controlador geral, Controlador da perna esquerda e cintura, Controlador da perna direita.

A utilização destas três partes deve-se ao facto de ser necessário comunicar com os dois *bricks* referidos na construção do robô. Assim, cada um dos controladores associados às pernas tem a função de comunicar com um *brick*, estando o controlador geral responsável por coordenar o funcionamento dos outros dois controladores, enviando para eles mensagens com a informação que eles necessitam para fazer o robô actuar (figura 4.9).

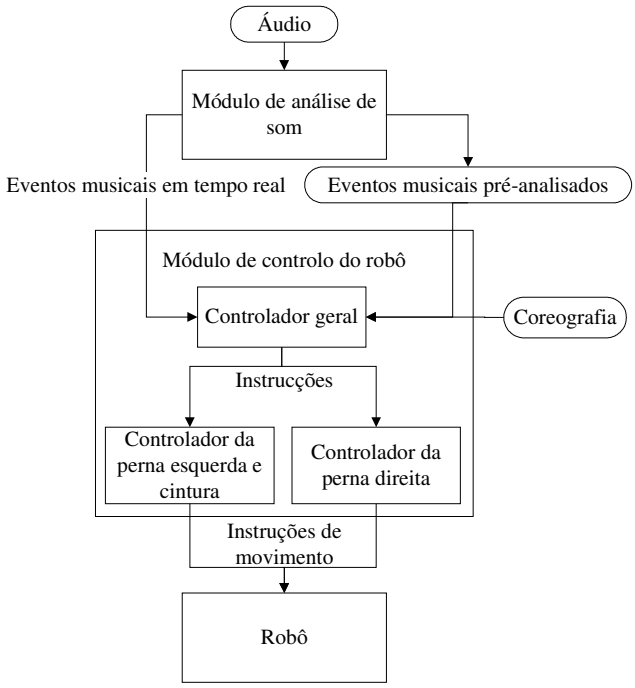


Figura 4.9: Arquitectura do sistema com foco para o módulo de controlo do robô.

4.5.1 Controlador geral

Este controlador é o responsável por filtrar os *onsets* e validar os *beats* executando sequencialmente a coreografia. Ao primeiro *beat* irá corresponder o primeiro movimento da coreografia e assim sequencialmente.

Para isso, ele coordena o funcionamento dos outros dois controladores presentes no módulo de controlo do robô enviando-lhe instruções para eles executarem. Para que isto seja possível, o controlador geral estabelece um canal de comunicação por *socket TCP* com cada um dos outros dois controladores, sendo ainda necessário estabelecer um terceiro canal de comunicação, também ele por *socket TCP*, com o sistema Marsyas, que se encontra no módulo de análise de som. O estabelecimento dos canais de comunicação é o que marca o início da execução em tempo real da dança. Para tal, o controlador começa por estabelecer um canal de comunicação com os controladores das pernas e só quando estes respondem informando que estão aptos é que é enviado um pedido de ligação ao sistema de análise de som, Marsyas. Se todas as ligações forem estabelecidas, o Marsyas inicia o seu processo de reprodução e análise da música, enviando para o controlador geral, através da ligação estabelecida, os *onsets* detectados. No momento em que o controlador geral recebe o aviso de que a música vai começar regista qual o tempo em que esse início ocorreu.

Sempre que é recebido um aviso enviado pelo Marsyas, informando que ocorreu um *onset*, o controlador geral verifica se este corresponde ao próximo *beat*. Para isso,

calcula-se o tempo em que o *onset* ocorre em relação ao tempo inicial e se for menor que o tempo do próximo *beat*, é ignorado por não ser considerado um *beat*. Sendo improvável que os tempos do *onset* e do *beat* coincidam, devido ao atraso provocado pela comunicação, considerei que qualquer *onset* que ocorra depois do tempo do próximo *beat* seja validado como *beat*.

Cada movimento da coreografia corresponde à acção de um ou mais motores que vai durar um certo tempo. Quando identifica um *beat*, o controlador geral lê a próxima linha do ficheiro de coreografia de onde obtém qual será o movimento a realizar e calcula ainda o tempo que tem para o realizar. O movimento é interpretado e convertido num comando que diz quais os IDs dos motores que devem actuar e a localização para onde estes devem rodar, em localização *rotation count* (diferença em unidades de rotação face à posição inicial do motor), sendo que essa conversão é feita segundo a tabela 4.2. *Rotation count* é a unidade que é utilizada pelo iCommand para definir a posição para onde o motor se vai mover ou a posição este se encontra. É impossível fazer uma conversão precisa de *rotation count* para graus, no entanto, visto a necessidade de explicar melhor o movimento do motor, fica definido a taxa de conversão 1 *rotation count* ≈ 1,125º graus.

Nome do movimento	Motor da coxa (CD ou CE) ID: B	Motor do joelho (JD ou JE) ID: A	Motor da cintura (C) ID: C
Kick	-80	0	-
DescePerna	-	0	-
KickDobrado	-80	40	-
DesceDobrado	0	0	-
RodaEKick	-80	-	20
RodaEKickDobrado	-80	40	20
RodaDKick	-80	-	-20
RodaDKickDobrado	-80	40	-20
DesceRodaKick	0	-	0
DesceRodaKickDobrado	0	0	0
Highkick	-150	-	-
RodaEHighkick	-150	-	20
RodaDHighkick	-150	-	-20
DesceRodaHighkick	0	-	0

Tabela 4.2: Tabela que descreve a conversão dos movimentos lidos do ficheiro de coreografia em valores de *rotation count*.

Para uma melhor percepção dos valores apresentados na tabela 4.2, apresenta-se uma explicação através da figura 4.10 que se refere aos motores com ID ‘A’ e ‘B’. No caso do motor com ID ‘C’, o valor 20 corresponde a uma rotação para a esquerda, o valor -20 corresponde a uma rotação para a direita e o valor 0 corresponde a uma rotação para o centro.

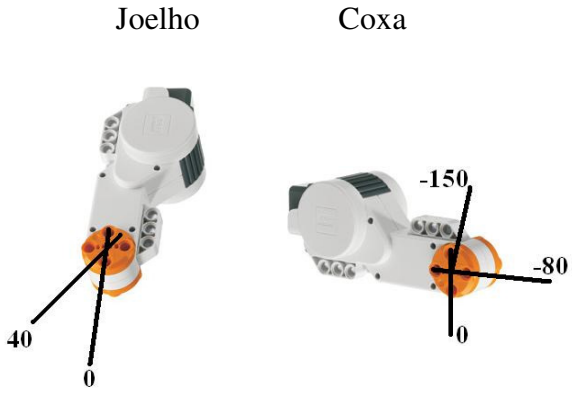


Figura 4.10: Ilustração dos valores de *rotation count* para os joelhos e coxas.

A partir da linha do ficheiro de coreografia, o controlador geral recebe ainda informação de qual dos controladores das pernas será responsável pela execução daquele movimento. Essa informação é dada pela letra que sucede ao nome do movimento. O controlador é o da perna esquerda e cintura, caso a letra seja um ‘E’ e será o da perna direita, caso a letra seja um ‘D’. Todos os movimentos que envolvam rotação de cintura são decompostos acção de cintura e acção da perna. Se a perna que deve actuar é a esquerda, toda a instrução de movimento é enviada para o controlador da perna esquerda e cintura. Se a perna que deve actuar é a direita, a instrução da perna é enviada para o controlador da perna direita e a instrução da cintura é enviada para o controlador da perna esquerda e cintura.

Após fazer a interpretação e conversão da instrução, é calculado qual o período de tempo para realizar o movimento. Este período de tempo é calculado fazendo a diferença entre o tempo do próximo *beat* e o tempo do *beat* corrente, utilizando o ficheiro com a informação do tempo dos *beats*. Quando ocorre o ultimo *beat* da música, não existe um intervalo até ao próximo. Nesse caso o período de tempo para realizar o movimento é igual ao período de tempo utilizado para o movimento anterior.

O período de tempo calculado é acrescentado no comando onde já se encontram o ID dos motores a mover e a localização para onde estes se vão mover. Desta forma, o controlador geral tem já acesso a toda a informação que necessita de comunicar aos controladores das pernas. Como tal, é enviado para o controlador que será responsável pela acção, o comando com os dados referentes a que motores devem ser movidos, para onde devem ser movidos e quanto tempo têm para realizar o movimento.

Para melhor compreensão apresento um exemplo de comando a ser enviado com instruções para realizar um "KickDobrado E" que consiste no movimento do motor da coxa (B) para o *rotation count* -80 e o movimento do motor do joelho (A) para o *rotation count* 40, ambos em 490 milissegundos.

Exemplo de comando a enviar: "490 A 40 B -80"

O controlador geral continua em execução até que o sistema de análise de som, Marsyas, informe que a música terminou. Durante o tempo de execução, o controlador geral responde apenas aos eventos enviados pelo Marsyas, não aguardando em nenhuma altura que o movimento a desempenhar pelos controladores das pernas terminem.

4.5.2 Controlador da perna esquerda e cintura

O controlador da perna esquerda e da cintura é responsável por movimentar a perna esquerda e a cintura. Para isso ele precisa de ter uma ligação a um dos *bricks* do robô e precisa ainda da informação relativa ao movimento a executar.

Para iniciar o funcionamento, este controlador aguarda que lhe seja feito um pedido de ligação por *socket TCP*, vindo do controlador geral. Após receber este pedido de ligação, o controlador cria um canal de ligação Bluetooth com o *brick* que se encontra ligado aos motores da perna esquerda e da cintura. Este controlador só fica operacional quando o canal de Bluetooth for estabelecido, altura em que é enviada para o controlador geral, por *socket TCP*, uma mensagem que avisa que a perna esquerda e a cintura estão prontas para receber acções.

Estabelecidas as ligações, este controlador fica apto para receber instruções vindas do controlador geral. Estas instruções vêm no formato *string* e contêm os IDs dos motores a mover, a posição absoluta para onde esses motores se devem mover e o intervalo de tempo que têm para executar determinada acção.

Para enviar uma acção para os motores é necessário indicar quais os motores que vão ser utilizados, qual posição que os motores vão assumir e qual a velocidade que cada motor deve utilizar. Visto que a informação relativa a que motores serão utilizados e que posição é que estes devem assumir, é enviada pelo controlador geral, apenas falta obter a velocidade que cada motor deve utilizar para garantir que cumpre a sua acção antes que seja disparado um novo evento que dê origem a uma nova acção. Para calcular estas velocidades é utilizado o intervalo de tempo recebido do controlador geral e a distância que os motores têm de percorrer para completar a acção. Assim, para cada motor é aplicada a fórmula da velocidade:

$$v = \frac{d}{\Delta t}$$

Nesta fórmula que permite encontrar a velocidade necessária para que cada motor complete o movimento no tempo que tem disponível, o 'v' corresponde à velocidade a assumir pelo motor, o 'd' corresponde à distância que o motor terá de percorrer e o 'Δt' corresponde ao intervalo de tempo que deve durar o movimento. Infelizmente, devido aos atrasos nas comunicações de todo o processo que vai desde o disparar do evento sonoro até ao envio para o motor do robô, esta velocidade deixou de ser a correcta. Para resolver este problema, fiz vários testes de velocidade até que fosse visível que o tempo perdido na comunicação não prejudicava o desempenho. Cheguei com estes testes a um valor de 200 graus por segundo que foi somado ao resultado da fórmula. Estes 200 graus por segundo passaram a ser a compensação pelas perdas de tempo de comunicação. Assim a fórmula que será aplicada é:

$$v = \frac{d}{\Delta t} + 200$$

Calculadas as velocidades restava enviar para os motores as acções a executar, no entanto, deparei-me aqui com outro problema. Devido à falta de precisão dos motores da Lego, os motores por vezes ultrapassavam, ou não atingiam, o destino criando situações de bloqueio. Para resolver isto foi criado um mecanismo que força o movimento. Este mecanismo é dividido em duas partes, avaliação do estado do movimento e insistência no desempenho.

A avaliação do movimento consiste numa análise da posição dos motores no decorrer da acção. Durante a acção, o controlador vai obtendo dados relativos à posição em que o motor se encontra atribuindo um estado de movimento incompleto caso se verifique que o motor ficou parado sem chegar ao destino. Caso se verifique que o movimento se encontra incompleto, é necessário insistir no desempenho. O mecanismo de insistência envia novamente a acção ao motor que se encontra com o estado de movimento incompleto fazendo com que o motor comece a actuar novamente.

Este controlador fica em execução até que seja informado pelo controlador geral que a música terminou. Durante o tempo de execução a ligação Bluetooth é mantida para não ser perdido tempo a estabelecer novamente a ligação.

4.5.3 Controlador da perna direita

O controlador da perna direita tem o mesmo sistema de funcionamento que o controlador da perna esquerda e cintura. As diferenças são apenas relativas ao *brick* ao qual este controlador estabelece a ligação e ao número de motores que este controlador

controla. Assim, este controlador estabelece ligação Bluetooth ao *brick* que se encontra ligado aos motores da perna direita. Este *brick* não tem um terceiro motor como acontece com o *brick* associado ao controlador da perna esquerda e cintura.

Ao colocar este controlador em execução, deparei-me com um problema com a API utilizada para controlo do robô, o iCommand. Esta API tem uma limitação que faz com que não seja possível comunicar com dois *bricks* diferentes em simultâneo. Foi testada a ligação à vez a um *brick* e depois ao outro, mas devido aos elevados tempos que demorava a estabelecer a ligação, o ritmo da música era completamente perdido, comprometendo assim o objectivo do projecto. Para solucionar este problema passei a utilizar duas ligações Bluetooth colocadas em computadores distintos. Como tal, o controlador da perna direita e o controlador da perna esquerda e cintura, passaram a funcionar em computadores diferentes (figura 4.11).

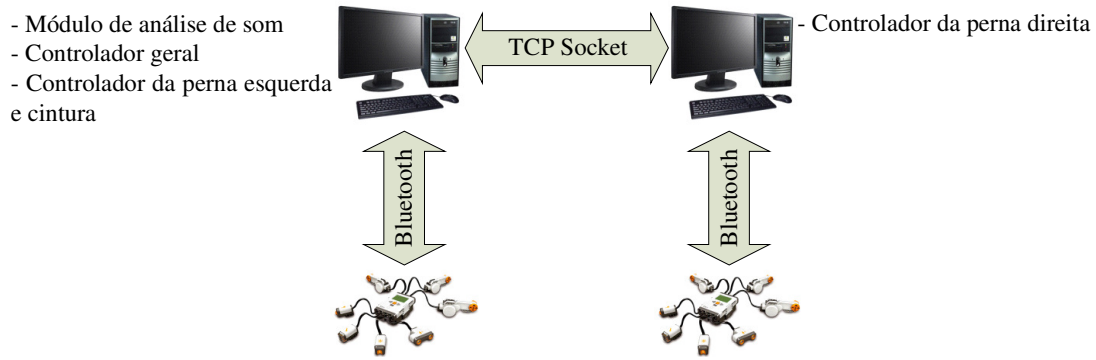


Figura 4.11: Representação do funcionamento do sistema com os dois computadores no qual se mostra o controlador da perna direita a funcionar no segundo computador que se liga por Bluetooth ao segundo *brick*.

4.6 Discussão de resultados

Apesar das diversas dificuldades encontradas no desenvolvimento deste modelo final, consideram-se os resultados bastante interessantes como se pode verificar no vídeo [24]. Notou-se contudo um certo atraso em algumas das acções do robô que se ficam a dever à demora na comunicação Bluetooth. Tal como referido anteriormente, utilizei um incremento em termos de velocidade para compensar estas perdas, no entanto o tempo de comunicação não é sempre igual. Outra dificuldade que observei no resultado final foi que por vezes, devido ao elevado número de pedidos que são realizados por Bluetooth para resolver os problemas de imprecisão dos motores, alguns passos coreográficos são ignorados.

Foi ainda possível notar que alguns dos movimentos compostos que utilizam mais de um motor em simultâneo, por vezes não operam da forma esperada. Este problema também adveio da grande quantidade de pedidos feitos por Bluetooth, pois no caso de

estarem dois motores a funcionar em simultâneo a quantidade de pedidos que é feita, para responder aos problemas de imprecisão, é mais elevada.

Em suma, posso dizer que grande parte dos problemas veio da necessidade de manter a comunicação Bluetooth entre o computador e o robô, esta necessidade podia ser resolvida tentando colocar o processamento a ser realizado no próprio robô. Infelizmente para que isso fosse possível era absolutamente necessário que o microfone do Kit Lego NXT tivesse melhor qualidade e que o processador do *brick* tivesse capacidade de processamento para correr um sistema de análise de som.

Os resultados finais deste trabalho poderão ser melhor observados através de um conjunto de vídeos [24] sendo ainda possível visualizar o aspecto físico final do robô nas figuras 4.12 e 4.13.

Figura 4.12: Pernas Cancan vestido (de frente)

Figura 4.13: Pernas Cancan vestido (de lado)

Capítulo 5

Conclusões e trabalho futuro

A dança robótica é uma área com bastante interesse pois pode ser usada quer no entretenimento quer como ferramenta de auxílio a coreógrafos. Infelizmente, produzir uma ferramenta com qualidade suficiente para servir estes interesses exige um financiamento elevado. O resultado final ficou um pouco abaixo do que era o objectivo inicial pois foi encontrada uma serie de obstáculos no percurso.

Com a experiencia que obtive durante este trabalho posso afirmar que alguns dos obstáculos poderiam ter sido evitados se houvesse melhor documentação do software e hardware utilizados. Um exemplo de obstáculo que consumiu grande parte do tempo foi a exploração do código do Marsyas. A documentação estava em péssimas condições e o código da aplicação encontrava-se num estado tão mau que nem as pessoas que o desenvolveram conseguiram explicar que funcionalidades estavam operacionais e o que faziam ao certo. Se a duração do trabalho que levou a este documento fosse maior teria sido uma boa solução pensar no desenvolvimento de um sistema de análise de som próprio para a aplicação.

Outro dos grandes obstáculos que provocou grandes dificuldades foi a precisão e capacidade dos Kits Lego NXT. Como já referi, são kits pensados para serem de utilização fácil para crianças de 10 anos, e apesar de hoje em dia existirem APIs que permitem usufruir de mais capacidades do que as pensadas inicialmente, os kits continuam a ter grandes limitações. Isto tornou a coordenação e sincronização dos movimentos do robô muito difícil, pois existiam sempre pequenas perdas de precisão no movimento que acumuladas durante uma música de mais de dois minutos se tornavam em perdas bastante consideráveis. Mais uma vez, com a experiencia poderia evitar estes problemas, uma solução teria sido a criação do robô Pernas Cancan virtual para que fosse possível começar por uma simulação. Obviamente haveria sempre o eterno problema da robótica que é a diferença entre o mundo simulado onde tudo corre sem problemas e o mundo real onde até o atrito do ar pode influenciar o comportamento de

um robô, contudo esta medida podia oferecer um modelo de comparação com o modelo no mundo real.

Trabalho futuro

Numa perspectiva de melhorar o trabalho do Cancan Robótico, foram pensadas algumas modificações que podem ser realizadas em trabalho futuro.

São propostas de trabalho futuro:

- Inclusão de um sistema de análise de som que permita realizar previsão de *beats*. Espera-se que a inclusão da previsão de *beats* dê ao robô a possibilidade de dançar em tempo real sem análise prévia da música.
- Desenvolvimento de mais movimentos básicos e compostos. Esta modificação visa alargar o leque de movimentos que o robô pode desempenhar, enriquecendo a coreografia.
- Inclusão de técnicas de aprendizagem aplicadas à coreografia para que seja possível gerar coreografias através de algoritmos genéticos.
- Criação de um método de definição de coreografias automático. Esta modificação visa transformar a coreografia para que o robô possa ter um comportamento adaptativo no qual não repita as mesmas coreografias. A ideia deste método será que as coreografias possam ser criadas em tempo real utilizando um processo probabilístico.
- Criação de um equipa de robôs que possa desempenhar a coreografia em *chorus line* tal como acontece na dança cancan quando é interpretada por um conjunto de bailarinas. Esta modificação pressupõe que em certas partes da música todos os robôs dançam em sincronia mas que em certas partes da música cada um dos robôs se destaca dos outros realizando um solo.

Referências

[1] Oliveira, J. "Towards an Interactive Framework for Robot Dancing Applications" Dissertação realizada no âmbito do Mestrado Integrado em Engenharia Electrotécnica e de Computadores, Faculdade de Engenharia da Universidade do Porto, Portugal, 2008.

[2] Marsyas: http://marsyas.info/ Consultado pela última vez a 6 de Abril de 2011.

[3] Tzanetakis G., Cook P. "Marsyas: A framework for audio analysis." Buckland, M., Programming Game AI by Example, *Wordware Publishing*, Organised Sound 4(3) pp. 169-175 2005.

[4] Dancing Cockatoo Snowball: http://www.youtube.com/watch?v=N7IZmRnAo6s Consultado pela última vez a 6 de Abril de 2011.

[5] Lourenço, M., Urbano, P., Teixeira, C., "The First Steps of Robotic Cancan", *Proceedings of the 5º CISTI, pp. 594-597*, 2010.

[6] Riley, M., Ude, A. and Atkeson, C.G., "Methods for Motion Generation and Interaction with a Humanoid Robot: Case Studies of Dancing and Catching" *Proceedings of Workshop on Interactive Robotics and Entertainment*, Robotics Inst.,Carnegie Mellon Univ., Pittsburgh, pp. 35–42, 2000.

[7] Nakazawa, A., Nakaoka, S., Ikeuchi, K. and Yokoi, K., "Imitating Human Dance Motions through Motion Structure Analysis" *Proceedings of International Conference on Intelligent Robots and Systems*. IROS, pp. 2539–2544, 2002.

[8] Kosuge K., Hayashi T., Hirata Y. and Tobiyama R. "Dance partner robot MsDanceR". *Proceedings of the 2003 IEEE/RSJ international conference on intelligent robots and systems,* pp 3459–346, 2003.

[9] 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)*, Kurashiki, Japan, pp. 419-424, 2004.

[10] 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)*, Osaka, Japan, pp. 142-147, 2005.

[11] Shinozaki, K., Iwatani, A. and Nakatsu,R. "Study of Dance Entertainment Using Robots" *Pro literature Verlag*, ch.27, pp.535–544, 2007.

[12] Shinozaki, K., Iwatani, A. and Nakatsu, R. "Concept and Construction of a Robot Dance System". *The International Journal of Virtual Reality 6*, pp. 29-34, 2007.

[13] 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" *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-2007)*, 1743-1750, IEEE, RSJ, San Diego, 2007.

[14] Aucouturier, J.-J. and 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.

[15] Kozima, H., Nakagawa, C. "A robot in a playroom with preschool children: Longitudinal field practice", *Robot and Human interactive Communication, 2007*. RO-MAN 2007. The 16th IEEE International Symposium on 26-29, pp. 1058–1059, 2007.

[16] Oliveira, J., Gouyon, F., and Reis L. (2008) "Robot Dance based on Online Automatic Rhythmic Perception", *3rd International Workshop on Intelligent Robotics*, 2008.

[17] Oliveira, J., Gouyon, F. and Reis, L. P. (2008) "Towards on Interactive Framework for Robot Dancing Applications", *Proceeding of Artech 2008 - 4th International Conference on Digital Arts,* Universidade Católica, Porto, Portugal, 2008.

[18] Grunberg D., Ellerberg R., Dr Kim Y. and Dr Oh P., "Creating an Autonomous Dancing Robot", *Proceedings of the 2009 International Conference on Hybrid Information Technology,* (ICHIT), pp.221–227, 2009.

[19] Bello, J. P., Daudet, L., Abdallah, S., Duxbury, C., Davies, M. and Sandler, M. "A Tutorial On Onset Detection in Musical Signals." *IEEE Transactions on Speech and Audio Processing*, pp. 1035–1047, 2005.

[20] Allen, P. and Dannenberg, R. "Tracking musical beats in real time". *Proceedings of the International Computer Music Conference*, San Francisco CA. International Computer Music Association, pp 140–143, 1990.

[21] Goto, M. and Muraoka, Y. “A real-time beat tracking system for audio signals” *Proceedings of the International Computer Music Conference*, San Francisco CA. International Computer Music Association, pp 171–174, 1995

[22] LabMAg Cancan Robótico: http://bookmark.labmag.di.fc.ul.pt/?page_id=680 Secção de videos, Catatua Dançarina. Consultado pela última vez a 6 de Abril de 2011.

[23] Dixon, S. “Evaluation of audio beat tracking system beat-root”. *Journal of New Music Research*, 36(1), 2007.

[24] LabMAg Cancan Robótico: http://bookmark.labmag.di.fc.ul.pt/?page_id=680 Secção de videos, Pernas Cancan. Consultado pela última vez a 6 de Abril de 2011.