Technical University of Crete, Greece
Department of Electronic and Computer Engineering

Interleaving of Motion Skills for Humanoid Robots

Astero - Dimitra Tzanetatou

Thesis Committee
Assistant Professor Michail G. Lagoudakis (ECE)
Assistant Professor Aikaterini Mania (ECE)
Assistant Professor Georgios Chalkiadakis (ECE)

Chania, March 2012

Πολυτεχνειο Κρητης

Τμημα Ηλεκτρονικων Μηχανικων και Μηχανικων Υπολογιστων

Διαπλοκή Κινητικών Δεξιοτήτων για Ανθρωποειδή Ρομπότ

Αστέρω - Δήμητρα Τζανετάτου

Εξεταστική Επιτροπή

Επίκουρος Καθηγητής Μιχαήλ Γ.Λαγουδάκης (ΗΜΜΥ)

Επίκουρη Καθηγήτρια Αικατερίνη Μανιά (ΗΜΜΥ)

Επίκουρος Καθηγητής Γεώργιος Χαλκιαδάκης (ΗΜΜΥ)

Χανιά, Μάρτιος 2012

Acknowledgements

First of all, I would like to thank Prof. Michail G. Lagoudakis for his guidance through every step of the present thesis and helpful hints for completing this work. I have earned much knowledge and experience from our collaboration.

Of course, I would like to thank my parents, who were always besides me and supported my choices.

I would also like to thank all the teammates, older and younger, of team Kouretes. Lefteris, Manos, Iris, Vagelis, Nikos Sp., Maria, Nikos K. and Nikos P. your help was very important to me and, without your valuable advices, I could not have completed this work. Furthermore, I would like to thank two previous members of the team, Alex and George, whose contribution to this team is very important.

Separately, I would like to thank the two people closest to me for the last six years. My best friend and also teammate Angelica, with whom we have been together from the first day at TUC and I hope our friendship will never perish. Of course, I would like to thank my dear Pavlos, who supports me and gives me the energy that I need to move on.

Finally, I would like to thank my good friends at the Technical University of Crete for all the wonderful experiences we shared all these years.

Abstract

RoboCup is an international robotic soccer competition aiming at advancing research in autonomous robotics and artificial intelligence. Participating teams in the RoboCup Standard Platform League (SPL) focus on algorithm design and software development on the same robot platform, the Aldebaran Nao humanoid robot. Walking speed and special action completion times are critical factors generally in humanoid robots and particularly in SPL games. It has been observed that significant time is lost in approaching the ball, stopping to assume the right pose, and then executing an appropriate kick action. This thesis explores the idea of interleaving walk and kicks, aiming at kicking the ball directly without slowing down while approaching it, in order to decrease the total response time. The (forward, side, and back) kicks of our RoboCup team Kouretes have been designed as timed sequences of robot poses. Therefore, the leg poses of our kick actions were analyzed against the recorded walk leg poses to identify close matchings and some of them were bookmarked as potential entry points into the kick sequences. This analysis involved the 8 most significant out of the 11 leg joints, as well as their gradients (rate of change), and the Euclidean (L2) norm over these 16-dimensional vectors as a similarity metric. As the robot walks up to the ball and gets to kick distance, these features are monitored in real-time. As soon as the current walk pose comes close to one of the bookmarked poses of the desired kick (norm below a threshold), the robot switches immediately from walking to the corresponding entry point and executes the kick sequence from that point. This way, the kick is executed before the robot stops walking. Our implementation is fully parameterizable through XML files to accommodate any special actions, walk gaits, and similarity metrics. The proposed motion interleaving technique has significantly reduced the total time of approaching and kicking the ball, yielding significant advantage to our robots.

Περίληψη

Το RoboCup είναι ένας διεθνής διαγωνισμός ρομποτικού ποδοσφαίρου που στοχεύει στην προώθηση της έρευνας στις περιοχές της αυτόνομης ρομποτικής και της τεχνητής νοημοσύνης. Οι συμμετέχουσες ομάδες στο πρωτάθλημα Standard Platform League (SPL) του RoboCup εστιάζουν στη σχεδίαση αλγορίθμων και στην ανάπτυξη λογισμικού για την ίδια ρομποτική πλατφόρμα, το ανθρωποειδές ρομπότ Aldebaran Nao. Η ταχύτητα του βαδίσματος και οι χρόνοι εκτέλεσης δεξιοτεχνικών κινήσεων είναι κρίσιμοι παράγοντες γενικά στα ανθρωποειδή ρομπότ και ειδικότερα στα παιχνίδια του SPL. Έχει παρατηρηθεί ότι σημαντικός χρόνος αναλώνεται στο να προσεγγίσει το ρομπότ τη μπάλλα, να σταματήσει για να λάβει τη σωστή στάση, και στη συνέχεια να εκτελέσει το κατάλληλο λάκτισμα. Η εργασία αυτή διερευνά την ιδέα της διαπλοκής βαδίσματος και λακτίσματος, στοχεύοντας στο λάκτισμα της μπάλλας απευθείας κατά την προσέγγιση χωρίς καθυστέρηση, προκειμένου να μειωθεί ο συνολικός χρόνος απόκρισης. Τα (εμπρός, πλάγια, και πίσω) λακτίσματα της ομάδας RoboCup Κουρήτες έχουν σχεδιαστεί ως χρονισμένες ακολουθίες στάσεων του ρομπότ. Ως εκ τούτου, οι στάσεις των ποδιών κατά το λάκτισμα αναλύθηκαν σε αντιπαράθεση με τις καταγεγραμμένες στάσεις των ποδιών κατά το βάδισμα για να προσδιορισθούν κοντινά ταιριάσματα και κάποιες στάσεις προκαθορίστηκαν ως πιθανά σημεία εισόδου στην ακολουθία του λακτίσματος. Η ανάλυση αυτή συμπεριέλαβε τις 8 πιο σημαντικές από τις 11 αρθρώσεις των ποδιών, καθώς και τις παραγώγους τους (ρυθμός μεταβολής), και την Ευκλείδεια (L2) νόρμα πάνω σ΄ αυτά τα 16-διάστατα διανύσματα ως μετρική ομοιότητας. Καθώς το ρομπότ βαδίζει προς τη μπάλα και φτάνει σε απόσταση βολής, τα παραπάνω χαρακτηριστικά παρακολουθούνται σε πραγματικό χρόνο. Μόλις η τρέχουσα στάση των ποδιών έρθει κοντά σε κάποια από τις προκαθορισμένες στάσεις του επιθυμητού λακτίσματος (τιμή νόρμας κάτω από κάποιο όριο), το ρομπότ μεταβαίνει αμέσως από το βάδισμα στο αντίστοιχο σημείο εισόδου και εκτελεί την ακολουθία του λακτίσματος από το σημείο αυτό. Μ΄ αυτόν τον τρόπο, το λάκτισμα εκτελείται πριν το ρομπότ σταματήσει να κινείται. Η υλοποίησή μας είναι πλήρως παραμετροποιήσιμη μέσω αρχείων XML για να μπορεί να εξυπηρετήσει οποιεσδήποτε κινητικές δεξιότητες, τεχνικές βαδίσματος, και μετρικές ομοιότητας. Η προτεινόμενη τεχνική διαπλοκής κινητικών δεξιοτήτων έχει μειώσει σημαντικά το συνολικό χρόνο προσέγγισης και λακτίσματος της μπάλλας, προσφέροντας σημαντικό πλεονέκτημα στα ρομπότ μας.

Contents

List of Figures

Chapter 1

Introduction

In RoboCup Standard Platform League games, all teams use the same robot, in other words the same hardware. So, the teams need to design better and faster kicks to be able to shoot the ball first. After all these years, that the RoboCup SPL has taken place, most teams have already developed fast walk and kicks, so there is a need of a new way of approaching and shooting the ball that will be even faster. This thesis describes a new approach and shoot method, in order to decrease the total response time, concerning the interleaving of walk and kick. This method's purpose is the direct kicking of the ball, without slowing down, while the robot is approaching the ball. For those reasons an algorithm is developed to compare the values of the joints while the robot is walking with the values of the joints while the robot is kicking. After a few different implementations we succeeded our goal and the robot acquired the ability to walk and kick directly. Also the major goal of this thesis was succeeded by the reduction of the total time of the approach and ball shooting.

1.1 Thesis Contribution

As it was mentioned before, there is a need of a new way of approaching and shooting the ball that will be even faster. This thesis succeeded to reduce the total time of the robot's approach and kick the ball. The contribution of this thesis to our SPL team, Kouretes, is the reduction of the time because it gives to the robot more chances to acquire the ball before the opponent robot and score.

1.2 Thesis Outline

Chapter 2 provides some necessary background information on the RoboCup competition focusing on the Standard Platform League and the robot used in SPL, namely the Aldebaran Nao humanoid robot. Additionally, it briefly describes Monas, the software architecture of team Kouretes in which the work presented in this thesis was implemented. In Chapter 3 we state the problem we study in this thesis and we discuss related work. Continuing to Chapter 4 the core ideas of our approach to walk and kick interleaving are presented. Chapter 5 provides implementation details pertaining to the Monas architecture and the Nao robot. In Chapter 6 we provide detailed examples to demonstrate the effectiveness of our approach. Finally, in Chapter 7 we discuss similar approaches and we propose directions for extending and improving our work in the future.

Chapter 2

Background

2.1 The RoboCup Competition

The RoboCup competition, in its short history, has grown to a well-established annual event bringing together the best robotics researchers from all over the world. The initial conception by Hiroaki Kitano [1] in 1993 led to the formation of the RoboCup Federation with a bold vision: "By the year 2050, to develop a team of fully autonomous humanoid robots that can win against the human world soccer champions". The uniqueness of RoboCup stems from the real-world challenge it poses, whereby the core problems of robotics (perception, cognition, action, coordination) must be addressed simultaneously under real-time constraints. The proposed solutions are tested on a common benchmark environment through soccer games in various leagues, with the goal of promoting the best approaches and ultimately advancing the state-of-the-art in the area. The RoboCup Soccer event is the flagship competition with the most fans. In this domain, researchers exploit their technical knowledge in order to prepare the best robotic soccer team among all participants. The name RoboCup is a contraction of the competition's full name, "Robot Soccer World Cup", but there are many other divisions of the competition, such as search-and-rescue missions (RoboRescue), home- keeping tasks (RoboCup@Home), robotic performances (RoboDance), and simplified soccer leagues for K-12 students (RoboCup Junior). Broadening the research areas where RoboCup focuses was a very interesting and clever addition, which enables more scientists and researchers to combine their expertise in order to solve

Figure 2.1: Standard Platform League at RoboCup 2011

real-world problems. A lot of progress has been made so far in many disciplines of robotics and RoboCup has been established as one of the most important events around the world.

The main focus of the RoboCup Soccer activities is competitive soccer. Here robots are built and programmed in order to play autonomously or semi autonomously, soccer. This competition includes five leagues.

Standard Platform League

In the Standard Platform League (SPL), shown in Figure 2.1, all teams compete with identical robots, the Aldebaran Nao humanoid robots [2]. Therefore, the teams concentrate on software development only, while still using state-of-the-art robots. The robots operate fully autonomously, i.e. there is no external control, neither by humans nor by computers. No additional hardware is permitted including off-board sensing or processing systems. The robots can communicate through the wireless network. The only external information communicated to the robots comes from a game controller which declares the state of the game and keeps the time and the score.

SPL games are conducted on a $4m \times 6m$ soccer field built on a green carpet marked with white lines using an orange hockey ball. There are two goals, one in yellow color and another in sky blue color. The players are distinguished by

colored belts, blue for the team defending the sky blue goal and red for the team defending the yellow goal.

Each SPL game consists of two 10-minute halves and teams switch colors and sides at half-time. Each team consists of four robots, one goal keeper and three field players. There are strict rules applied by referees during the game. For example, robots pushing other robots, defenders entering their own penalty area, and inactive robots are penalized and removed from the game for 30 seconds. These rules are modified and change every year to increase the level of difficulty as the league progresses.

Small Size League

In the Small Size League (Figure 2.2) a robot soccer game takes place between two teams of five robots each. The robot must fit within an 180mm diameter circle and must be no higher than 15cm unless they use on-board vision. The robots play soccer on a green carpeted field that is 4.9m long by 3.4m wide with an orange golf ball.

Figure 2.2: Small Size League at RoboCup 2010

Robots come in two types, those with local (on-board) vision and those with global vision. Global vision robots, the most common variety, use an overhead camera, which is attached to a camera bar located 4m above the playing surface. Local vision robots have their camera on-board. The vision information is either

processed on-board the robot or is transmitted to an off-field PC for processing. An off-field PC is used to communicate referee commands and, in the case of overhead vision, position information to the robots. Typically the off-field PC also performs most, if not all, of the processing required for coordination and control of the robots. Communication is wireless and typically uses dedicated commercial FM transmitter/receiver units.

Middle Size League

In the Middle Size league (Figure 2.3), two teams of up to 6 robots play soccer on an $18m \times 12m$ indoor field. The robots' bodies are heavy enough, having powerful motors, heavy batteries, omni-directional camera, and a full laptop computer running on every robot. Through wireless communication they can establish inter-team cooperation and receive all referee commands. Once again, no external intervention by humans is allowed, except to insert/remove robots in/from the field.

Figure 2.3: Middle Size League at RoboCup 2009

Humanoid League

In the Humanoid League (Figure 2.4), teams build both the hardware and the software of the competing humanoid robots. The robots are divided into three size classes: Kid-Size (30-60 cm height), Teen-Size (100-120 cm) and Adult-Size (130 cm and taller). Dynamic walking, running, and kicking the ball while maintaining

balance, visual perception of the ball, other players, and the field, self-localization, and team play are among the many research issues investigated in the Humanoid League. In the Kid-Size class, teams consist of three robots, in the Teen-Size class, teams consist of two robots, and in the Adult-Size class, there is only one robot per team.

Figure 2.4: Humanoid League (Teen-Size class) at RoboCup 2010

Simulation League

Figure 2.5: Simulation League 2D (left) and 3D (right)

The RoboCup Simulation League (Figure 2.5) focuses mostly on team strategy as there are no physical robots and therefore no necessity to maintain any robot hardware. In the 2D Simulation League, two teams of eleven autonomous

software agents play soccer in a two-dimensional virtual soccer stadium. The 3D Simulation League increases the realism of the simulated environment by adding an extra dimension and more complex physics. Teams in the 3D Simulation League consist of nine humanoid simulated robots and the focus is placed on the design and implementation of multi agent, higher-level behaviors, based on solid low-level behaviors for humanoid robots.

2.2 Kouretes Team

Kouretes (Figure 2.6) is the first RoboCup team founded in Greece at the Technical University of Crete in February 2006.

Figure 2.6: Kouretes at RoboCup 2011 in Istanbul

The first participation of the team was in the Technical Challenges of Robocup 2006 in Bremen, Germany, when still the Four-Legged Sony AIBO robots were used. Next year the team participated also in the Four-Legged league of the RoboCup German Open 2007 competition in Hannover, Germany and ranked in the 7th/8th place. Months later the team's participation in the MSRS Simulation Challenge at RoboCup 2007 in Atlanta led to the placement of the team at the 2nd place worldwide. In RoboCup 2008 in Suzhou, China, Kouretes team participated and won two trophies: 1st place in the SPL-MSRS league and 3rd place in the

SPL-Nao league. In 2009 the team participated both at the RoboCup German Open 2009 competition in Hannover and in RoboCup 2009 in Graz, Austria, in the SPL. In 2010 the team participated in the 1st ever RoboCup Mediterranean Open competition in Rome and in RoboCup 2010 in Singapore. In 2011, the team participated in the RoboCup German Open 2011 competition in Megdeburg, in Iran Open 2011 in Tehran and in RoboCup 2011 in Istanbul. Team Kouretes joined Team Noxious from UK to form a joint team for RoboCup 2011. The joint team won the 2nd place in the SPL Open Challenge competition with 139 points, only 3 points behind the top team.

2.3 The Aldebaran Nao Robot

Nao is an autonomous, programmable, medium-sized humanoid robot, developed by Aldebaran Robotics, a French company headquartered in Paris. Project Nao was launched in 2004 [3]. On August 15, 2007, Nao officially replaced Sony's quadruped robot Aibo in the RoboCup Standard Platform League. The first version V1 of the robot was delivered in the first quarter of 2008 exclusively to RoboCup teams. Some months later version V2 replaced the too fragile first version and was used in RoboCup 2008. Version V3 was released in the first quarter of 2009 and the company replaced all previous versions. The current version V3.3, released in the beginning of 2011, overcomes most problems of the previous versions. Aldebaran Robotics offers a special edition of the Nao robot for teams competing in the RoboCup SPL.

2.3.1 Nao Hardware

Nao (version V3.3) is a 58 cm and 5 Kg humanoid robot. The Nao robot carries a fully capable computer on board with an x86 AMD Geode processor at 500 MHz, 256 MB SDRAM and 2 GB flash memory running an Embedded Linux distribution. It is powered by a 6-cell Lithium-Ion battery which provides about 30 minutes of continuous operation and communicates with remote computers via an IEEE 802.11g wireless or a wired ethernet link.

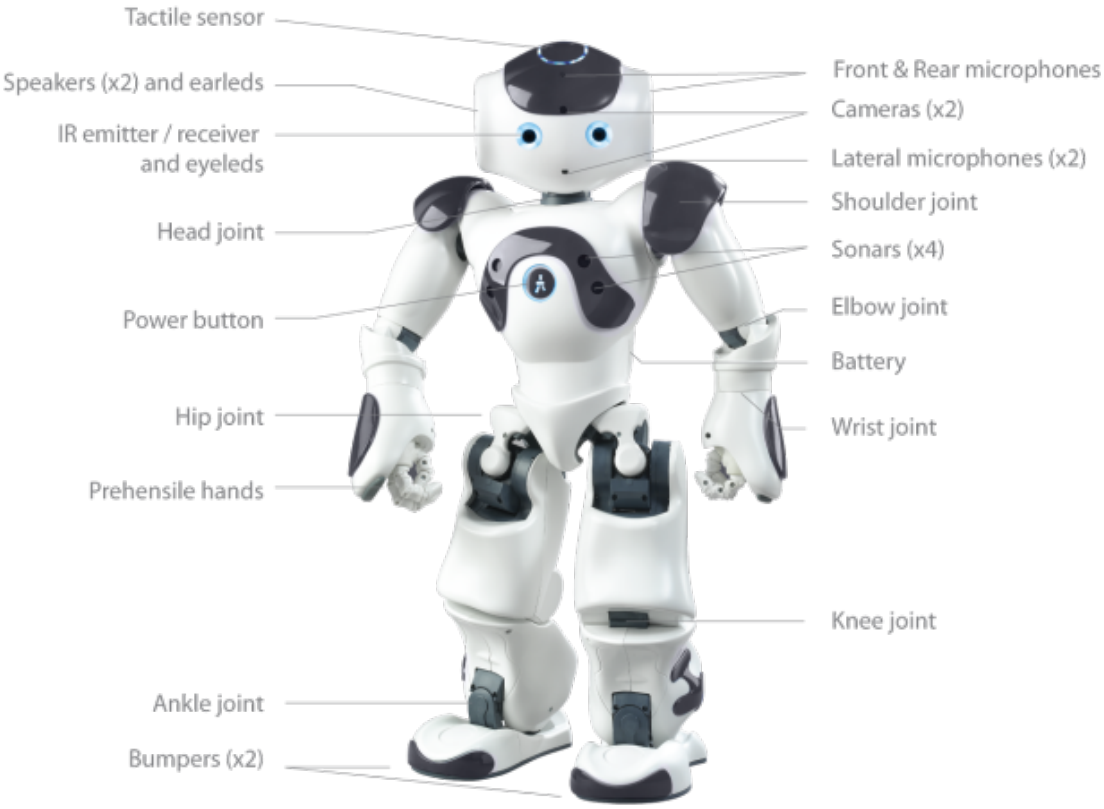


Figure 2.7: Aldebaran Nao V3.3 humanoid robot

Nao V3.3 (RoboCup Edition) has 21 degrees of freedom; 2 in the head, 4 in each arm, 5 in each leg and 1 in the pelvis (there are two pelvis joints which are coupled together on one servo and cannot move independently). In particular, these joints are the following:

- *Head*: HeadYaw, HeadPitch
- *Left Arm*: LShoulderPitch, LShoulderRoll, LElbowYaw, LElbowRoll
- *Right Arm*: RShoulderPitch, RShoulderRoll, RElbowYaw, RElbowRoll
- *Left Leg*: LHipRoll, LHipPitch, LKneePitch, LAnklePitch, LAnkleRoll
- *Right Leg*: RHipRoll, RHipPitch, RKneePitch, RAnklePitch, RAnkleRoll
- *Pelvis*: LHipYawPitch, RHipYawPitch

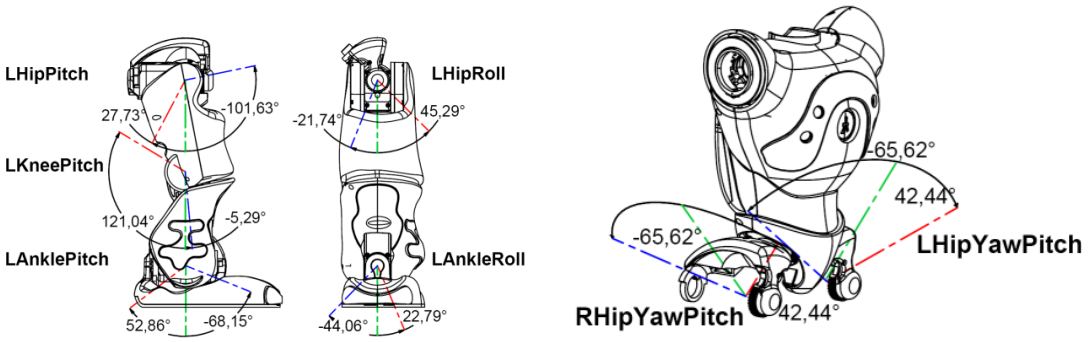


Figure 2.8: Aldebaran Nao V3.3 leg joints

The left leg and the pelvis joints are shown in Figure 2.8.

Nao V3.3 (RoboCup Edition) features a variety of sensors. Two cameras are mounted on the head in vertical alignment providing non-overlapping views of the lower and distant frontal areas (Figure 2.9), but only one is active each time and the view can be switched from one to the other almost instantaneously. Each camera is a 640 × 480 VGA camera at 30 fps. Four sonars (two emitters and two receivers) on the chest allow Nao to sense obstacles in front of it. In addition, the Nao has a rich inertial unit, with two 2-axis gyroscope and one 3-axis accelerometer, in the torso that provide real-time information about its instantaneous body movements. Two bumpers located at the tip of each foot are simple ON/OFF switches and can provide information on collisions of the feet with obstacles. Finally, an array of force sensitive resistors on each foot delivers feedback of the forces applied to the feet, while encoders on all servos record the actual joint position at each time. The audio devices for the Nao are two loudspeakers with frequency range up to 20 kHz and two microphones with sensitivity -40 +/- 3 dB and frequency range 20 Hz - 20 kHz. Sensor and audio devices are shown in Figure 2.7.

2.3.2 NaoQi

Nao's operating system is a Linux distribution for embedded systems customized by Aldebaran Robotics for use with the robot. As an interface to the robotic hardware, Aldebaran Robotics provides the NaoQi framework, which is capa-

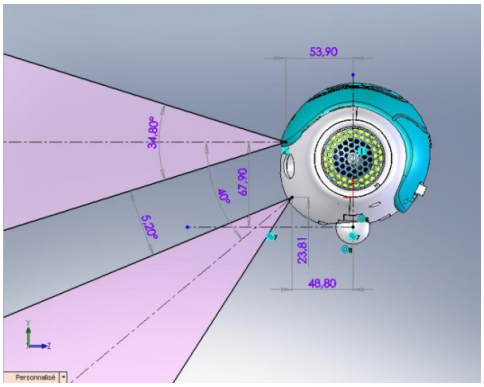


Figure 2.9: Nao's field of view

ble of controlling the robot's actuators and gathering sensors information. This framework allows transparent communication between different modules (motion, audio, video), transparent programming, and transparent information sharing. The NaoQi framework is cross-platform, meaning that it is possible to develop code on top of it using any operating system (Windows, Linux, or Mac). It is also cross-language, offering an identical API for both C++ and Python, and provides introspection, meaning that the framework knows which functions are available in the different modules and where. The NaoQi executable which runs on the robot acts a broker. When it starts, it loads a settings file called `autoload.ini` that defines which libraries should be loaded. Each library contains one or more modules which use the broker to advertise their methods. The broker provides lookup services, so that any module in the tree or across the network can find any method that has been advertised. The process of loading modules forms a tree of methods attached to the modules and of modules attached to the broker. These concepts are illustrated in Figure 2.10.

The ALMotion Module provides methods that make the Nao move. It contains commands allowing the user to manipulate joint stiffness and individual joint angles. In general, the user can execute any simple motion with the help of the ALMotion module. Apart from the execution of simple motion commands, NaoQi provides an interface to more complex body-part movements in Cartesian space, as well as an omni-directional walk engine.

An important component inside the NaoQi middleware is the Device Com-

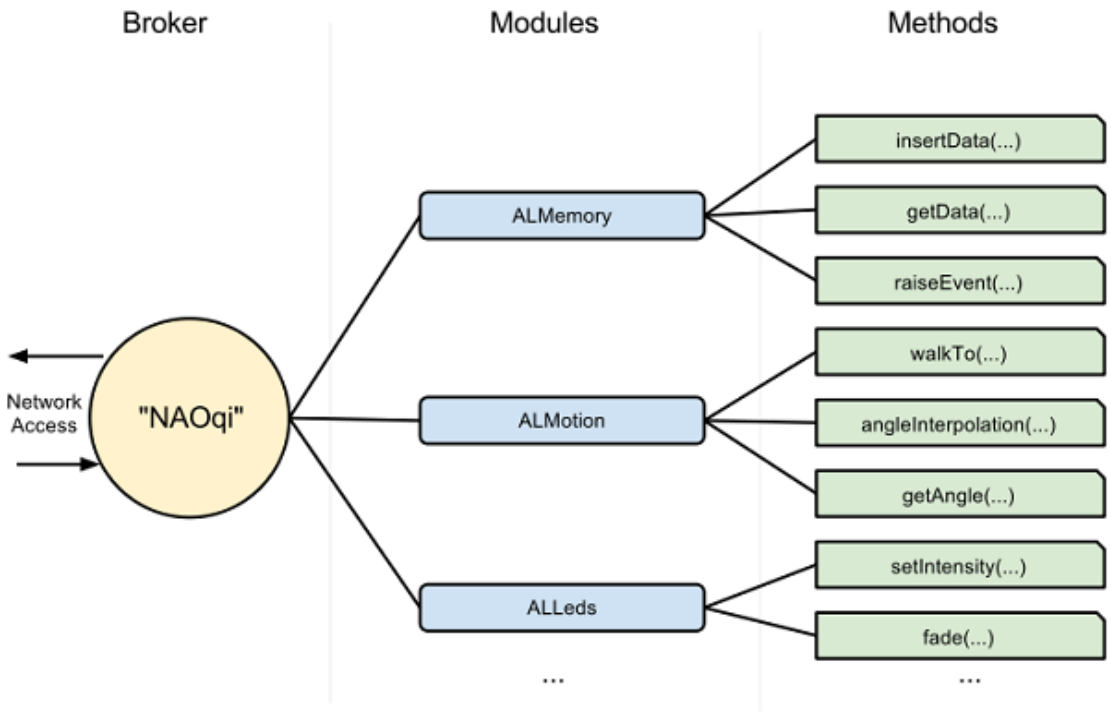


Figure 2.10: The tree of the broker, modules, and methods in NaoQi

munication Manager (DCM). The DCM is in charge of communicating with all electronic devices on the robot (boards, sensors, actuators, etc.) with the exception of microphones, speakers, and cameras. It manages the main communication line, the USB link between the onboard computer and the control board located in the chest and establishes the I2C link with the devices in the robot head. Thus, the DCM is the link between the "upper-level" software and the "lower-level" software. Modules like "Motion" and "LEDs" can send commands directly to actuators through DCM, while extractors and other modules can obtain sensor readings in the shared memory by the DCM.

2.4 The Architecture MONAS

The team's code is based on MONAS, a software architecture developed in-house to address the needs of the team [4]. MONAS provides an abstraction layer

of the Nao robot and allows the synthesis of complex robotic teams in various ways: (a) NaoQi modules, (b) XML-specified MONAS agents, and (c) model-based (ASEME) statechart agents. Apart from the trivial method (a), method (b) allows the coding of different agents (e.g. perception, motion, behavior, etc.) each executed independently at any desired frequency completing a series of activities at each execution. Method (c) is based on the Agent Systems Engineering Methodology (ASEME) [5], whereby the target team behavior is specified through a series of model-based transformations as a statechart, which is executed using a generic multi-threaded statechart engine [6]. A high level view of MONAS is shown in Figure 2.11.

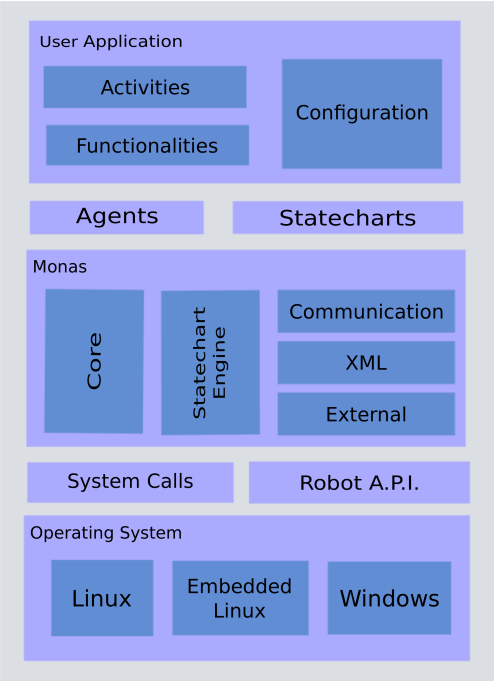


Figure 2.11: MONAS Architecture

2.4.1 Activities

In MONAS Architecture, as shown in figure 2.11, the code is classified in activities. Each activity has a specified role, so the code is very easy to use and easily altered. The most basic activities are “MotionController”, “Vision”, “Behavior”.

Also there are other activities that are important for the better performance of the robot such as the "Localization", "ObstacleAvoidance" and "RobotController". The activities that are used in this thesis are "MotionController" and "Sensors". The MotionController is the activity that controls all the motions of the robot (walk, kicks etc). Except the control part, the MotionController activity has the ability to read and process the data from files that are created for the special actions. The files are divided in two categories, the kme files and the xar files. The kme files are designed using Kouretes Motion Editor (KME). Further analysis of the KME will be done in the next chapter 4.1. The xar files are created by the motion editor Choregraphe, a tool designed by Aldebaran robotics for the robot Nao. The tool Choregraphe can also produce python and cpp code that corresponds to a special action. The Sensors activity has the ability to read the values from the sensors of the robot (gyroscope, joints etc) and also send setpoints to sensors.

2.4.2 Kouretes Motion Editor

Kouretes Motion Editor or KME is a tool created in 2009 for the needs of the Kouretes Team [7]. It is used for designing complex motion patterns on robots with many degrees of freedom like the Nao. Now the Team uses version 2 of the KME tool that was developed shorty after. The new version of the tool has two separated windows the Motion Editor (figure 2.12) and the Motion Designer (figure 2.13).

Any complex motion pattern created using KME is a timed sequence of robot poses in the configuration space. A set of the values of the 21 joints is called a pose. A motion consists of one or more poses. Robot poses can be created either by setting joint values through the sliders of the GUI or by capturing the current joint values of the real or simulated robot. As shown in figure 2.12 the current robot pose coded in the sliders can be captured and stored at the end of the sequence using the "Store Pose" button or right after the currently selected position using the "Insert Pose" button. The tool can update the currently selected pose with new values automatically in Motion Editor. Alternatively, for pose generation the user may manually move the robot joints to any desired

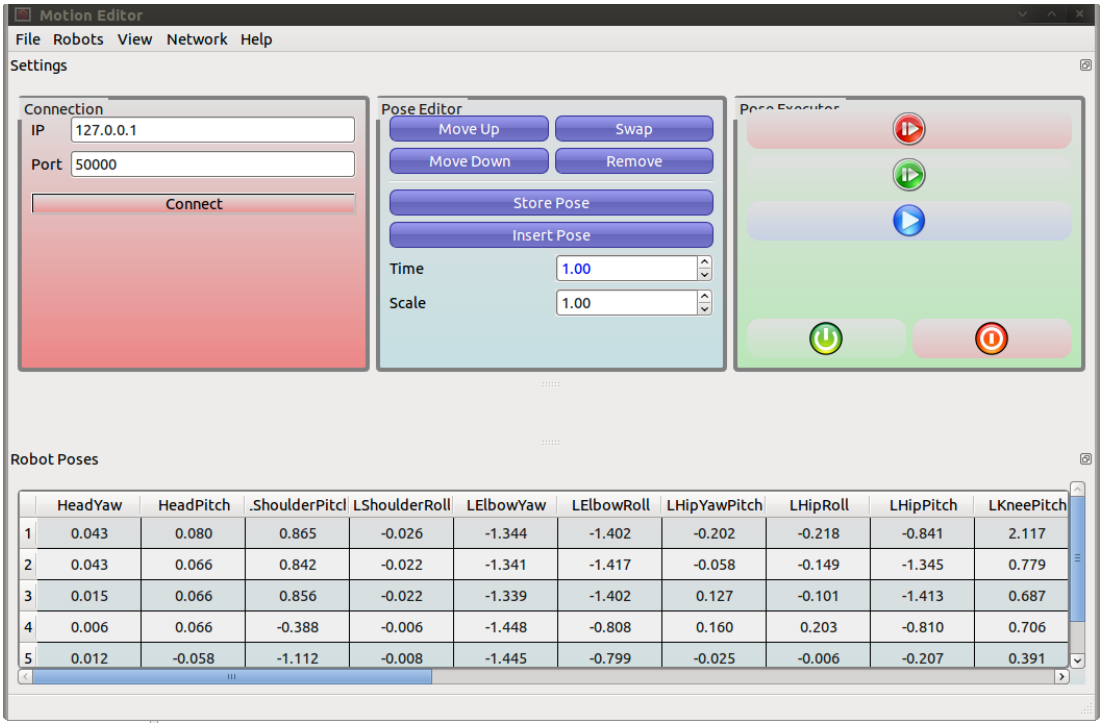


Figure 2.12: Kouretes Motion Editor Version II, The Editor

configuration in Motion Designer. If there is an empty pose in Motion Editor, it will automatically be filled with the current joint values, else the selected pose will be altered so the user must be careful. The user can also adjust the transition time between subsequent poses, which implicitly determines the speed in the motion of each joint. Note that the time value stored with each pose is the transition time from the previous pose to the current one. Finally, the user can "play" the current pose sequence from any point (either in a step-by-step fashion or continuously) to observe the complete motion pattern on the robot. Once the desired movement is complete, the pose sequence can be exported to a file and can be further used within any robot controller by simply invoking a motion execution routine.

A pose of a .kme file has the following form, play%[+/-]JointValue%...%[+/-]JointValue%Time. Each line of the file represents a pose as shown in figure 2.14. The values of the 21 joints are separated with the symbol % and at the end of the pose the time that is needed for the pose to complete is set.

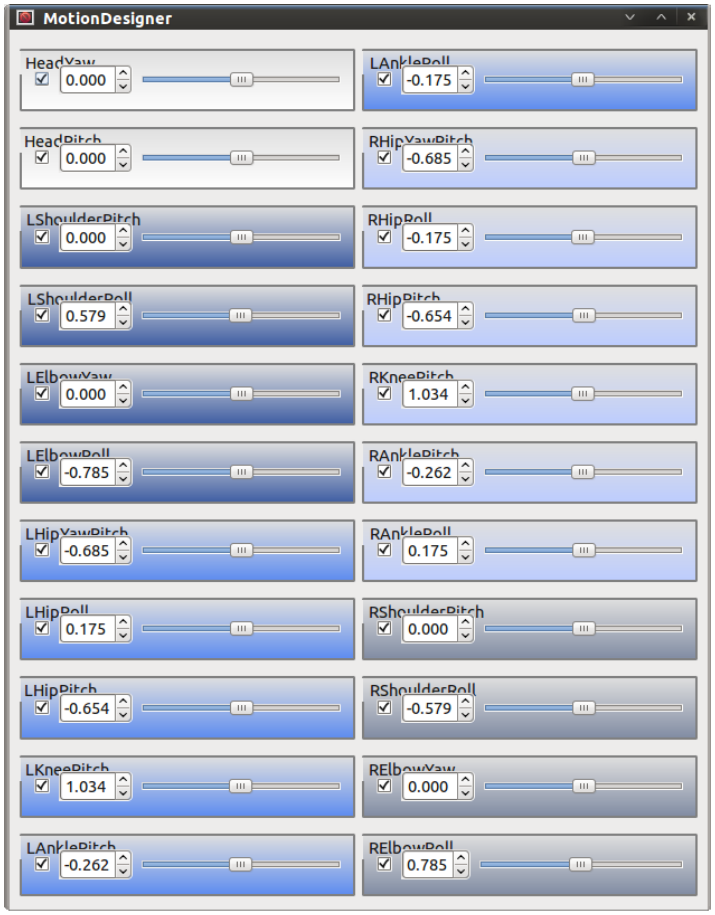


Figure 2.13: Kouretes Motion Editor Version II, The Designer

zeroPosToSitPos.kme

```
1 play%0.015%-0.031%1.542%0.127%-0.250%-0.035%0.000%0.000%0.000%0.000%0.000%0.000%0.000%0.000%0.000%
  0.000%-0.001%0.000%1.529%-0.146%0.400%0.035%1.000
2 play%0.110%-0.049%1.341%0.072%-0.368%-0.035%0.000%0.000%-0.973%2.113%-1.189%0.028%0.000%0.112%-0.940%
  2.113%-1.186%-0.051%1.359%-0.074%0.503%0.035%0.700
```

Figure 2.14: A KME file

Chapter 3

Problem Statement

3.1 The need of walk and kick interleaving

In the SPL, as it is described in Chapter 1, response time is crucial. The teams need to have a fast scan for the ball, a fast approach to the ball and of course a fast kick. However after all these years, that the SPL uses the humanoid robot Nao, all the teams have the previous characteristics. There are two main ways for the robot to acquire the ball. The first way is the "dribbling". The "dribbling" in a game of soccer with humanoids robots is pretty much like the dribbling in a game with humans, which means that the robot approaches the ball and continues to walk controlling the ball at its feet. Very few teams use this approach, because it is not very easy and sometimes it is more difficult to score with the dribbling. The second and most successful way, according to its use by the RoboCup SPL teams is the classical method, in which the Nao approached the ball, stopped, "stretched" its legs in order to take the kick starting pose to ensure stability and then kicked. This approach is easier than dribbling and also if the kick is properly oriented, a goal is almost certain. However this way makes the kicking of the ball a time consuming operation. Kouretes team uses this method but the competition between the teams is so high, that it is necessary to find a new faster way to acquire the ball.

Furthermore it is very important to remember the great phrase of Hiroaki Kitano which said "By the year 2050, to develop a team of fully autonomous humanoid robots that can win against the human world soccer champions". This

expression makes us think that the robots should act like humans. This means that the robots should find the ball very fast, run and kick the ball without stopping, like a human player would do.

Thus, the idea of the smoothly interleaving of the robot's walk with a special action, such as the kicks, was born. This new approach involves that the robot goes to the ball and when it gets close enough it directly shoots it, like in a real soccer game, so it does not lose valuable time in order to stop and take the correct position for the kick. It is very important to eliminate this time loss, so the robot will have more possibilities to score against the opponent. However the walk and kick interleaving is a difficult problem to solve, especially when the walk or the kicks are in a different level software. Kouretes Team uses the dynamic walk offered by Aldebaran Company for the robot Nao. The Aldebaran walk is a part of an upper lever software, which means that is not possible to make basic changes or intervene in the walk operation [8]. The design of a new walk engine by the team Kouretes it is very difficult because so far we doesn't have inverse kinematics. The kicks, that the team uses, belong to a lower level software and that's why the team designs them. These kicks are static, which means that it is not possible to intervene at any time slot. Although the kicks are not dynamic, the way they are designed by the Kouretes team, gives the opportunity to be able to modify or intervene in a motion with the help of its poses. Motions have a number of poses and it is not very difficult to start a motion from a specific pose, other than the first pose.

3.2 Related Work

In this section, a short overview of what other RoboCup teams have published about similar systems for walk and kick interleaving is given. So far there is only one similar work and belongs to the SPL team B-Human. Their approach is included in "B-Human Team Report and Code Release 2011" [9] and it is titled "In-Walk Kicks". B-Human are from the University of Bremen and the DFKI research area Safe and Secure Cognitive Systems and is the winning team in the SPL league since RoboCup 2009. Comming from the humanoid league and with a lot of experience they have made the most complete system among the teams.

They develop a WalkingEngine, which manages robot's walk and stand. In the Team Report, B-Human describe their implementation as follows:

Besides walking and standing the WalkingEngine has also minor kicking capabilities. The tasks of walking and kicking are often treated separately, both solved by different approaches. In presence of opponent robots, such a composition might waste precious time as certain transition phases between walking and kicking are necessary to ensure stability. A common sequence is to walk, stand, kick, stand, and walk again. Since direct transitions between walking and kicking are likely to let the robot stumble or fall over, the WalkingEngine is able to carry out sidewards and forward kicks within the walk cycle. Such an in-walk-kick is described individually for each kick type (forwards or sidewards) by a number of parameters, which are defined in the configuration files of the Config/Kicks/directory. On the one hand one configuration defines the sizes and speeds of the step before and during the kick, on the other hand a 6-D trajectory (three degrees in both translation and rotation). This 6-D trajectory overlays the original trajectory of the swinging foot and thereby describes the actual kicking motion (Figure 3.1). In doing so the overlaying trajectory starts and stops at 0 in all dimensions and also in position and velocity. Thus, the kick retains the start and end positions as well as the speeds of a normal step. The instability resulting from the higher momentum of the kick is compensated by the walk during the steps following the kick.

As shown the team B-Human developed a similar approach with us, but this issue will be discussed with details in Chapter 7.1.

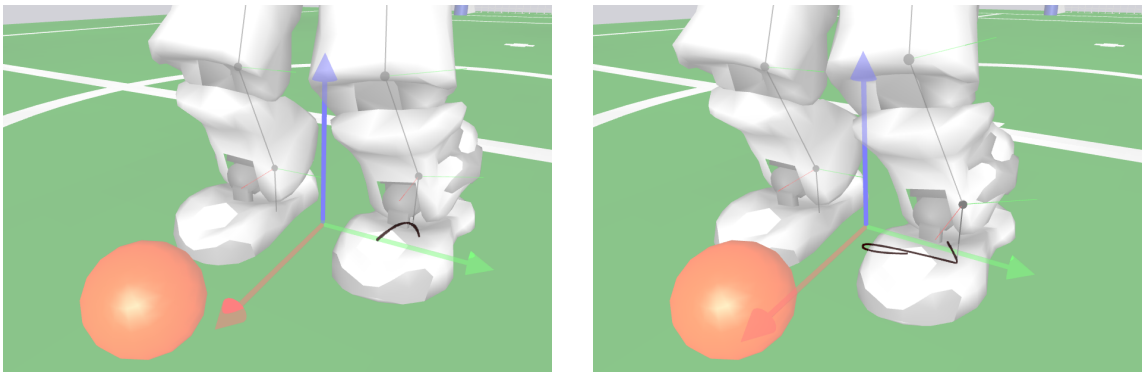


Figure 3.1: Visualization of swing foot trajectories (dark curves) while the robot approaches the ball. (left) A swing foot trajectory for walking forwards and (right) a modified swing foot trajectory for performing a sidewards in-walk kick to the right.

Chapter 4

Our Approach

4.1 Problem Approach

The Interleaving of motion skills such as the walk and the kicks is a very interesting but also a difficult domain. It is difficult because of the static kicks and the inaccessible dynamic walk. As it describes in chapter an array with the values of the 21 joints is called a pose. A motion consists of one or more poses. So, the main problem that had to be addressed was to match a pose of walk with a pose of a kick, so the interleave of the two motion skills can be carried smoothly. This means that the values of the joints of the robot's legs on a kick (forward, backward and sideward) and walking had to be read and compared. Maybe it seems easy to find the values of a joint of the robot but it not easy to compare this values to confirm if a pose of walk matches a pose of the kick. It is very important to find the right postures, where it is likely to become the transition from the walk to the kick, in order to have quite smooth transitions so the NAO doesn't fall down.

4.2 First Approach: Validation

Initially the implementation was done in the Python programming language just for validation reasons. A script was created by which the NAO begins to walk. Then the values of the joints "HipPitch", "HipRoll" and "KneePitch" for both

legs of 5 different poses of a forward kick (right or left) were placed in 5 different lists. Also a flag for each joint is stored in the appropriate list that determines the direction of the feet. For this reason the values of the joints of the previous pose minus the values of the joints of next pose are calculated. If the difference is positive the flag is set to 1 else the flag is set to -1, indicating that the feet in the two poses have different direction velocity. The values of the joints "HipPitch", "HipRoll" and "KneePitch" for both legs are read while the robot walks and are placed in another list. In order to determine the direction of the feet while the robot walks, the "1/-1" flag for each joint is also stored in the previous list. A control system was implemented, which compares the values of the joints while the robot is walking and the values in the 5 lists and checks if the joint values of the walk are between the n by n+1 poses of the kick, where n = 1,2,3,4. If the value of a joint is actually within the range of the joints between two adjacent poses, then a counter corresponding to that range is increased. Furthermore, this system compares the "1/-1" flags of the "walk" list and the 5 lists of poses for the same joint. If the flags are identical, then another counter is increased. Then the system checks if the two counters are getting values above 4, which means that at least four of the six joints match. If so, the algorithm halts the robot and at the same time the robot starts to kick from the one of the five kick poses that has the most matching joints, with the appropriate function call.
This implementation purpose was to verify the accuracy of the idea of the walk-kick interleaving. The success of this work lead to the next level of implementation which was done in MONAS Architecture and will be described in chapter 4.3.

4.3 Second approach : MONAS Architecture

The next step for the implementation was to develop a similar system to the one described above, inside the architecture MONAS.

4.3.1 Configuration File

Initially a configuration file was implemented, in which there are all the necessary information for the in-walk kicks. In this xml file, the name of the special action,

the names of the joints that will be compared and the number of those joints are included. The chosen joints are the "HipPitch", "HipRoll" and "KneePitch" for both legs and the "AnklePitch" and "AnkleRoll" of the standing leg during the kick. Also a system was implemented that finds the potential poses from the kicks that will be used for the walk-kick interleaving. This system stores the values of the joints of the poses from a motion and a walk file. The design of these files will be describe below in Chapter 4.3.3. Also the system stores the result values of the joints of the previous pose minus the values of the joints of the next pose both for the special action and the walk. Then the L2 norm of those values for each pose is calculated and if the norm is close to zero the pose is acceptable. An example of the pseudocode that implements the previous calculation and finds the norms of the values of the joints of the walk with the kick is shown below. The vector that contains the values of the joints of the kick and the corresponding differences is named k and the vector that contains the values of the joints of the walk is named w. The i shows the total poses of the kick, j shows the total poses of the walk and c shows the results of the L2 norm calculation.

Algorithm 1 Matlab pseudocode

for $i = 0...N-1$ **do**
  for $j = 0...M-1$ **do**
    $c(i,j) = (k(i) - w(j))^2 + c(i,j);$
    $c(i,j) = sqrt(c(i,j));$
  end for
end for

A threshold is defined for the values that are close to zero, that corresponds to the L2 norm results. The id numbers of the acceptable poses, the total number of those poses and the value of the threshold are also included in the xml file with the information for each motion. An example of this file is shown below.

```
<!-- Pose number id starts from 0-->

<Motion numOfPoses = 4 threshold = 0.7 numOfJoints = 8>
<actionName>KickForwardRight.kme</actionName>
<Joints>LHipRoll</Joints>
```

```
<Joints>LHipPitch</Joints>
<Joints>LKneePitch</Joints>
<Joints>LAnklePitch</Joints>
<Joints>LAnkleRoll</Joints>
<Joints>RHipRoll</Joints>
<Joints>RHipPitch</Joints>
<Joints>RKneePitch</Joints>
<Poses pose = 5></Poses>
<Poses pose = 7></Poses>
<Poses pose = 10></Poses>
<Poses pose = 25></Poses>
</Motion>

<Motion numOfPoses = 4 threshold = 0.7 numOfJoints = 8>
<actionName>KickForwardLeft.kme</actionName>
<Joints>LHipRoll</Joints>
<Joints>LHipPitch</Joints>
<Joints>LKneePitch</Joints>
<Joints>RHipRoll</Joints>
<Joints>RHipPitch</Joints>
<Joints>RKneePitch</Joints>
<Joints>RAnklePitch</Joints>
<Joints>RAnkleRoll</Joints>
<Poses pose = 5></Poses>
<Poses pose = 9></Poses>
<Poses pose = 14></Poses>
<Poses pose = 26></Poses>
</Motion>

<Motion numOfPoses = 3 threshold = 0.6 numOfJoints = 8>
<actionName>KickBackRight.kme</actionName>
<Joints>LHipRoll</Joints>
<Joints>LHipPitch</Joints>
<Joints>LKneePitch</Joints>
```

```
<Joints>LAnklePitch</Joints>
<Joints>LAnkleRoll</Joints>
<Joints>RHipRoll</Joints>
<Joints>RHipPitch</Joints>
<Joints>RKneePitch</Joints>
<Poses pose = 4></Poses>
<Poses pose = 8></Poses>
<Poses pose = 11></Poses>
</Motion>
```

4.3.2 Interleaving Algorithm

Those information are necessary for the next step of the implementation and also is the only part of the implementation that the user has to modify in order to add or change a special action. Furthermore those information are read and stored in structures for further use. The name of the special action, the number of a pose and the names of the joints that are stored in the file, make it is easy not only to find the file of the motion, but also the values of the joints of the poses for that motion and the difference of the values of the joints of the previous pose with the values of the joints of the next pose (rate of change). So, when the higher level decision send the name of the special action that is chosen to execute, the system easily provides the values, of the joints and the poses of the kick needed for comparison with the equivalent joint values of the walk which are read every time step. For the comparison of those values for each pose separately, the Euclidean norm/Euclidean distance was used. As mentioned before the vector that contains the values of the joints of the kick is called k and the rate of kick change, dk. The vector that contains the values of the joints of the walk is named w and the rate of walk change, dw. The values of the k vector are read from the xml file, while the values of the w vector are read in real time. So, the w vector, for the current values, is called w(t). The k vector for a specific pose is named k_i, where i is the number id of the pose. Thus, the $u_i(t)$ shows the calculation of the Euclidean norm for every i and it is represented as:

$$u_i(t) = \left\| \left(\binom{w(t)}{dw(t)} - \binom{k_i}{dk_i} \right) \right\|_2$$

An example of the pseudocode that implements the previous calculation and finds the norms of the values of the joints of the walk with the kick is shown below. The number of the chosen joints for the comparison is represented by the letter j and t represents the threshold for the corresponding kick.

Algorithm 2 Interleaving algorithm pseudocode

```
w(t) = current angles of the selected joints;
dw(t) = current gradients of the selected joints;
u_i(t) = 0;
du_i(t) = 0;
for i = 0...N − 1 do
    for j = 0...M − 1 do
        u_i(t) = (k_ij − w_j(t))^2 + u_i(t);
        du_i(t) = (dk_ij − dw_j(t))^2 + du_i(t);
    end for
    norm_i(t) = sqrt(u_i(t) + du_i(t));
    if norm_i(t) < threshold then
        best_entry_pose = i;
    end if
end for
execute kick k_i from best_entry_pose
```

If the Euclidean norm is less than the threshold that was set previously, this pose is chosen for the walk-kick interleaving. If there are more than one poses that have their Euclidean distance lower than the threshold, the chosen pose is the one with the highest id number. Also if there is not a pose that the Euclidean distance is less than the threshold, then the procedure is repeated. So when a right pose is chosen, the robot's walk is killed immediately and the special action is executed from that pose.

4.3.3 Motion Recording

In parallel with the implementation of the comparison algorithm, a system was developed that records all the motions of the robot, special actions and walk. A separate module was designed, in which the motion that is needed to be recorded, is called to execute. At the same time the values of all the joints of the robot are read and saved in a two-dimensional array, until the end of the motion. So, this array contains the values of the joints for an entire motion. Also it is possible for this module to record the robot's walk for a given time interval. When the motion is executed, the module saves the values of the joints in a file for each pose of the motion. Each line represents a pose with the values of the 22 joints.

Chapter 5

Implementation

The implementation of this work followed a number of different approaches that were implemented in different programming languages like Python for validation, Matlab for testing and of course C++ for the MONAS Architecture.

5.1 The walk-kick interleaving: Validation

In the first approach that is presented in Chapter 4.2, python's lists for the storage of the values of the joints are used. Also this implementation uses some functions that are offered by the Aldebaran company. More specifically the software module ALMotion and the functions getAngles() and killAll() were used. The function getAngles() returns a list of the current joint's values and it is used to get the values of the joints when the robot walks. After the pose for the interleaving is chosen, the function killAll() is called to kill the walk. The kicks that are used, are python script functions and were designed with the motion editor Choregraphe (Chapter 2.4.1).

5.2 The walk-kick interleaving: MONAS Architecture

The second approach, as it is presented in Chapter 4.3, was done inside the MONAS architecture. Two implementations with similar method and comparison

algorithm were created. The fist attempt was done with the help of the Aldebaran motion editor (Choregraphe) and Aldebaran modules. Although this method succeeded the walk and kick interleaving, it had a few disadvantages so we decided to implement a new methodology. This time the Kouretes Motion Editor was used and instead of the Aldebaran module for the motion execution, the DCM software module was used. The second attempt faced and solved the problems of the previous implementation and achieved the walk and kick interleaving goal. The following images 5.1 show the current process for the execution of a special action and the process that will be implemented for the execution of both simple and interleaved kicks.

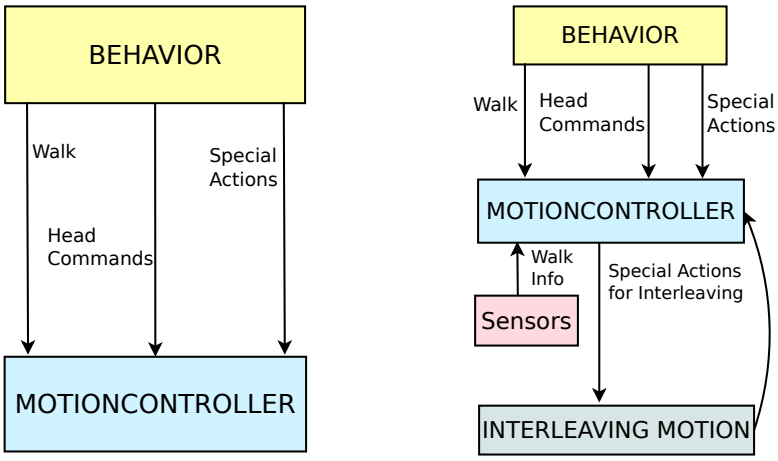


Figure 5.1: Old approach diagram(left) and new approach diagram(right)

5.2.1 Initial Implementation

In the first attempt to implement the walk and kick interleaving, the ALMotion module and the Choregraphe were used. The implementation of the comparison algorithm is similar to the one of the 1st Approach that was implemented in the Python programming language. The implementation followed the steps described below:

New motion's design

Initially, the motion editor Choregraphe, was used in order to design the special actions. The motion editor Choregraphe is a user friendly tool, with many possibilities in the design of a motion. So a forward kick was designed, that consists of a number of poses. After testing with the robot Nao, the possible poses that allow a smooth transaction were found. However we had to face the problem that the Choregraphe provides a motion file (.xar) in which no access is allowed after the design of it. During the design of a special action intervation is allowed, but after the export of the motion in a XAR file, the motion can only be executed. So it is not possible to start the kick from the appropriate pose for the interleaving. To overcome this problem it was decided to copy the initial kick and delete the poses before the first possible pose for the transaction. Having found four possible poses, we had to design four new kicks which start from a different, appropriate, pose.

XML file update

After the design of the new kicks, it is important to update the xml file with the details of every special action. The file has to contain:

- the name of the special action,
- the names of the joints that will be compared,
- the values of the joints for each chosen pose,
- and the number of those joints,

After a lot of experimentation the joints that are chosen for comparison, are the "HipPitch", "HipRoll" and "KneePitch" for both legs. The joints of hands are useless and the other joints of the legs are unnecessary.

Comparison Algorithm

Now the MONAS Architecture can read the xml file, put the values of the joints for each chosen pose of a kick to a vector (Sequence containers) and then save all the details of this kick to a map (Associative containers) using as key, the name

of the kick. This map is used in MotionController. The first job that the MotionController does for the walk-kick interleaving, after the reading of the map, is the comparison of the joint's values of a pose of a motion with the values of the same joints of the previous pose of that motion. If the value of the joint of the previous pose is smaller than the next pose, a flag is set to 1 else the flag is set to -1, indicating that the feet in the two poses have different direction velocity. Those flags are stored in an array explicit to the specific pose. Also the MotionController reads from the sensors of the robot the values of the joints, that are needed for comparison, while the robot is walking. Furthermore the values of the same joints of the current pose of the walk with the previous pose which is saved in every circle are compared. Just like before if the value of the joint of the previous pose is smaller than the next pose, a flag is set to 1 else the flag is set to -1. Thus, both the values of the joints from the kick and from the walk are ready to be compared. A control system was implemented. which in every time step, compares the values of the joints while the robot is walking and the values of the map for the kick and checks if the joint values of the walk are between the n and n+1 poses of the kick, where n = 1,2,3,4. If the value of a joint is actually within the range of the joints between two adjacent poses, then a counter corresponding to that range is increased. Furthermore, this system compares the "1/-1" flags of the "walk" list and the array with the flags of the poses for the same joint. If the flags are identical, another counter is increased. Then the system checks if the two counters are getting values above 4, which means that at least four of the six joints match for this pose. If so, the appropriate motion file is chosen that has as starting pose of the kick the pose with the more matching joints.

Motion Execution

In the next step, the algorithm halts the robot with the function killWalk() from ALMotion module and at the same time the motion is sent to ALMotion module for execution and the robot starts to kick from the chosen kick pose.

This approach has some disadvantages. If many poses are chosen for the transitions, a lot of motion files are needed and this is expensive in storage. Also this method is more difficult if a new kick is designed, because the user has to

modify the motion files. Thus we proceeded in another implementation that is more flexible and more user friendly.

5.2.2 Final Implementation

This implementation was done in order to fix the disadvantages of the previous design. This time the Kouretes Motion Editor (KME) was used for the design of the special actions and the Device Communication Manager (DCM) was used for the execution of the motions. The DCM is the NAO software module that is in charge of the communication with every electronic devices in the robot (boards, sensors, actuators etc.) with the only exception of the sound (in or out) and the cameras. More details in chapter 2.3.2. The steps for this new design are the following :

New motion's design

The Kouretes Motion Editor (Chapter 4.1) was chosen because the .kme motion files are more flexible for reading or editing. The KME file gives the opportunity to the user to read the values of the joints contained and to copy a desirable value.

Motion Recording

A new special action can be designed by the KME tool, but also it is possible to design a KME file with our motion record module (Chapter 4.3.3). This module sends a motion to the MotionController activity for execution. The motion can be a XAR file, a KME file, a .cpp motion file or an Aldebaran's motion command. At the same time the Sensors activity reads the values of all the joints of the robot, until the end of the motion. The sampling frequency that the Sensors activity reads the values, is defined by the frequency of the thread that the Sensors activity runs. The module saves those values, in a two-dimensional array, for all the joints and all the poses. The total number of the poses is calculated by multiplying the time that the motion needs to execute with the sampling frequency. When the motion is executed the module stored the values that the array contains, in two files. The first file is a KME file, where the module stores the joint values, by using as separator the character %, and in the end of a line places the time that

the pose needs to complete. The module calculates this time, that is equal to the sampling period. The second file is a TXT file and its usefulness is shown in the next step, the Matlab system. This file contains only the values of the joints of the motion and has spaces as separators between those values. Except the special actions the module is able to record the robot's walk for a given time interval. This time, the command for execution will be the Aldebaran's command for the walk "setWalkTargetVelocity". For much better recording, the user can hold the robot in the air throughout the execution of the motion, in order to eliminate the vibration.

Matlab System

For the development of the xml file, that contains all the necessary information for the special actions, a system in Matlab was developed. Initially the motion record module recorded the XAR motion file of the forward kick and created the KME and the TXT file. Furthermore the robot's walk is recorded and both the TXT files are loaded in Matlab. The number id of the joints which will be compared, are chosen and placed in a matlab array. The values of the joints that are chosen are read from the two files and are placed in two arrays. In the same arrays, the results of the differences of the joint's values of the next poses with the previous poses both for the kick and the walk motion file are also placed. Thereafter there is a mathematical calculation of those matrices with the Euclidean norm. The results of the norm, provides the difference of the values of the joints of a kick pose with the values of the same joints of a walk pose. If the norm is close to zero, the transaction from the walk to the kick can be done with success for that specific pose. This method allows the distinguishing of the best poses for the walk-kick interleaving. Furthermore, it is very important to choose a good threshold just over the Euclidean norms of the poses that are selected, in order for it to be used in the comparison algorithm later. Examples and results of the Matlab implementation, that show how the appropriate poses are chosen for the transaction are shown in Chapter 6.

XML file update

After the design of the kicks and the extraction of the appropriate poses for

interleaving from the Matlab system, it is important to update the xml file with the details of every special action. So the xml file for the second attempt of the walk and kick interleaving contains:

- the name of the special action,
- the names of the joints that will be compared,
- the total number of those joints,
- the id number of each chosen pose for the transaction,
- the total number of those poses,
- and the threshold which was calculated,

After a few experiments we concluded that by adding some specific joints for the comparison, the results are better. So this time the joints that are chosen for the comparison, are the "HipPitch", "HipRoll" and "KneePitch" for both legs and also the "AnklePitch" and "AnkleRoll" of the standing leg during the kicking.

Comparison Algorithm

In this implementation, as described above, the xml file has to contain only the id numbers of the appropriate poses for the transaction and not all the values of their joints. So, except of the placement of the information of the xml file in associative containers (in a map), there is an extra processing to get the values of the joints from the names of the joints that we have in this case. This is a little complicated, because for the user it is easier to choose a joint with the name of it, but the format of the motion files recognize the id number of the joint in the chain of all the joints. Thus a function had to be implemented that takes the name of the joint and gives the id number that reflects to the joint inside the chain.

The information that are provided by the previous function allows the distinguish, from the motion file and from the appropriate poses,of the values of the joints that are needed. Furthermore the differences of the joint's values from the previous poses with the next poses are calculated. Those values are stored in two

different vectors. Also with the .kme motion file the total number of the poses of the kick can be read and can be used in order to execute the kick until the end. So this number and the two vectors, for each motion, are placed in the map with the special actions.
The MotionController activity uses this map with key the unique name of the special action. The joint's values from the sensors of the robot's walk are read and as in the first attempt the values of the joints of the previous walk pose are stored in a vector. In fact, the first time that the MotionController is called this vector is initialized to zeros. So when the comparison algorithm for the interleaving is called, a mathematical calculation of the Euclidean norm takes place for a) the values of the joints of the appropriate poses of the kick with the values of the same joints of the walk and b) the vector with the results of the differences of the joint's values of the kick with the vector of the results of the differences of the walk. From this mathematical calculation an array with norms is returned, with size, the total number of the appropriate poses for the transaction. After the norm calculation the values of the joints of the walk are saved in order to be used as previous walk pose the next time that the comparison algorithm is called. Thereafter if a norm of a pose is less or equal to the threshold that is calculated by the matlab system, this pose's id number is stored. If there are more than one such norms, the pose with the highest id number is stored. If no pose of the kick matches the current pose of the walk, the comparison algorithm is called again with the new pose of the dynamic walking.

Motion Execution

Then, the algorithm halts the robot with the function killWalk() from the ALMotion module. At the same time, having the id number of the pose that is qualified for the interleaving of the walk and the kick, the function which is doing the execution of the special action is called, with arguments the id number of the starting pose and the id number of the ending pose of the kick. The default value for the second argument is the total number of the poses that we get from the KME file, as described above. The function that executes the special action is named "ExecuteFrameDCM". It has all the necessary information for the motion and needs only the two arguments that were described above. With the number

of the starting pose, the function ignores the unnecessary previous poses. Furthermore this function fills the fields of the Alias with the values of the joints and the execution time of the poses that are needed by the DCM. The Alias is a way to send faster commands to many actuators at the same time. The creation of the Alias for a special action is done the first time that the MotionController is called and contains the Alias name, and the list of the actuator names. So in the function ExecuteFrameDCM, as it is described above, the appropriate information for the motion are set and when the Alias is ready, it is sent to the DCM for execution.

As a result, this approach managed to overcome the difficulties and the disadvantages of the previous implementation. So more special actions were designed to be tested with this implementation. In the same way the backward and the sideward kicks are designed and incorporated in our code. After a few tests the walk and kick interleaving of those kicks succeeded.

Finally the most important achievement of this approach is the successful completion of our goal. As shown the walk and kick interleaving isn't an impossible goal but a major issue for treatment.

Chapter 6

Results

6.1 Matlab Results

In Chapter 5.2.2 the implementation that solves our problem with success is presented. As shown above, after the design of the kick, the first step is the development of a Matlab system, which finds the possible poses for the transition from the robot's walk to the kick. The figures that helped to extract those entry poses are presented and analysed in this chapter. The figures show the results of the Euclidean norm of two arrays. The first array has the values of the chosen joints of all the poses of a kick and also has the results of the differences of the joint's values of the next poses with their previous poses. The second array has the values of the corresponding to the kick joints of some poses of the dynamic walk and the results of the differences of the joint's values of the next poses with their previous poses of the walk. The vertical axis of the figures is for the multitude of the poses of the kick and the horizontal axis for the multitude of the poses of the walk. The number id of a pose that is taken from the matlab system, it is in the range 1 to the total poses of the motion. The number id of the pose which will be written in xml file it is in range 0 to the total poses of the motion minus 1. So, if in the figure an entry pose has id number 6, in the xml file will be written with the id number 5.

- The first figure 6.1 shows the norms of the left forward kick with the walk. As shown in the figure, the results of the Euclidean norms for the initial

poses have deep blue color. Those norms are close to zero indicating that the corresponding poses of the walk are close enough to the corresponding poses of the kick. Furthermore the results of the Euclidean norms for the kick poses with id number in the range of 5-15 have light blue color. After

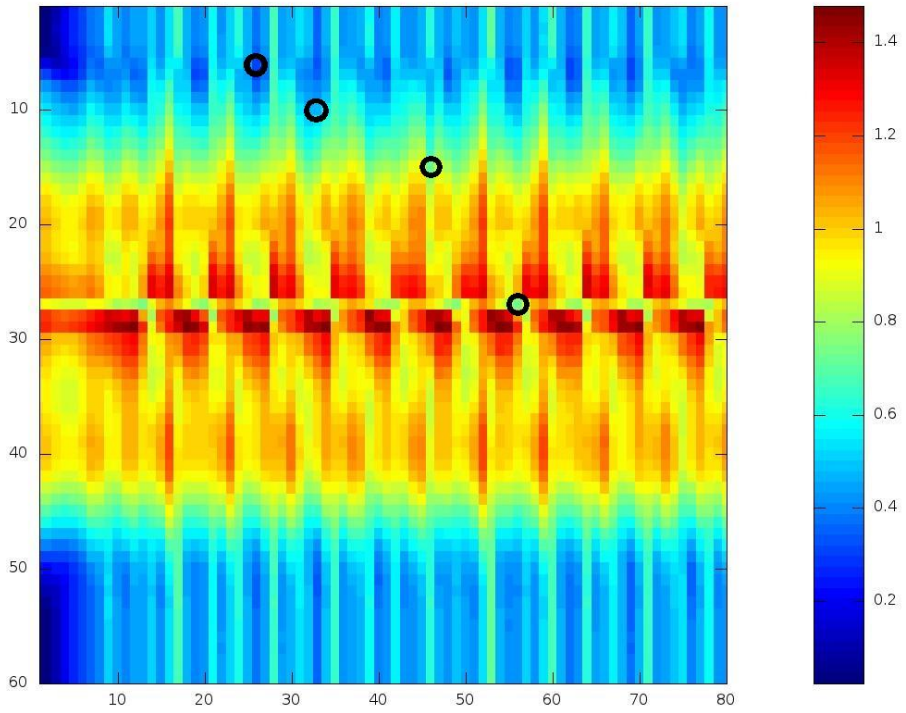


Figure 6.1: Norms of the left forward kick and robot's walk

the poses with id number 15, the results become green and yellow. Thus, the norms take values in the space of 0.8 to 1.0 and the corresponding poses of the walk are far from the corresponding poses of the kick. The red color shows that the results of the norms are very high and the two motions are quite different in those poses. It is very possible that the red color appears in the poses that the robot has its leg in the air while kicking the ball. In the pose with id number 45 the results of the Euclidean norms, become again close to zero until the end of the kick. This means that the part of the robot's kick finished and the remaining poses of the motion are

for the return of the robot to a appropriate pose (a pose near to the walk pose). Also, as shown in the figure, in the walk pose with id number 16 there is a peak which is repeated every 8 walk poses. Those peaks give the periodicity of the kick over the walk. After the search of some values for the threshold, we concluded that a good threshold for the left forward kick is the 0.7. Under this value there are a few poses which can make the stable interleaving from the walk to the kick stable enough. The part of the xml file for the entry poses of the left forward kick that were chosen is shown below.

```
<Poses pose = 5></Poses>
<Poses pose = 9></Poses>
<Poses pose = 14></Poses>
<Poses pose = 26></Poses>
```

- The second figure 6.2 shows the norms of the right forward kick with the robot walk. As shown in the figure, same as before the results of the Euclidean norms for the initial poses have deep blue color. Those norms are close to zero indicating that the corresponding poses of the walk are close enough to the corresponding poses of the kick. Furthermore the results of the Euclidean norms for the kick poses with id number in the range of 5-15 have light blue color. After the poses with id number 15, the results become green and yellow. Thus, the norms take values in the space of 0.8 to 1.0 and the corresponding poses of the walk are far from the corresponding poses of the kick. Same as before the red color shows that the results of the norms are very high and the two motions are quite different in those poses. In the pose with id number 44 the results of the Euclidean norms, become again close to zero until the end of the kick. This means that the part of the robot's kicking finished and the remaining poses of the motion are for the return of the robot to an appropriate pose. Also, as shown in the figure, in the walk pose with id number 12 there is a peak which is repeated every 8 walk poses. Those peaks give the periodicity of the kick over the walk. Same as before after the search of some values for the threshold, we concluded that a good threshold for the right forward kick is the 0.7, like

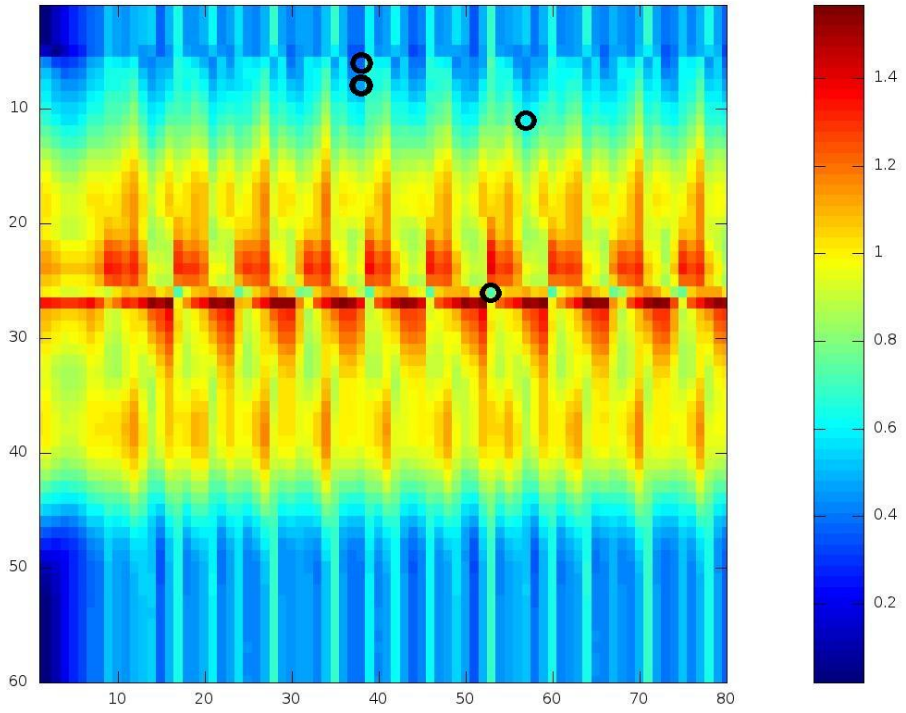


Figure 6.2: Norms of the right forward kick and robot's walk

the threshold for the left forward kick. Under this values there are a few poses which can make the stable interleaving from the walk to the kick. The part of the xml file for the right forward kick is shown below and it isn't the same with the part of the left forward kick because the two kicks have differences in their design.

```
<Poses pose = 5></Poses>
<Poses pose = 7></Poses>
<Poses pose = 10></Poses>
<Poses pose = 25></Poses>
```

- Continuing to the next figure 6.3 the norms of the left side kick with the walk are shown.As shown in the figure, the results of the Euclidean norms for the initial poses have blue/light blue color. Those norms are close to

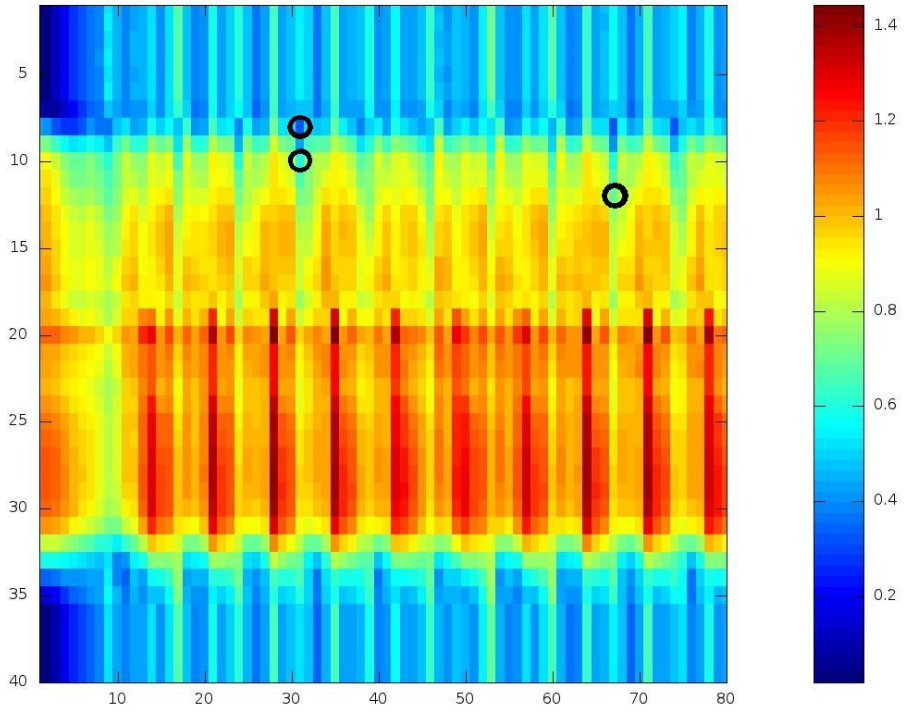


Figure 6.3: Norms of the left side kick and robot's walk

zero indicating that the corresponding poses of the walk are close enough to the corresponding poses of the kick. After the pose with id number 10, the results become yellow. Thus, the norms take values in the space of 0.8 to 1.0 and the corresponding poses of the walk are far from the corresponding poses of the kick. The red color shows that the results of the norms are very high and the two motions are quite different in those poses. It is very possible that the red color appears in the poses that the robot has its leg in the air for the kick. In the pose with id number 32 the results of the Euclidean norms, become again close to zero until the end of the kick. This means that the part of the robot's kicking finished and the remaining poses of the motion are for the return of the robot to an appropriate pose. Also, as shown in the figure, in the walk pose with id number 14 there is a peak which is repeated every 8 walk poses. Those peaks give the periodicity of

the kick over the walk. This time a good threshold for the left side kick is the value 0.5. Under this values there are a few poses which can make the stable interleaving from the walk to the kick. The part of the xml file for the left side kick is shown below.

```
<Poses pose = 7></Poses>
<Poses pose = 9></Poses>
<Poses pose = 11></Poses>
```

- The figure 6.4 shows the norms of the right side kick with the walk. As

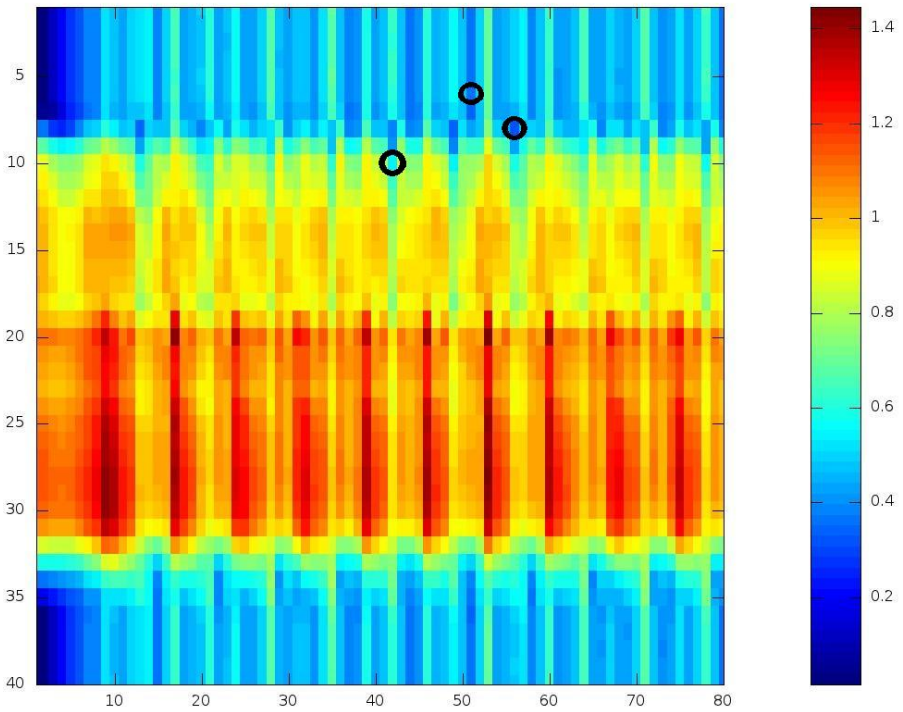


Figure 6.4: Norms of the right side kick and robot's walk

shown in the figure, the results of the Euclidean norms for the initial poses have blue/light blue color. Those norms are close to zero indicating that the corresponding poses of the walk are close enough to the corresponding poses of the kick. After the pose with id number 10, the results become

yellow. Thus, the norms take values in the space of 0.8 to 1.0 and the corresponding poses of the walk are far from the corresponding poses of the kick. The red color shows that the results of the norms are very high and the two motions are quite different in those poses. In the pose with id number 32 the results of the Euclidean norms, become again close to zero until the end of the kick. This means that the part of the robot's kicking finished and the remaining poses of the motion are for the return of the robot to an appropriate pose. Also, as shown in the figure, in the walk pose with id number 10 there is a peak which is repeated every 8 walk poses. Those peaks give the periodicity of the kick over the walk. As before a good threshold for the right side kick is the 0.5. Under this values there are a few poses which can make the stable interleaving from the walk to the kick. The part of the xml file for the poses of the right side kick that were chosen is shown below. The poses are not the same with the poses of the left side kick because the two kicks have differences in their implementation.

```
<Poses pose = 5></Poses>
<Poses pose = 7></Poses>
<Poses pose = 9></Poses>
```

- Continuing to the next figure 6.5 the norms of the left back kick with the walk are shown. As shown in the figure, same as before the results of the Euclidean norms for the initial poses have deep blue color. Those norms are close to zero indicating that the corresponding poses of the walk are close enough to the corresponding poses of the kick. Furthermore the results of the Euclidean norms for the kick poses with id number in the range of 10-19 have light blue color. After the pose with id number 19, the results become green and yellow. Thus, the norms take values in the space of 1.0 to 1.3 and the corresponding poses of the walk are far from the corresponding poses of the kick. The red color shows that the results of the norms are very high and the two motions are quite different in those poses. In the pose with id number 42 the results of the Euclidean norms, become again close to zero until the end of the kick. This means that the part of the robot's

kicking finished and the remaining poses of the motion are for the return of the robot to an appropriate pose. Same as before, in the walk pose with id number 27 there is a peak which is repeated every 8 walk poses and gives the periodicity of the kick over the walk. For the left back kick, the threshold is defined to the value 0.6. Under this values there are a few poses which can make the stable interleaving from the walk to the kick. The part of the xml file for the poses of the left back kick that were chosen is shown below.

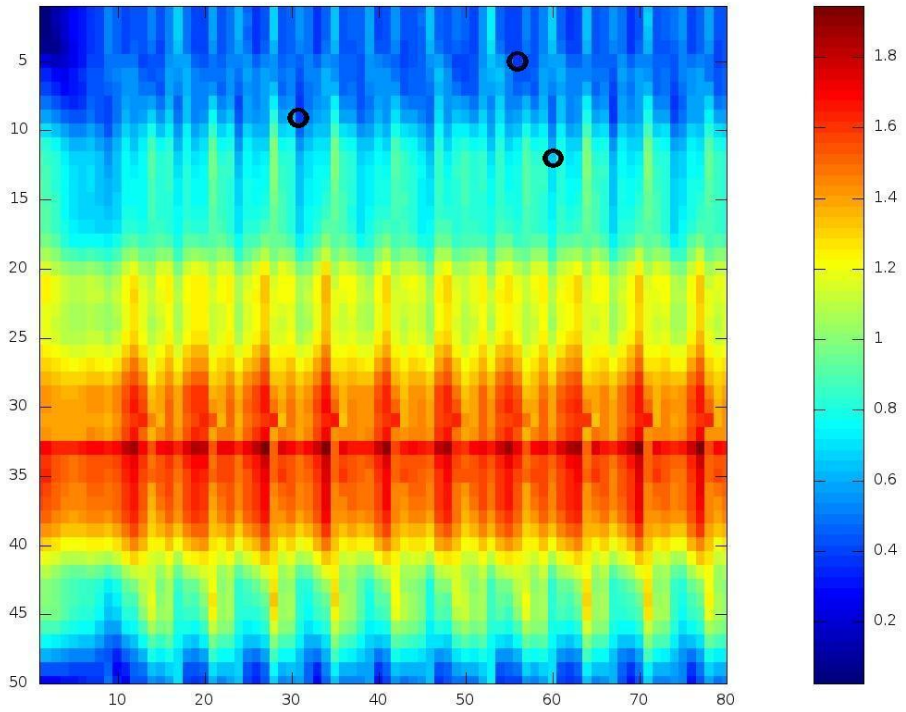


Figure 6.5: Norms of the left back kick and robot's walk

```
<Poses pose = 4></Poses>
<Poses pose = 8></Poses>
<Poses pose = 11></Poses>
```

- The last figure 6.6 shows the norms of the right back kick with the walk. As shown in the figure, same as before the results of the Euclidean norms for the initial poses have deep blue color. Those norms are close to zero indicating that the corresponding poses of the walk are close enough to the corresponding poses of the kick. Furthermore the results of the Euclidean norms for the kick poses with id number in the range of 10-19 have light blue color. After the pose with id number 19, the results are light blue and

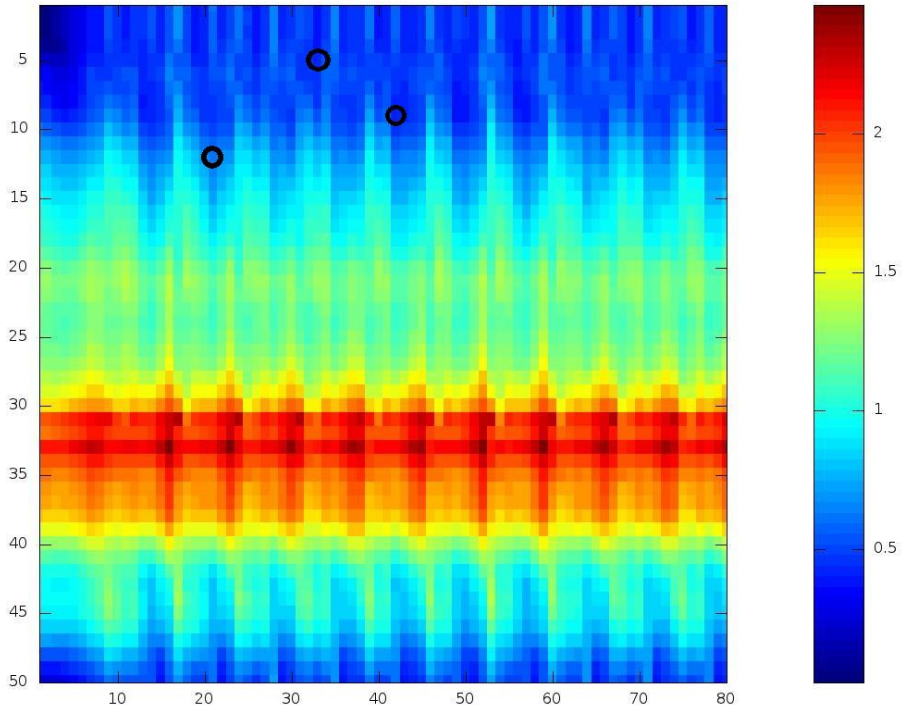


Figure 6.6: Norms of the right back kick and robot's walk

green and after the pose with id number 22, the results become yellow. The red color shows that the results of the norms are very high and the two motions are quite different in those poses. It is very possible that the red color appears in the poses that the robot has its leg in the air for the kick. At the pose with id number 41 the results of the Euclidean norms, become

again close to zero until the end of the kick. This means that the part of the robot's kicking finished and the remaining poses of the motion are for the return of the robot to an appropriate pose. Also, as shown in the figure, in the walk pose with id number 16 there is a peak which is repeated every 8 walk poses. Those peaks give the periodicity of the kick over the walk. Also for this kick, a good threshold is defined to the value 0.6. Under this value there are a few poses which can make the stable interleaving from the walk to the kick. The part of the xml file for the right back kick is shown below. The entry poses are the same with the entry poses of the left back kick because the one kick is the mirroring of the other.

```
<Poses pose = 4></Poses>
<Poses pose = 8></Poses>
<Poses pose = 11></Poses>
```

6.2 Old approach vs New approach

This chapter shows the progress that this implementation provided, compared to the previous approach/shooting of the ball. The following table 6.1 presents the average time that the robot needs to go to the ball from a specific distance and shoot the ball without stopping (walk and kick interleaving) and the average time that the robot needs to go to the ball from the same distance and kick the ball after stopping and taking the correct kick pose (old approach). The previous way

Forward	Sideward	Backward
Old way | New way	Old way | New way	Old way | New way
19.0 sec | 12.0 sec	18.0 sec | 11.5 sec	19.0 sec | 13.0 sec

Table 6.1: Average time of approaching and kicking the ball for the two methods

of approaching and kicking the ball has not the Nao's full speed like the current proposal, but has a lower speed that is calculated by taking into account the ball's distance. The robot is placed on the penalty mark and the ball is placed

on the center of the circle, so the distance is 1.2 meters. The table describes the three special actions, the forward kick, the sideward kick and the backward kick.

As we can see the timing results of the new implementation are much better than those of the previous implementation. More accurately, for the forward kick, the deference of the 7 seconds is great. So this thesis has achieved its purpose, to reduce the total time for approaching and kicking the ball by the Nao robot.

Furthermore the two different approaches are shown in the next snapshots to explain the time deference of the previous method of approaching and shooting the ball with the walk and kick interleaving.

The first snapshots 6.7 referred to the forward kick. The snapshots are separated in three columns. The two first columns represent the previous approach that the robot approaches the ball, stops, takes the appropriate pose and kicks the ball forward. The snapshots of the right column show that the Nao robot approaches the ball and kicks forward when it is close enough, without stopping. So, in the three first snapshots of the left column, the robot walks, in the next two snapshots the robot tries to align with the ball. The next snapshot shows that the robot takes the appropriate posture for the kick and after this snapshot the robot starting the forward kick. In the two first snapshots of the right column the robot's walk is shown and in the third snapshot the walk and kick interleaving is shown. As shown the two right columns synchronized in the two last snapshots, the ball shooting snapshots.

The second page of the snapshots 6.8 shows the sideward kick. The snapshots again are separated in three columns, the two first represent the previous approach and the third column shows the walk and kick interleaving. Thus, in the first four snapshots of the left column, the robot walks, in the next four snapshots the robot tries to align with the ball. The next two snapshots show that the robot takes the appropriate posture for the kick and after those the robot starting the sideward kick. In the two first snapshots of the right column the robot's walk is shown and in the third snapshot the walk and kick interleaving is shown.

The last snapshots 6.9 represent the backward kick. As mentioned before the snapshots are separated in three columns. The two first columns represent the previous approach that the robot approaches the ball, stops, takes the appropriate pose and kicks the ball backward. The snapshots of the right column show that

the Nao robot approaches the ball and kicks backward when it is close enough, without stopping. So, in the first four snapshots of the left column, the robot walks, in the next three snapshots the robot tries to align with the ball. The next two snapshots show that the robot takes the appropriate posture for the kick and after those the robot starting the backward kick. In the two first snapshots of the right column the robot's walk is shown and in the third snapshot the walk and kick interleaving is shown.

6.3 User Friendliness

This implementation is user friendly. If the developer designs a new motion, to include it in the implementation for the motion skills interleaving, only the details in the xml file have to be changed. The name of the special action, the names of the joints that will be compared and the number of those joints have to be added. Also the user has to add the numbers of the poses that are possible for the transition and their threshold values, which can be calculated with the Matlab system. The code for reading the xml and the comparison algorithm, do not need any changes because they are customizable and general for all the special actions. Of course the part of the code that needs a change is the behavior, in which the user must give the name of the kick that will be used at the transition from the walk.

Figure 6.7: The snapshots for the forward kick

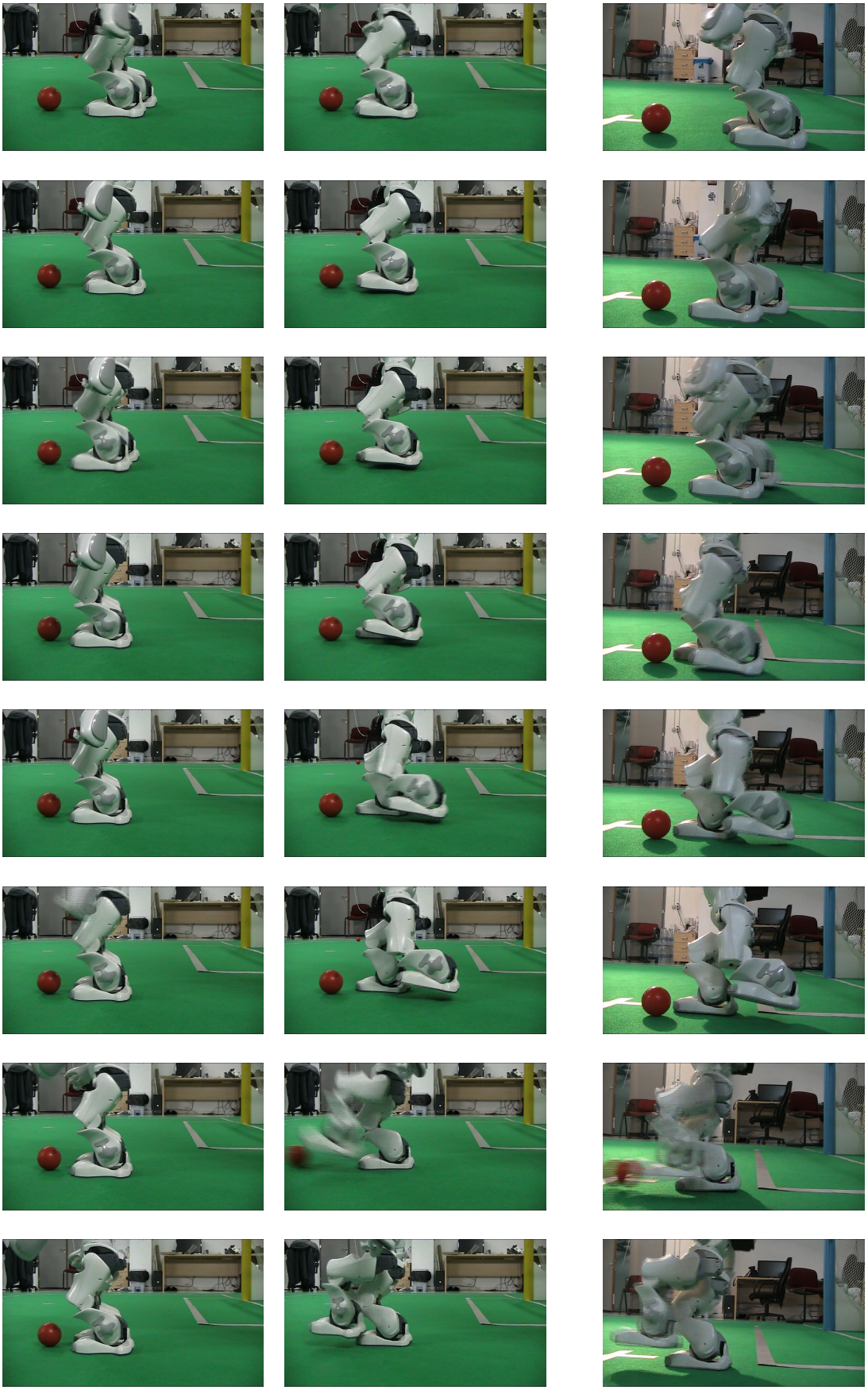


Figure 6.8: The snapshots for the sideward kick

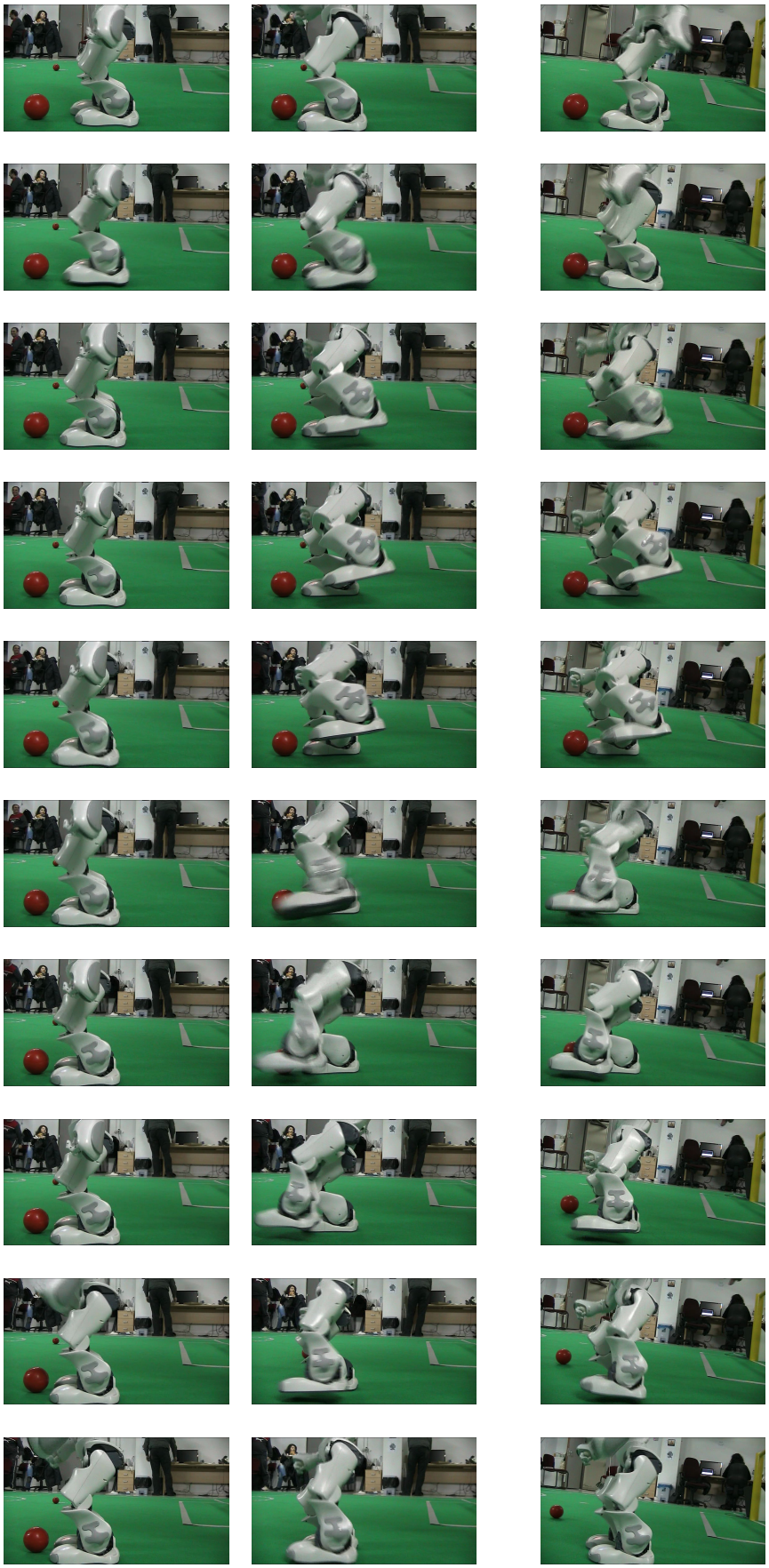
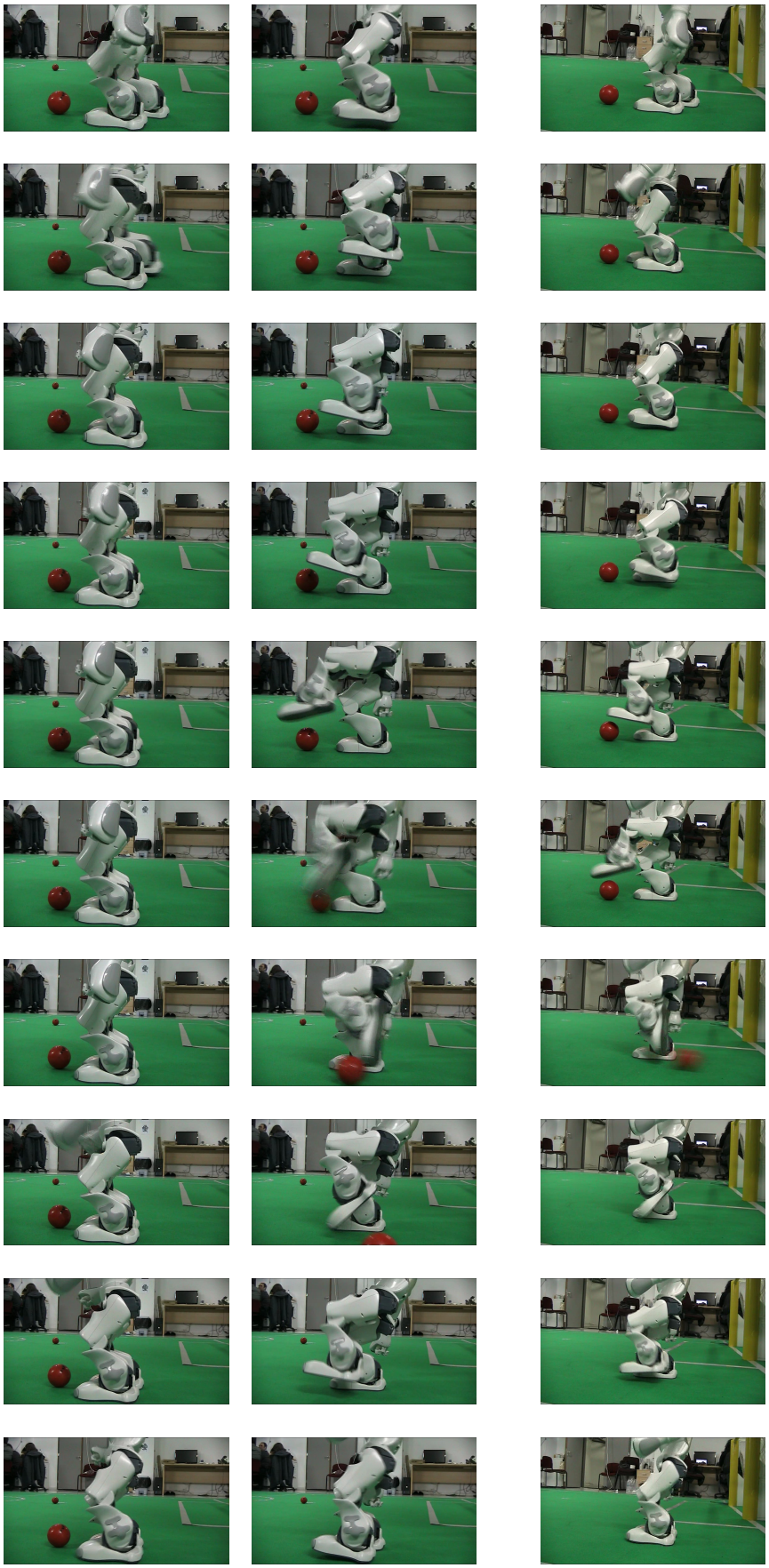


Figure 6.9: The snapshots for the backward kick



Chapter 7

Conclusion

To sum up our thesis, in this chapter the work of the SPL team B-Human will be discussed. Furthermore a future work on extending and improving our system will be presented.

7.1 Discussion

The Standard Platform League team B-Human designed the "In-Walk Kicks", a system similar of our walk and kick interleaving. B-Human have implemented a dynamic walk engine, which is in trajectory based. Furthermore their dynamic kicks (forward and sideward) are in trajectory based and they use a balancing system in the support leg of the kick. So, the basic work of that system, is the appropriate "merge" of those trajectories. On the other hand, in our implementation we used the dynamic walk engine of Aldebaran and the static kicks (forward, sideward and backward). We didn't have trajectories, neither we knew the trajectory of Aldebaran's dynamic walk. As the reader can conclude, those two implementations do not have any similarities, since each of them is based on different design. Thus, a comparison can not be done.

7.2 Future Work

Our work is the first step of a new methodology, designed for robots, in order to approach and kick the ball directly. Thus, there are plans for extending and

optimizing further the proposed idea of walk and kick interleaving.

The forward kick is a stable special action, as also is the sideward kick. Moreover the backward kick is an unstable kick because of the high lift of the leg of the kick motion. Those stable, or unstable, motions are the special actions that are used in the walk and kick interleaving process and have provided the same stabilization results. So, the unstable kicks need more "support" with a balancing system for the standing leg. This system will give more stability to the walk and kick interleaving by avoiding the possible falls of the robot.

Moreover this implementation is applicable to any kind of motion interleaving. The walk and kick interleaving is a first approach of motion interleaving. It is possible this approach to be extended to other special actions besides walk and kicks. For example, we can replace the walk with another special action and design the special actions interleaving, in order to design a robot choreography.

Same approach can be applied to other RoboCup leagues, such as the Humanoid League and the 3D Simulation League, which use humanoid robots. Furthermore same design is applicable to robots with many degrees of freedom in domains outside of the RoboCup.

References

[1] Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., Osawa, E., Matsubara, H.: Robocup: A challenge problem for AI. AI Magazine **18**(1) (1997) 73–85 14

[2] RoboCup Technical Committee: Standard Platform League rule book (2011) 15

[3] Gouaillier, D., Blazevic, P.: A mechatronic platform, the Aldebaran robotics humanoid robot. 32nd IEEE Annual Conference on Industrial Electronics, IECON 2006 (November 2006) 4049–4053 20

[4] Paraschos, A.: Monas: A flexible software architecture for robotic agents. Diploma thesis, Technical University of Crete, Greece (2010) 24

[5] Spanoudakis, N., Moraitis, P.: Using ASEME methodology for model-driven agent systems development. In: Intl Workshop on Agent-Oriented Software Engineering (AOSE). (2010) 25

[6] Paraschos, A., Spanoudakis, N., Lagoudakis, M.G.: Model-driven behavior specification for robotic teams. In: 11th International Conference on Autonomous Agents and Multiagent Systems(AAMAS 2012). (2012) 25

[7] Pierris, G.F., Lagoudakis, M.G.: An interactive tool for designing complex robot motion patterns. In: IEEE Intl Conf on Robotics and Automation (ICRA). (2009) 26

[8] Aldebaran Robotics: Nao software documentation (2011) Only available online: http://www.aldebaran-robotics.com/documentation/index.html. 30

[9] Röfer, T., Laue, T., Müller, J., Fabisch, A., Feldpausch, F., Gillmann, K., Graf, C., de Haas, T.J., Härtl, A., Humann, A., Honsel, D., Kastner, P., Kastner, T., Könemann, C., Markowsky, B., Riemann, O.J.L., Wenk, F.: B-human team report and code release 2011 (2011) Only available online: http://www.b-human.de/downloads/bhuman11_coderelease.pdf. 30