THE UNIVERSITY OF NEW SOUTH WALES SCHOOL OF

COMPUTER SCIENCE & ENGINEERING

From AIBO to Nao

The Transition from 4Legged to 2Legged Robot Soccer

David Collien (3160034) Gary Huynh (3159104)

B.Sc (Computer Science) $Hons.$

Supervisor: Maurice Pagnucco

Assessor: Claude Sammut

Submitted: August, 2008

Contents

Abstract

With a team and a code base that is well accustomed to the specific domain of Sony AIBO robotic soccer, moving to a new biped platform creates a vast set of new challenges.

This thesis explores many of these challenges and presents the theory and practical trials involved in developing and optimising locomotion behaviours for the new Robocup Standard Platform League. It gives an overview of research, practical approaches and insights to the development of various parts of a locomotion system.

As a result, the rUNSWift team were able to produce software for a biped robot that was capable of competing in a robotic soccer game.

Chapter 1

Introduction

1.1 Background

UNSW has been participating in the Robocup competition since 1999 and remains as one of the most successful universities to participate. rUNSWift (formerly known as UNSW United) has become world champions on three occasions (2000, 2001, 2003) within the Robocup 4-legged league with no other universities being able to meet this feat. rUNSWift has also been ranked second in the world three times (1999, 2002, 2006) and third once (2005). 2008 saw the Robocup 4-legged replaced with the Robocup standard league in which focus was shifted from quadrupedal robots to bipedal robots. Robocup 2008 was selected to be held in Suzhou, China during July 14th 20th, where rUNSWift competed in the Standard Platform League (SPL).

The current rules dictate that a game of robot soccer lasts 20 minutes with 10 minutes halves. A game consists of two teams of two "Nao" humanoid robots. The flat carpeted green field consist of an orange hollow plastic ball, two goals one blue and the other yellow and white field line markings. Robots must be able to 'see' features on the field with their cameras, use this information to localise their position and be able to track the position of the ball. Previously, coloured beacons were placed on the edges of the field to simplify the task of localisation. These beacons are no longer in the rules. Once the robot and ball's positions are known (or estimated as accurately as possible), the challenge lies in walking to the ball, and playing a strategic game of soccer.

1.2 Change of Platform

In the lead-up to the 2008 Robocup competition, a new standardised robot platform was required to replace the Sony AIBO, after Sony closed its robotics division and ceased production of the robotic dog. The platform that was chosen to be introduced in 2008 was the humanoid ‘Nao’ from French start-up robotics company Aldebaran. The Robocup edition of Nao, initially designed to be an entertainment robot, is a 58-cm high humanoid robot with 21 powered joints. To cater for a biped platform in the previously named ‘4-Legged League’, the league was renamed to the Standard Platform League (SPL). This differentiates this league, where the hardware platform is standardised, and hardware modification is prohibited, from other leagues in Robocup that allow custom hardware. The 2008 Standard Platform League included soccer games using both the quadruped AIBO dogs and the biped Nao and is (supposed to be) purely a software challenge.

1.3 Aims

In the transition from a 4-legged to biped robot, the main hurdles to overcome are encountered within the field of robot locomotion. There are also many changes in the structure of hardware, software and sensor configuration that need to be addressed. This thesis aims to explore areas of research new to this Robocup league and experiment with its application to the new hardware. The changes to the sensor and actuator configuration of the Nao robot has a significant impact on the locomotion of the robot, in comparison to the previous AIBO platform. This thesis provides applicable calculations, theory and experimentation results, focussing primarily on achieving effective locomotion.

The Nao Platform gives an overview of the configuration of both hardware and software on the new Nao platform and highlights aspects of the platform that are useful for the Robocup competition, how they have been used and how they might be potentially useful in the future.

Software Architecture describes how the software architecture has evolved from the previous team's code to cater for the differences and similarities between the two platforms. Issues with the current architecture are identified and a simplified and more manageable architecture is proposed.

Locomotion provides the theoretical foundation required to implement and understand biped locomotion and kinematics. It derives the forward kinematic equations specific to the Nao, introduces some methods of performing inverse kinematics and presents how to calculate the zero moment point from two different sources of information. It also covers strategies to actively stabilise a biped gait.

Implementation of the Locomotion Module explains the intended structure for the locomotion code and identifies the eventual implementation.

Locomotion Systems presents the basic theory behind the gaits that were successfully implemented on the robot or simulator. It outlines possible areas of future development within the locomotion module.

Building our own Gait Engine outlines our investigations into the implementation of a simple closed-loop gait engine.

Walk Optimisation details our experiments in finding the optimal parameters for creating a straight walk that is both competitively stable, accurate and fast.

Other Gait Optimisations extends our experimentation to further variations of gaits and presents a process to optimise such gaits.

Kicking gives a theoretical approach to kicking and investigates the effectiveness of kicking variations. The development of a generic parameterised kick is also presented.

Conclusion concludes the work and discusses future possibilities.

Chapter 2

The Nao Platform

In early 2008, Aldebaran provided Robocup teams with a virtual Nao robot in the Cyberbotics' Webots simulation environment. It was not until the end of April that the first Nao robot arrived. Made from prototyping plastic, it was clear that this robot was not a product that was ready for market. Within days it started to fall apart and within a week the Nao had already suffered decapitation and a leg amputation. After sending the faulty Nao back to Aldebaran in France, we were forced to make do with the Webots simulator.

Very few parallels exist between the physical Nao and its simulated counterpart, each even requiring a different API. Code from one environment was not easily applied to the other and furthermore, there was a very slim chance of the code behaving similarly.

In mid-June four more robots arrived. Two arrived faulty and the others soon suffered a similar fate to the previous Nao. In the weeks leading up to the competition a single robot was able to walk.

The major drawback of this year's competition, for all teams, was the lack of time available to test new code due to the unreliability and the fragility of the new platform. Whilst this hindered the progress of any in-depth experimentation, the new platform provides a new area of research and development not yet explored by this particular league: humanoid locomotion.

2.1 Webots

Webots is a generic robotics simulator by company Cyberbotics. In conjunction with Cyberbotics, Aldebaran created a virtual robot to use in the simulator.

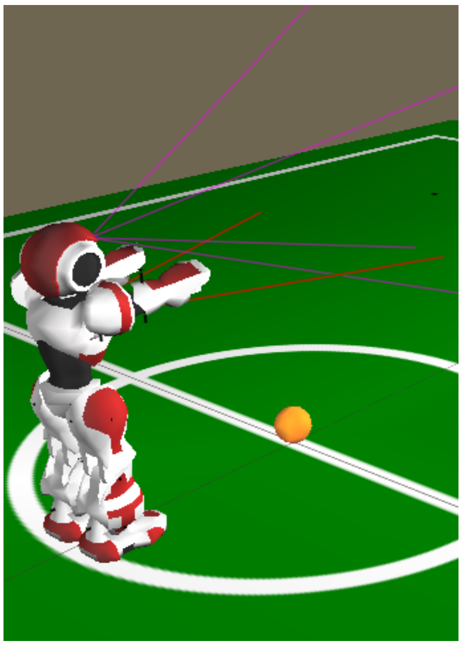

Figure 2.1: Webots Simulator Screenshot

Aldebaran provided Robocup teams with an initial set of libraries that controlled the Nao in the Webots simulator. This SDK did not provide much more than direct joint access and an interface to simulated sensor data. When the physical Nao was released, Aldebaran also released a new and rather different way of interfacing with the hardware, namely *NaoQi*.

2.2 NaoQi

> "NaoQi is the kernel of Nao's intelligence. It is run as soon as linux has completed initialisation. In a few seconds, it registers every device it can find in the robot, including motors (you should hear a quick noise on each joint), leds (blinks eyes and ears) and such..." [14]

The Nao runs a minimal Linux distribution, on which NaoQi runs as a start-up process. NaoQi offers two ways to interface code; by instructing NaoQi to load your dynamic libraries on start-up, or by writing your code to act as a network broker, allowing your modules to register and integrate with NaoQi's broker system over TCP/IP. NaoQi provides libraries for interfacing with sensors (through a shared memory object 'ALMemory'), motor control ('ALMotion'), text-to-speech, etc. NaoQi does not deliver camera frames in an efficient manner, but direct access to the camera driver is available as a Linux block device file.

Ruby scripting support is also offered in NaoQi. Ruby libraries are provided that allow Ruby to connect as a broker and use methods of any library or broker that has registered its interface with NaoQi.

Whilst connecting to NaoQi as a broker is useful for carrying out tests on the Nao, the broker requests over TCP/IP can cause significant overhead. Compiling code as a dynamic library is the fastest method to communicate with NaoQi. NaoQi can be told to load this library on start-up by editing a configuration file (modules/lib/autoload.ini).

NaoQi is slow and consumes a lot of system resources. In tests run when the Nao is idle, at least 15% of the CPU is being used by NaoQi alone. Each joint on the Nao occupies its own thread, and each subscription to NaoQi's memory device, ALMemory, runs in one of some 50 idle threads created on start-up.

2.2.1 Conclusion

NaoQi clearly does not lend itself well to the Robocup competition where efficiency is paramount (every last CPU cycle is valuable to cram in better vision processing). Developing rUNSWift code independently to NaoQi gives us other competitive advantages, such as not relying on the correctness of Aldebaran code and being held back by bugs outside the team's control.

2.3 Nao Hardware

2.3.1 Joints

The Nao has 21 magnetically encoded motors. Twenty joints are individually powered, and two mechanically-linked diagonal hip joints are powered by a single motor. This joint configuration forms five kinematic chains extending from the torso: the head chain, two arm chains and two leg chains.

The head has two degrees of freedom; pitch and yaw (primarily useful for camera positioning). Each arm contains two degrees of freedom at both the hip and elbow joints. The leg chain configuration consists of:

- Two degrees of freedom, pitch and roll, at each of the ankle and hip joints
- A single degree of freedom at each knee joint
- A single groin motor that simultaneously manipulates both leg chains at the hip, acting on both the yaw and pitch axes.

The Nao is unable to independently control each leg chain's yaw as the upper hip joint is mechanically linked to change both legs simultaneously. This is the only joint which affects the yaw in the leg chains, however the diagonal orientation of the joint, relative to the torso, causes this joint to act upon both the yaw and pitch. In order to change only the yaw of a leg, the first pitch joint in the chain needs to compensate for the pitch component of the upper hip joint. The linked yaw-pitch design delivers a more compact structure and saves on a motor, but the robot is limited to symmetrical yaw rotations of the legs, relative to the torso.

2.3.2 FSR

Four Force Sensitive Resistors (FSRs) sit under the foot plate on each of the Nao's feet (see Figure 2.2). Each FSR gives a reading relative to how much force is 'squeezing' them. The purpose behind placing FSRs on the feet is to measure the force distribution over the foot surface as it makes contact with the ground. This is useful to measure the moment of inertia.

During a walking motion, the robot experiences both acceleration due to gravity and the acceleration generated by the walking motion. The resulting forces are counterbalanced by the reactive force from the ground surface at the point of contact. When the robot is supported by a single foot, the FSRs attempt to measure the force at four points on this contact surface. Foot force readings are useful to determine the stability of the robot and to plan the desired joint trajectories of the robot in order to maintain stability during locomotion tasks. A common stability criterion and tool in walk planning is the Zero Moment Point (ZMP) - the point at which the reaction force of the ground on the foot does not produce any moment. With a suitable FSR arrangement, this can be calculated directly from the foot force readings (as shown in chapter 4.5.5) [4]. Other research focusses on deriving a direct relation between the reactive force and joint motion [6].

Unfortunately the force readings given from the Nao's FSRs are not independent of one another. The FSRs are mounted beneath the foot plate in such a fashion that the readings from a FSR sensor do not correspond to the force on a single point on the sole of the foot. A force on almost any single point on the foot can influence multiple FSR sensors. It is therefore very difficult to make sense of the data that they return, without a theoretical model of the exact mechanics of the FSR mounting method.

Another approach, as proposed by Steven Wong of the 2008 Robocup team, is to at-

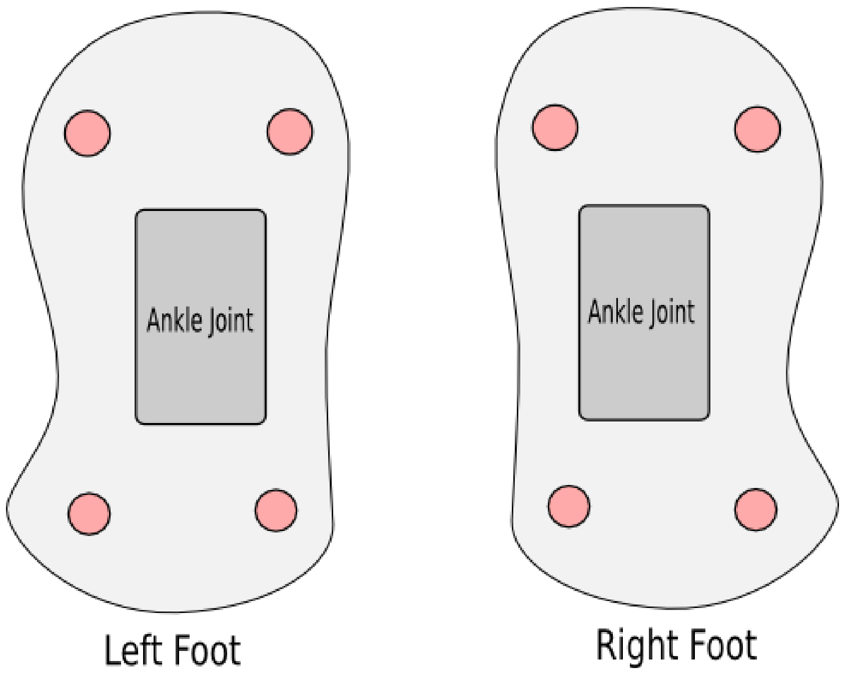

Figure 2.2: Location of FSRs beneath Nao’s foot plate

tempt to machine-learn the foot-force sensor conditions under which the system is stable, by means of a decision tree:

> “[This approach requires the recording of] a lot of readings and ’going to fall over’ pairs, and would let a decision tree work out what the boundary is between falling over and not falling over, as well as the probability. This is essentially a classification problem because you’re trying to match a value to a condition. The decision tree will also handle noise well because on every split, you’re splitting on the value that minimises the entropy...” Steven Wong, in email discussion

This is essentially determining if the ZMP lies inside or outside of the supporting area of the foot. The ability to calculate the actual coordinates of the ZMP is more beneficial to the closed-loop walking problem, in particular the ability to sense the ZMP trajectory during a walking pattern. The sensed trajectory of the ZMP can be mapped to the expected ZMP trajectory of the walk (or other planned motion) to look for early warning signs of instability as well as help with planning to place the next foot (See chapter 4.5.7 on Walk

Profiling).

Other biped robot designs are able to sense the ZMP accurately, as they ensure that the four FSRs are the only contact points with the ground [3], or have a force sensitive surface covering the sole [4].

2.3.3 IMU

A 2-axis gyroscope/gyrometer, and a 3-axis accelerometer are mounted in the lower torso of the Nao. These together form the Inertial Measurement Unit (IMU). The gyroscope measures rotational velocity around the X and Y axis in Figure 2.3. The accelerometer measures linear acceleration along all three axes.

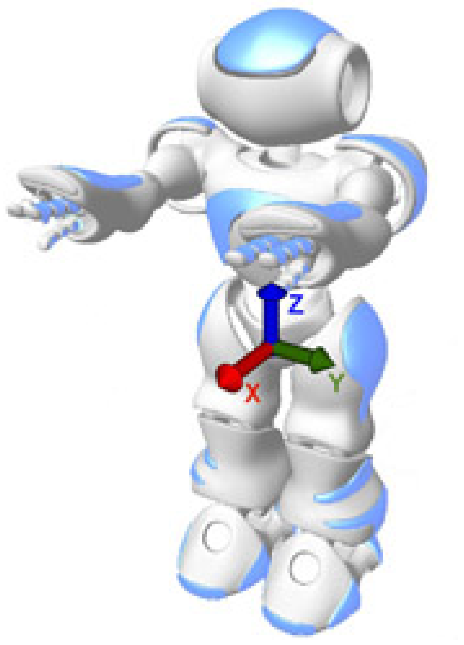

Figure 2.3: Axes of Nao's IMU

Whilst accurate in short time periods, most gyroscope systems have a tendency to drift and build up a bias over longer time periods. The values from the IDG-300 gyroscope in the Nao require frequent re-calibration from the accelerometer as significant drift is apparent over several seconds.

The gyroscope bias, which is the output when the angular velocity is zero, will drift over time due to factors such as temperature and stress. Such drift is present to some degree in all gyroscope systems, and must be addressed in the application... Accelerometers can be used to measure tilt, and provide enough of a signal to calibrate gyroscope biases over long periods of time... Other systems may use infrared, magnetic, or vision based sensing to calibrate the gyroscope bias. [9]

Using the accelerometer to re-calibrate the gyroscope and integrating the angular velocity reported by the gyroscope over time, the orientation of the torso relative to the ground can be determined. The angular orientation and velocity used in conjunction with a kinematic model of the robot can be used as another stability criterion for locomotion as well as detecting if the robot is in a 'fallen over' state.

IMUs have also been used with success to dead-reckon the position and orientation of mobile robots. The gyroscope provides the orientation of the robot by integrating the sensed angular velocities over time. The accelerometer measures the linear acceleration of the system in the inertial reference frame, but as the accelerometer is fixed to the system the acceleration measured is relative to the orientation of the system. Integrating gives the linear velocity relative to the orientation of the system. The orientation obtained from the gyro can then be used to convert these velocities into an absolute reference frame, and integrated again to give the dead-reckoned position [15]. This dead-reckoned position can then be incorporated into the robot's Baysian localisation.

The gyroscope in the Nao is limited to sensing in two axes, and does not record the velocity in the transverse (Z, in Figure 2.3) axis. This means that any change of orientation on the 2-dimensional coordinate space of the playing field cannot be tracked by the IMU. This restricts the information available to linear movements only.

2.3.4 Ultrasonic

The Nao is equipped with a left and a right ultrasonic distance sensor (each consisting of an emitter and a receiver) on the front of the chest.

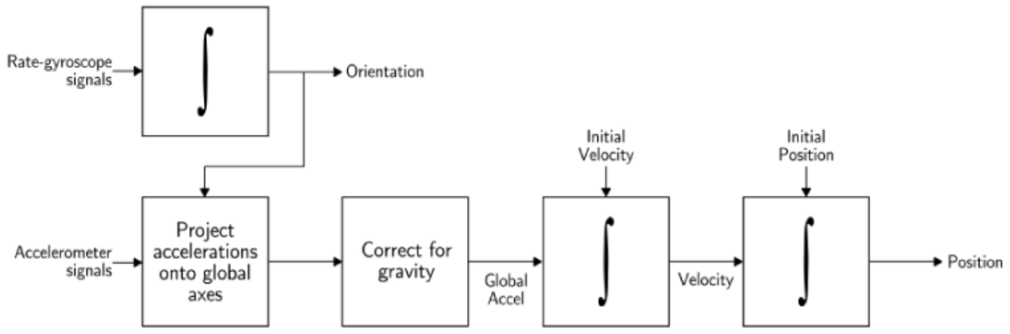

Figure 2.4: The process of dead-reckoning from an IMU.

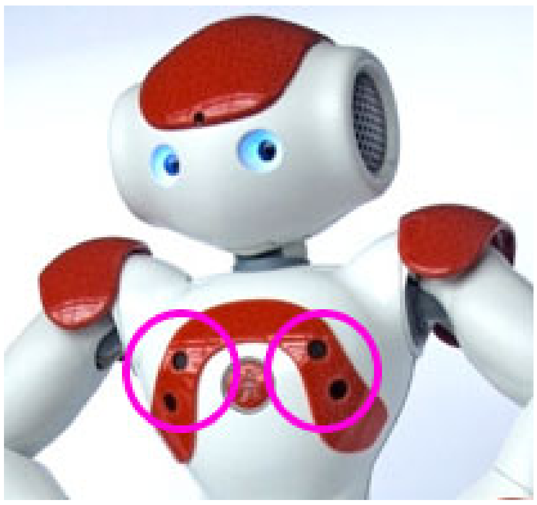

Figure 2.5: The location of the left and right ultrasonic sensors on Nao.

The transmitter emits an ultrasonic ping and the receiver awaits the echo returned. The time difference is proportional to the distance of the detected object. The ultrasonic sensors on the Nao have a detectable range of 0cm to 70cm, but can only return a distance measurement between 15cm and 70cm. The minimum time taken to acquire a distance reading is 240 milliseconds, although Aldebaran recommend specifying a 500 millisecond acquisition period [14]. These sensors are potentially useful for obstacle detection and avoidance on the field.

2.3.5 Buttons and Bumpers

The chest button (power on button) on the Nao has the following default functionality:

- Single press: Speaks ID number and IP address
- Double press: Stops all motors and removes motor stiffness
- Triple press: Shutdown
- Held for over 4 seconds: Force power off

Single and double press events each set a boolean value to True in ALMemory , NaoQi's shared memory object (at keys ALWatchDog/SimpleClickOccured and ALWatchDog/-DoubleClickOccured). This value can be read and reset by code linked to NaoQi so custom functionality for single and double presses of the chest button can be coded.

The default behaviour for a single and double press can be disabled by connecting as a broker to the undocumented ALWatchdog Proxy in NaoQi. Note: this does not work if defined in code that is linking to NaoQi as a dynamic library, due to a bug in NaoQi (attempting to run this code when linked as a dynamic library will cause NaoQi to throw an error and die.)

e.g. using a Ruby broker:

```
watchdogProxy = Aldebaran::ALProxy.new({}``ALWatchDog'', \$IP, \$Port)
watchdogProxy.enableDefaultActionSimpleClick( false );
watchdogProxy.enableDefaultActionDoubleClick( false );
```

```
Alexandre, Aldebaran Robocup Forums
```

There are bumper-buttons over the front of Nao's feet. These buttons are not sensitive enough to detect contact with the ball in most situations, but can be used as general purpose buttons for setting states, etc. These are accessible through the DCM Proxy provided with NaoQi (refer to DCM documentation).

2.3.6 Camera

The camera on the Nao has a maximum resolution of 640x480 and a maximum frame-rate of 30 frames per second, similar to the camera on the AIBO ERS-7. In the Robocup competition, the camera is the primary sensor used for detecting and tracking objects and features present on the field. Unlike the ERS-7 camera, the Nao does not suffer from ring distortion and does not require 'ring correction' present in the previous years' code.

A camera frame is directly accessible from a Linux device file descriptor using a read operation. The camera's video4linux drivers do not yet support memory mapping and are therefore suboptimal at delivering frames. The output bytes are in YUV422 format with a byte ordering of U, Y1, V, Y2, however a bug in the device, or its drivers, sometimes shifts this to V, Y2, U, Y3 ordering, thus flipping the U and V in the resulting image.

Detecting this flip from the image was a difficult task. A flip in U and V does not affect the colours green or white (e.g. the field, or field lines), turns yellow to blue, and blue to yellow (thus the goals are flipped, but not detectable as being erroneous) and the ball changes from orange to a violet-blue. The only element on the field that can therefore be used to determine if this error has occurred is the ball, which is rarely seen. Colours cannot be re-calibrated for both scenarios, as the yellow goal when flipped in U and V is almost identical in colour to the blue goal, and vice versa.

After capturing sample camera frames, another bug in the camera frame was noticeable: the pixels in the second-last row of each frame were tinted blue if the camera was operating “correctly”. If these pixels had a low enough contrast, the presence of a blue tint could be used to determine if the U and V in the image had flipped and the problem could be automatically corrected. It should also be noted that the last row of pixels in the image does not represent the visual frame whatsoever, and alternates between green or black (for 1 frame) and magenta (for 4 frames).

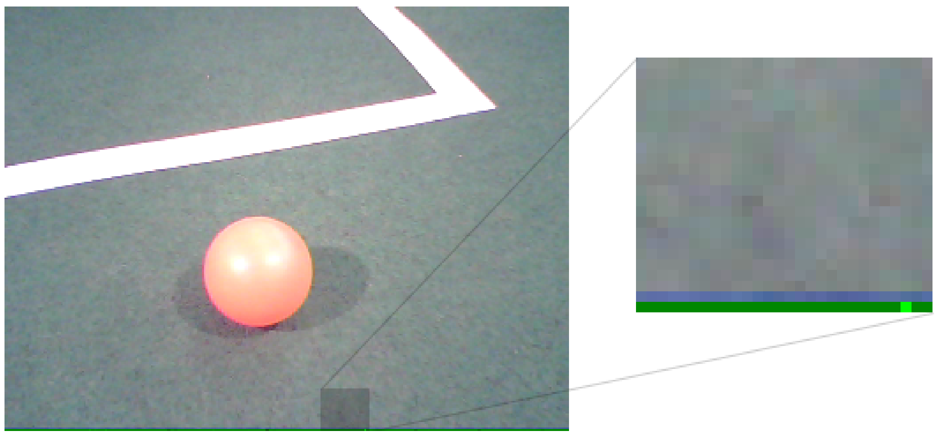
Figure 2.6: Close-up of the last two rows of pixels in a camera frame.

Aldebaran were unable to provide a patch for this colour-flipping issue before the competition, as it required a firmware upgrade of a proprietary Cypress PSOC board (used for camera control) that had its own proprietary protocol for in-system programming. When this patch was applied to robots at the competition, it was found to cause more problems than it solved and those patched robots were flashed back to the previous buggy version.

In comparison with the AIBO league, the Nao games are very slow moving. Running at a lower frame-rate and skipping frames (containing vital localisation information) is therefore not as much of a problem as it was with AIBO soccer. There also exists conjecture that running the vision (or any other processor intensive) thread with a higher priority than NaoQi’s motion object can cause locomotion problems.

This camera’s drivers also offer a lower output resolution of 320x240. Due to the processing capabilities of the Nao (slower than that of the ERS-7, even before the overhead of NaoQi), this lower resolution was used in the competition. A higher resolution image provides better long-distance vision, and there is future possibility for changing the camera mode intelligently to capture short-distance and long-distance frames in the correct circumstances.

2.4 Conclusion

There are issues with both the Nao's hardware and Software. The hardware issues are outside our control, but some workarounds are possible. Our general aim should be to make the rUNSWift code as independent from NaoQi as possible, so that as much of the system is inside the team's zone of control.

The major issues with Nao hardware and software are:

- The Nao is fragile
- NaoQi is slow, takes up resources and the majority of its functionality is unnecessary for Robocup
- Foot-force sensors do not accurately represent the force on any individual point on the foot's surface, making calculations and filtering difficult
- The gyro does not measure in the Z axis, limiting the ability to dead-reckon orientation on the field
- The processor is a downgrade from the ERS-7
- The chest button cannot be disabled from code linked to NaoQi as a shared object (it will crash), only linked as a broker
- The camera interfacing circuitry is proprietary and difficult for teams to upgrade
- The camera drivers do not allow fast access to the image (memory mapping)
- The camera suffers from a byte-ordering bug that performs an annoying flip of the colour space

The Nao hardware and software does have beneficial aspects:

- NaoQi's broker service allows easy testing of simple code that can communicate to the robot whilst running on another machine
- The Nao has a rich set of sensors that appear to be very useful (even if a few aren't)
- NaoQi provides a comprehensive set of sophisticated locomotion tools, useful as reference and to kick-start the new league

Chapter 3

Software Architecture

The 2008 code made significant changes to the architecture of the rUNSWift code in order to incorporate the NaoQi interface. This chapter shows the shift in architecture and proposes a new architecture for the future.

3.1 Previous Architecture

Figure 3.1 shows a simplified software architecture from 2006. The main bulk of the code resides in the C++ vision and localisation (formerly known as the ambiguously named 'GPS') modules. The decision and behaviour code was scripted and run in an extended and embedded Python interpreter.

3.2 2008 Architecture

Changing to a completely new platform has resulted in some dramatic changes to the architecture of the rUNSWift code. The 2006 AIBO code was used as the basis for the 2008 vision and localisation objects, with changes necessary for a different camera resolution and type, differences in field and object size and other rule changes. Completely new code was needed for locomotion and interfacing with NaoQi and Linux (as the AIBO ran on top of the Sony Open-R operating system).

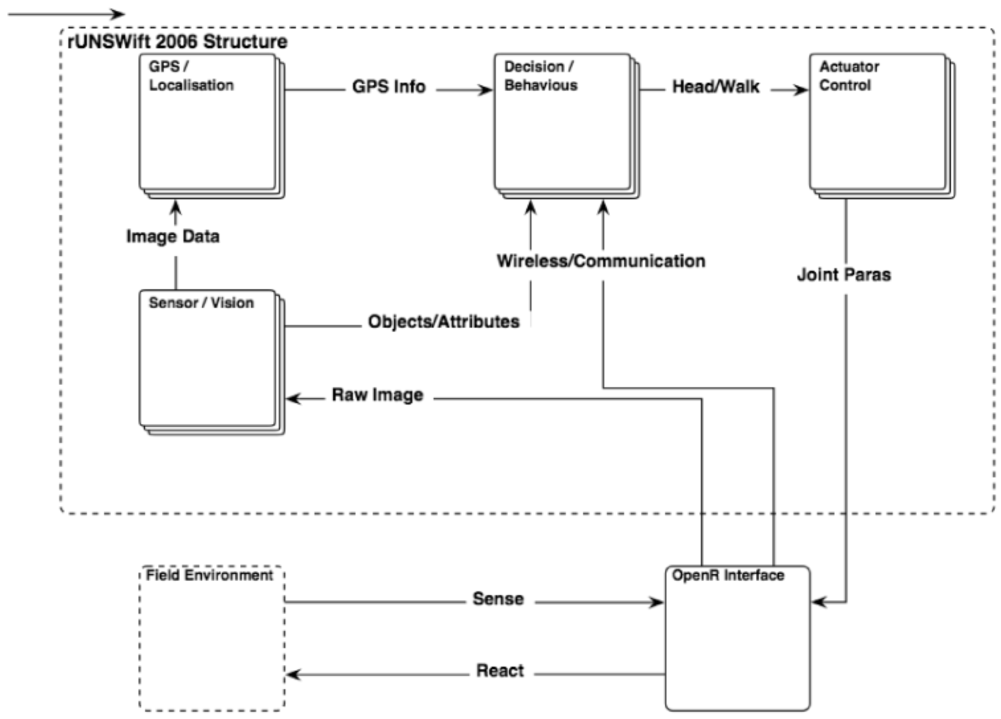

Figure 3.1: 2006 AIBO Architecture. [8]

There are three main areas of code in the new architecture:

- Modified 2006 vision and localisation code
- New locomotion, joint control and sensor code
- A simple behaviour layer

Due to delays in receiving the robots and the unreliability and fragility of the hardware when it was eventually received, the aim for 2008 was to limited to demonstrating a functioning vision and localisation, a stable walk and a simple ball-finding behaviour. Initially the behaviour code was scripted in Ruby, using NaoQi's Ruby-as-a-broker support. Connecting to NaoQi with a Ruby broker was found to be unreliable, with communication errors occurring in the NaoQi broker system causing the behaviour process to crash.

The behaviour code was re-written in C++ and compiled with the other modules into a dynamic library that linked with NaoQi at runtime. This also provided a significant increase in both the code and NaoQi's performance, previously slowed by broker communication through the SOAP protocol and the Ruby process consuming system resources.

The final architecture that was patched together for the competition is illustrated in Figure 3.2. Note: Networking, ultrasonic sensors, the IMU and the FSR sensors were never used during the 2008 competition.

This architecture was implemented with singleton classes for each object. Each object provided its own set of interface functions for receiving or sending the required data or events.

3.3 Proposed Future Architecture

In an attempt to simplify the architecture for future years, the following architecture in Figure 3.3 is proposed.

This architecture simplifies the previous layered structure by using a central 'blackboard' as shared memory between modules. Each thread (solid square in Figure 3.3) can read from and write to the blackboard, which holds the latest information available about the world model and state of the robot. This avoids the need for each module requiring its own public interface and simplifies communication between modules.

Threads can be prioritised, so that information that is more critical to the performance of the robot in a competition is updated more frequently. e.g. the Vision and Localisation thread would run with a higher priority than Behaviour.

For example, the camera frame would be saved onto the blackboard and then processed by the Vision and then Localisation objects, which returns the updated world model to the blackboard. Behaviour can read the latest world model and decide what actions to perform next - stored to the blackboard in a queue of walking/locomotion actions, a queue of head-tracking/other joint actions or a control signal to interrupt the current action. Locomotion can take the next action to perform from the blackboard, along with the latest sensor information to calculate and execute joint trajectories.

Another advantage of this architecture is the ability to send any part, or all of the

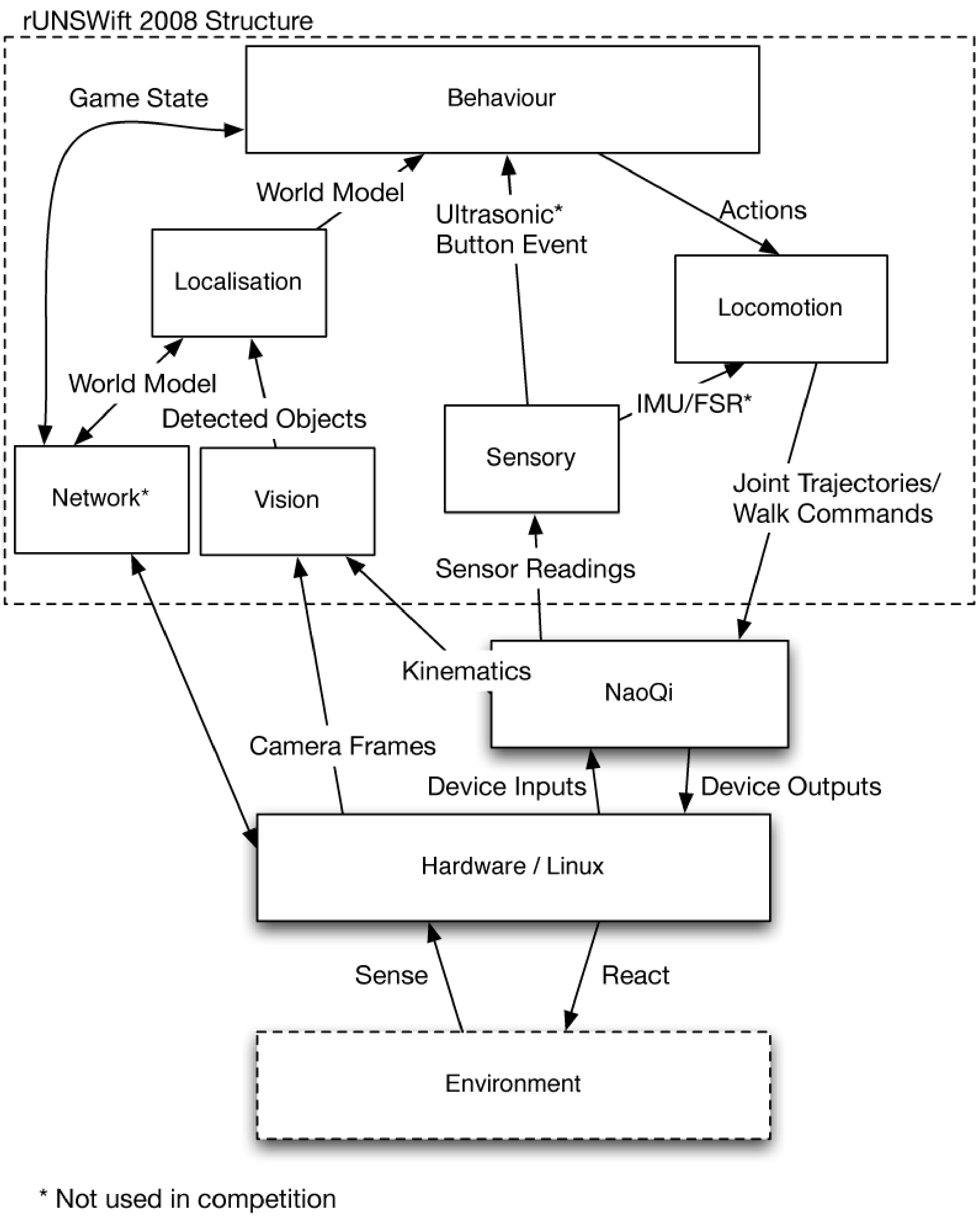

Figure 3.2: 2008 Nao Architecture

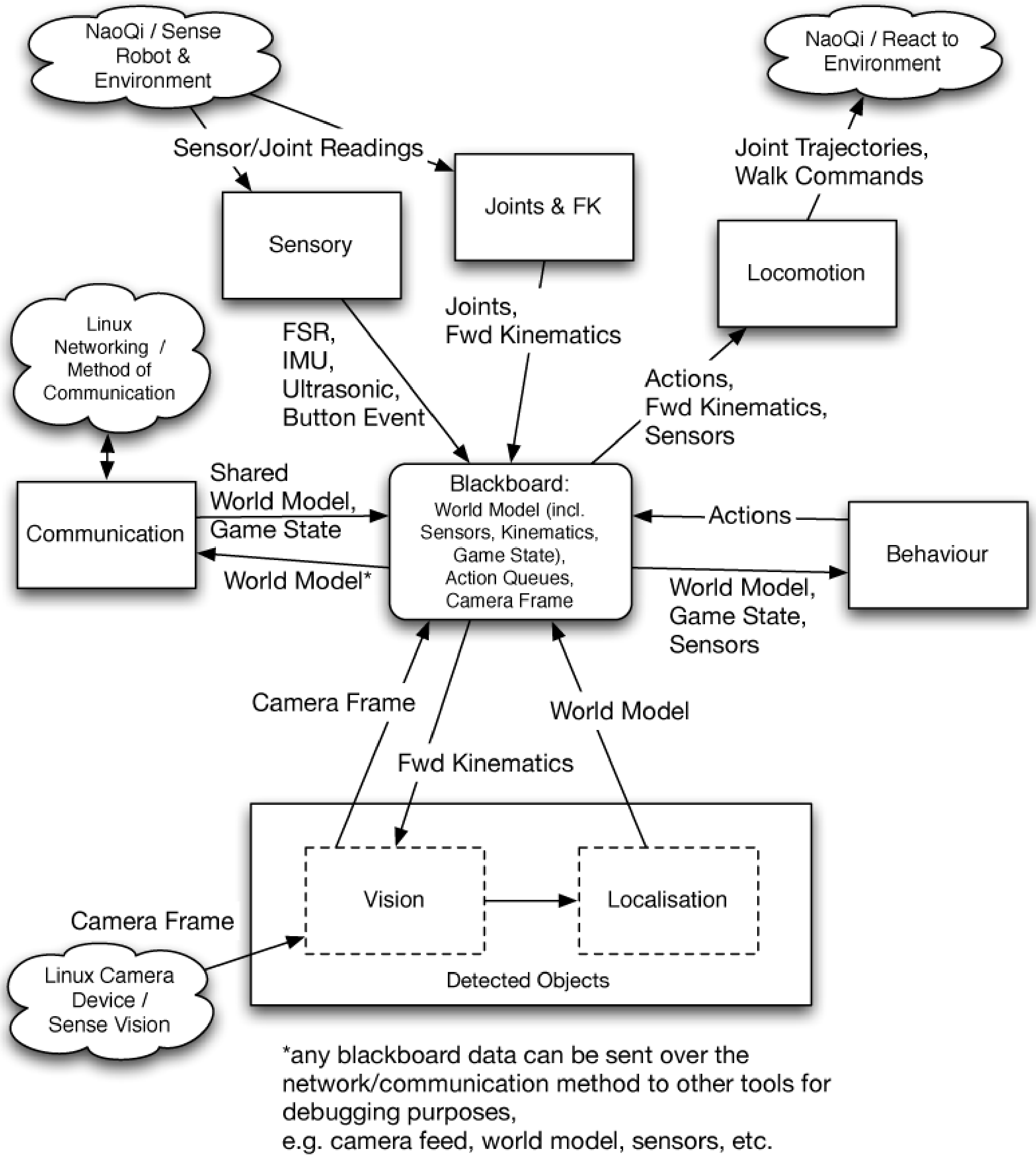
NaoQi / Sense Robot & Environment
Sensor/Joint Readings
Sensory
Joints & FK
NaoQi / React to Environment
Joint Trajectories, Walk Commands
Locomotion
Linux Networking / Method of Communication
FSR, IMU, Ultrasonic, Button Event
Joints, Fwd Kinematics
Actions, Fwd Kinematics, Sensors
Communication
Shared World Model, Game State
World Model*
Blackboard: World Model (incl. Sensors, Kinematics, Game State), Action Queues, Camera Frame
Actions
Behaviour
World Model, Game State, Sensors
Camera Frame
Fwd Kinematics
World Model
Vision
Localisation
Detected Objects
Camera Frame
Linux Camera Device / Sense Vision
*any blackboard data can be sent over the network/communication method to other tools for debugging purposes, e.g. camera feed, world model, sensors, etc.

Figure 3.3: Proposed Architecture

blackboard (world model and robot state) over the network to other robots or to debugging applications.

3.4 Conclusion

This year the architecture was cobbled together from old AIBO code for vision and localisation, and new code for the new platform's sensor configuration and locomotion. The result was a complicated and hard to maintain structure, which created a need to reconceptualise the architecture and create a simpler and more manageable design.

Chapter 4

Locomotion

4.1 Forward and Inverse Kinematics

Forward kinematics is the computation of the position and orientation of the end-point (end effector) of a chain of joints (kinematic chain). The Nao has five kinematic chains:

- The neck chain, with the head as the end-effector (as the camera is the primary point to control on the head, it would be useful to use the camera's position and orientation as the end-effector)
- The leg chains, with the feet as end-effectors
- The arm chains, with fingers as end-effectors

Inverse kinematics is the reverse process, computing the required joint trajectories to move an end-effector the required position. The problem of inverse kinematics is non-trivial as there are often multiple solutions to any given position.

4.2 NaoQi Kinematics

NaoQi provides a module 'ALMotion' which has limited methods that do both forward and inverse kinematic calculations. All methods that affect end-effectors have an option to be calculated relative to two different co-ordinate spaces, either the torso or the supporting foot.

The following methods affect the end-effectors of chains:

- getForwardTransform - given the name of a chain it provides the homogeneous transformation matrix for a complete transformation up the length of a chain in its current configuration.
- getPosition - returns the position of an end-effector on a chain. gotoPosition - attempts to move an end-effector to the desired position.
- changePosition - changes the end-effector position by a given vector.

The set, get and changeCom methods affect the centre of mass of the robot. The position vectors used are relative to the supporting foot.

As outlined in chapter 2.2, rUNSWift code can not reply upon NaoQi, therefore it is necessary to implement our own forward and inverse kinematics methods.

4.3 Forward Kinematics Calculation

An efficient, and standard method of calculating the forward kinematics is the Denavit-Hartenberg (D-H) convention. The D-H convention is used to parameterise each the links in a chain into a set of two rotations and two translations. Each rotation and translation is multiplied to give the 4 x 4 homogeneous transformation matrix for each link. Multiplying each of these 'link matrices' produces the end-to-end transformation matrix for the chain. From this final matrix the position and orientation of the end-effector can be obtained

4.3.1 The Denavit-Hartenberg Convention

The diagram below shows the four parameters that define the i'th link in a chain:

- The link length, a_i

- The link twist, α_i
- The link offset, d_i
- The link rotation, ϕ_i

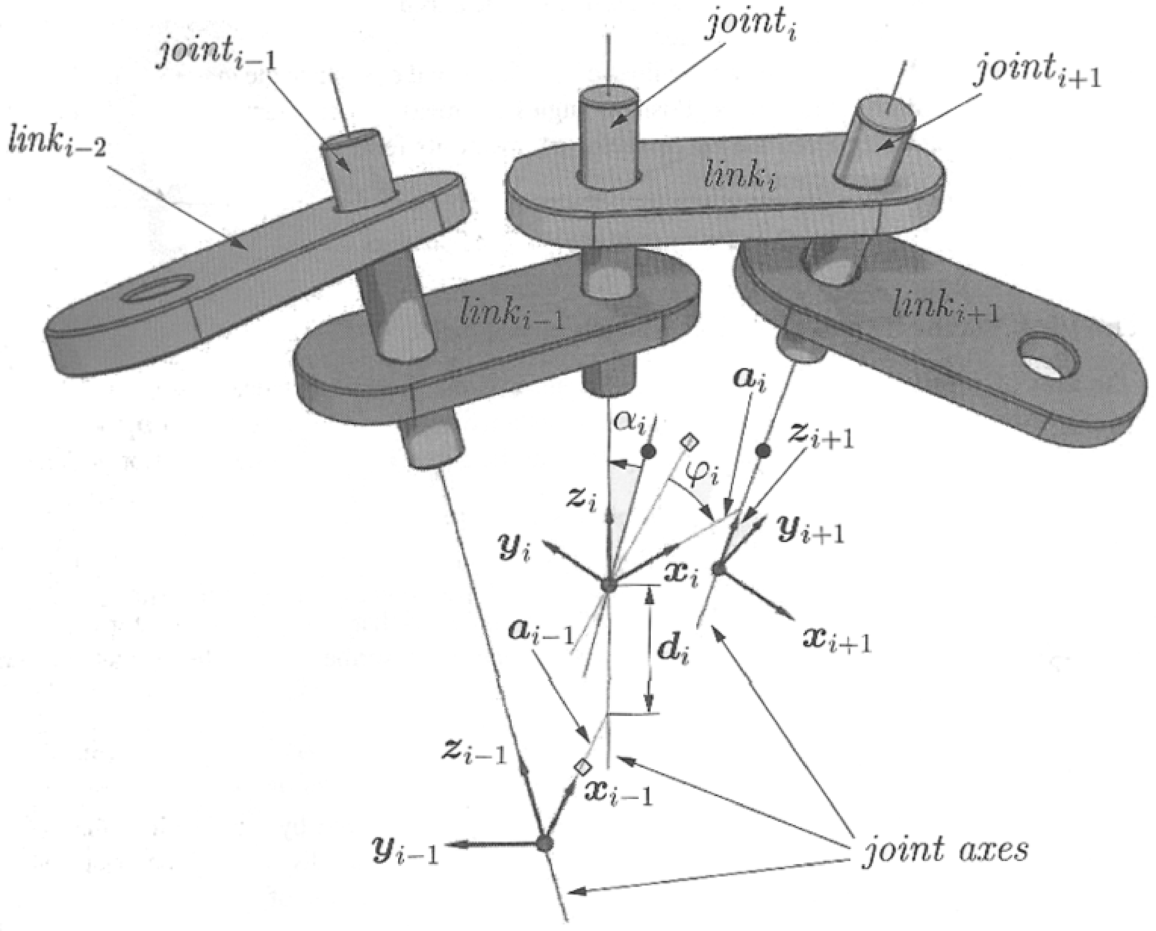

Figure 4.1: The Denavit-Hartenberg Parameterisation. Alfred Bruckstein, Yaniv Altshuler [1]

The Nao documentation outlines D-H parameter values for each of the five kinematic chains, including end transformations from the torso to the starting link of each chain, and to the end-effector position (as defined by Aldebaran) from the last link. These parameters are for the robot in its 'zero stance'.

As every joint in the Nao is rotational, only the rotation parameter (ϕ_i, around the z_i axis in Figure 4.1) needs to be modified to include the current state of the joint. The

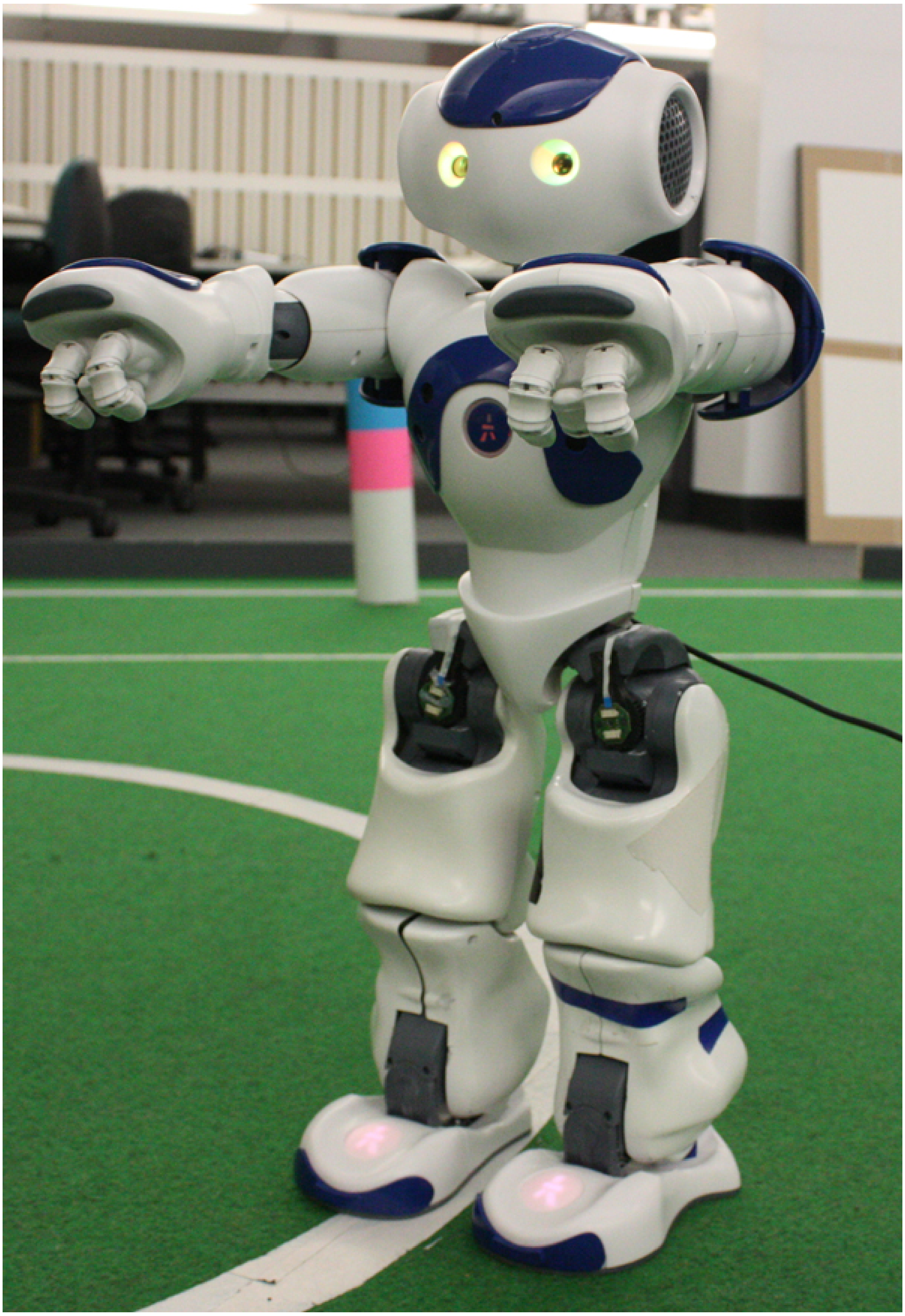

Figure 4.2: The ’zero stance’.

ϕ_i parameter therefore becomes the sum of the originalϕ_{zero_i} parameter from the ‘zero stance’ and the joint’s reported angle value (θ_i).

¿From each D-H parameter of each joint a rotation or translation 3D transformation matrix can be created:

$$Rot_{x_i} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\alpha_i & -sin\alpha_i & 0 \\ 0 & sin\alpha_i & cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Trans_{x_i} = \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Trans_{z_i} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Rot_{z_i} = \begin{bmatrix} cos\phi_i & -sin\phi_i & 0 & 0 \\ sin\phi_i & cos\phi_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The transformation for a single link is the multiplication of these matrices:

$$T_i = Rot_{z_i} Trans_{z_i} Trans_{x_i} Rot_{x_i}$$

Resulting in the following matrix for this link[1]:

$$T_i = \begin{bmatrix} cos\phi_i & -sin\phi_i cos\alpha_i & sin\phi_i sin\alpha_i & a_i cos\phi_i \\ sin\phi_i & cos\phi_i cos\alpha_i & -cos\phi_i sin\alpha_i & a_i sin\phi_i \\ 0 & sin\alpha_i & cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The end-to-end transformation for each chain is the matrix multiplication of each link's matrix.

In the joint configuration of the Nao, ϕ is the only variable in each link's matrix. Every other coefficient can be pre-computed and a constant-time function can be generated to create the end-to-end transformation matrix for a chain, in terms of ϕ_i (for each link i in the chain). Where $\phi_i = \phi_{zero_i} + \theta_i$.

The resulting end-to-end transform of a chain is a homogeneous 4 x 4 matrix of the form:

$$\begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The position and orientation of the end-effector can be extracted from this matrix in the following manner[1]:

$$x = m_{14}$$

$$y = m_{24}$$

$$z = m_{34}$$

$$yaw = arctan(m_{32}/m_{33})$$

$$pitch = arctan(-m_{31}/(m_{32}sin(yaw) + m_{33}cos(yaw)))$$

$$roll = arctan(m_{21}/m_{11})$$

4.3.2 Centre of Mass Calculation

The forward kinematics computation can also be used to determine the position of any intermediate link along the chain - by treating each intermediate link as an end-effector and deriving the appropriate function for each link using the method above (Note: the orientation is not required). Given the mass of each link, the centre of mass of the robot can be calculated by averaging the link positions $\mathbf{r}_i$, weighted by their masses m_i [7]:

$$\textbf{CoM} = \frac{\sum m_i \mathbf{r}_i}{\sum m_i}$$

Note: an additional end-transform may need to be added to the partial chains to ensure that the resulting end-to-end transform sits at the centre of mass of the individual link. As Aldebaran documentation does not provide the precise centre of mass for each link, it is assumed that the joint locations on each link (where the motor and gearing lies) are the heaviest parts, and this position is a good enough approximation of each link's centre of mass. Another alternative is to completely disassemble the Nao to determine the precise centre of mass of each link. However, given the non-modular design of the Nao, this would mostly likely render the robot useless.

4.3.3 Camera Height and Translation of the Ground Plane

The Vision system requires two important forward kinematic calculations:

- Finding the height of the camera off the ground, for ball point-projection
- Finding the orientation of the camera, relative to the ground plane, to calculate the horizon in an image

For the purposes of kinematic calculations, the ground plane can be assumed to be in contact with the surface of the supporting foot. The ground plane can be defined by the normal to the plane. Finding the height of the camera is therefore a combination of the end-to-end transform of the supporting leg chain, and the head chain (assuming the end-effector of the head chain has been defined as the camera position). To find the orientation of the head relative to the ground plane, the ground plane normal can be translated to the head's coordinate system using the same end-to-end transforms of the head and supporting leg chains.

The ground plane normal in its own coordinate space is defined as the vector g $= (0, 0, 1)$ which is a unit vector pointing straight up.

The first step is to perform a transform of the supporting foot chain from the torso. This gives the position and orientation of the supporting foot relative to the torso:

$$x_{support}$$

$$y_{support}$$

$$z_{support}$$

$$pitch_{support}$$

$$roll_{support}$$

$$yaw_{support}$$

The ground plane normal is then rotated first by $pitch_{support}$, then $roll_{support}$, then $yaw_{support}$. This gives us the vector $\mathbf{t}$, the ground plane in the torso's coordinate space.

The next step is to perform a transform of the head chain from the torso, giving the position and orientation of the head (camera) relative to the torso:

$$x_{head}$$

$$y_{head}$$

$$z_{head}$$

$$pitch_{head}$$

$$roll_{head}$$

$$yaw_{head}$$

Rotating the previous vector $\mathbf{t}$ by$pitch_{head}$, then $roll_{head}$, then yaw_{head} gives the vector$\mathbf{h}$, the ground plane in the head's coordinate space. The pitch, roll and yaw angles between$\mathbf{h}$ and the original ground plane gare the $pitch$, $roll$ and yaw of the head relative to the ground plane.

This gives the necessary angles to calculate the horizon.

The height of the head relative to the ground can be calculated by adding the height of the head relative to the torso, with the height of the torso relative to the supporting foot:

$$height = ((x_{head}, y_{head}, z_{head})\cdot\mathbf{t}) + ((x_{support}, y_{support}, z_{support})\cdot\mathbf{g})$$

4.3.4 Inaccuracy from Mechanical Limitations

Each joint in the Nao is equipped with a magnetic rotary encoder, so it is able to sense the rotation of each joint at any time. However, the Nao is not tightly constructed and there is some flexibility in the structure that does not cause the actuated joints to move. This creates and undetectable inaccuracy in the system. Figure 4.3 shows the range of movement (undetectable error) of a Nao with all joints at maximum stiffness.

The height of the camera on the Nao, in comparison to the AIBO dog has both advantages and disadvantages.

With the AIBO dogs, the camera is close to the ground. This means that a small error in the angle of the head projects to a large error on the ground.

The advantage of the Nao is that its camera is further away from the ground, so there is more room for error when projecting the location of an object in a camera frame to the ground, (see Figure 4.4).

The disadvantage is due to the number of joints and the length of each link in the kinematic chain. Set of links and joints from the ground to the head are much greater on the taller Nao than on the AIBO. With room for error in each joint and link, the sum of these errors causes greater undetectable inaccuracy. The gyroscope, as mentioned in chapter 2.3.3, can (to some extent) measure the orientation of the robot in axes parallel to the ground plane. This may be useful in detecting and correcting some of this error.

4.4 Inverse Kinematics

There are two approaches to inverse kinematics:

- Analytical: adding new constraints to the robot's joint configuration and reducing the parameters that define the robot's joint configuration.
- Iterative: moving each joint (of a virtual robot model) in turn, towards the required end-effector position until it's close enough, then move the robot's joints to that set of joint angles.

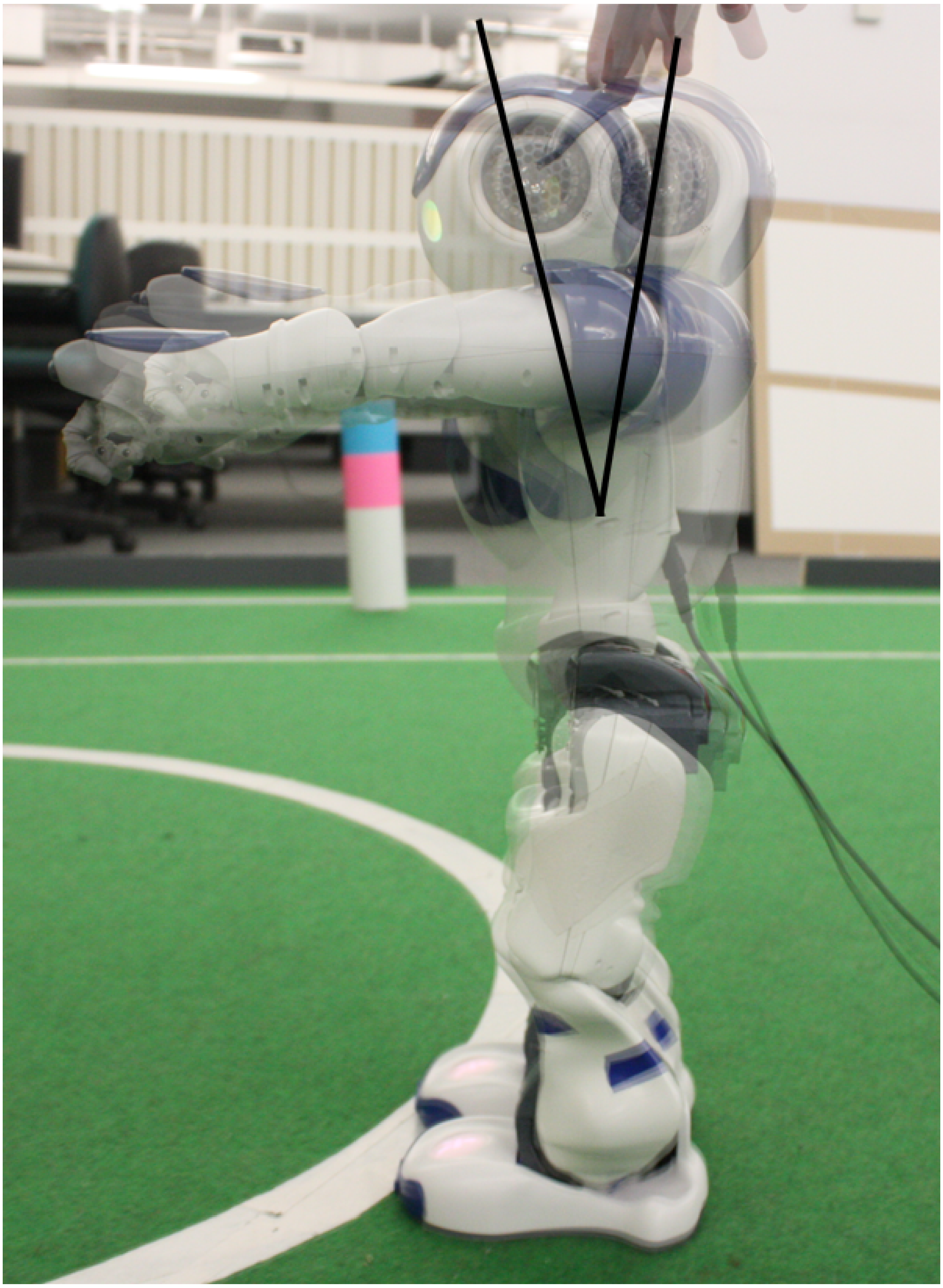

Figure 4.3: The Nao has a substantial amount of undetectable error in its range of movement.

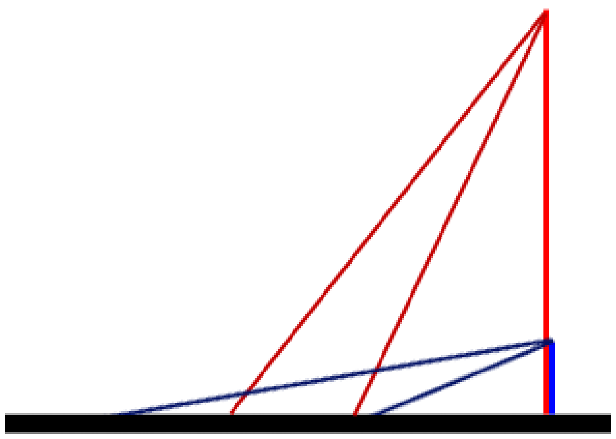

Figure 4.4: With the same angle of error, a higher viewpoint projects a smaller area.

For systems with long kinematic chains where an infinite number of solutions exists, iterative approaches are used, e.g. robotic arms with 5 or more degrees of freedom. [11]

The configuration of the Nao consists of short kinematic chains and the inverse kinematics can be solved analytically. This is achieved by simplifying the parameterisation of the joint space, adding constraints and solving the matrix equations for the end-to-end transform. For example, a common simplification of humanoid leg is to replace the knee joint with a virtual linear actuator. The leg can now be treated as a single link that changes length. The joint trajectories to reach the desired end-effector position can be calculated by solving each link's rotation and translation matrices (as defined in chapter 4.3.1). Due to the simple configuration of the Nao, there is negligible danger of choosing joint trajectories that cause collisions of links, so the computed joint angles can be executed simultaneously, or in any order.

Another approach, particularly useful for simplifying the creation of hand-crafted movements, is to re-define each joint to be relative to the supporting foot. The relation between the new parameterisation and the physical robot joint angles can be derived trigonometrically. For example; when a change is made to the supporting ankle pitch the torso still maintains its angle relative to the ground. Figure 4.5 demonstrates a possible re-parameterisation of the pitch joints, in the sagittal plane.

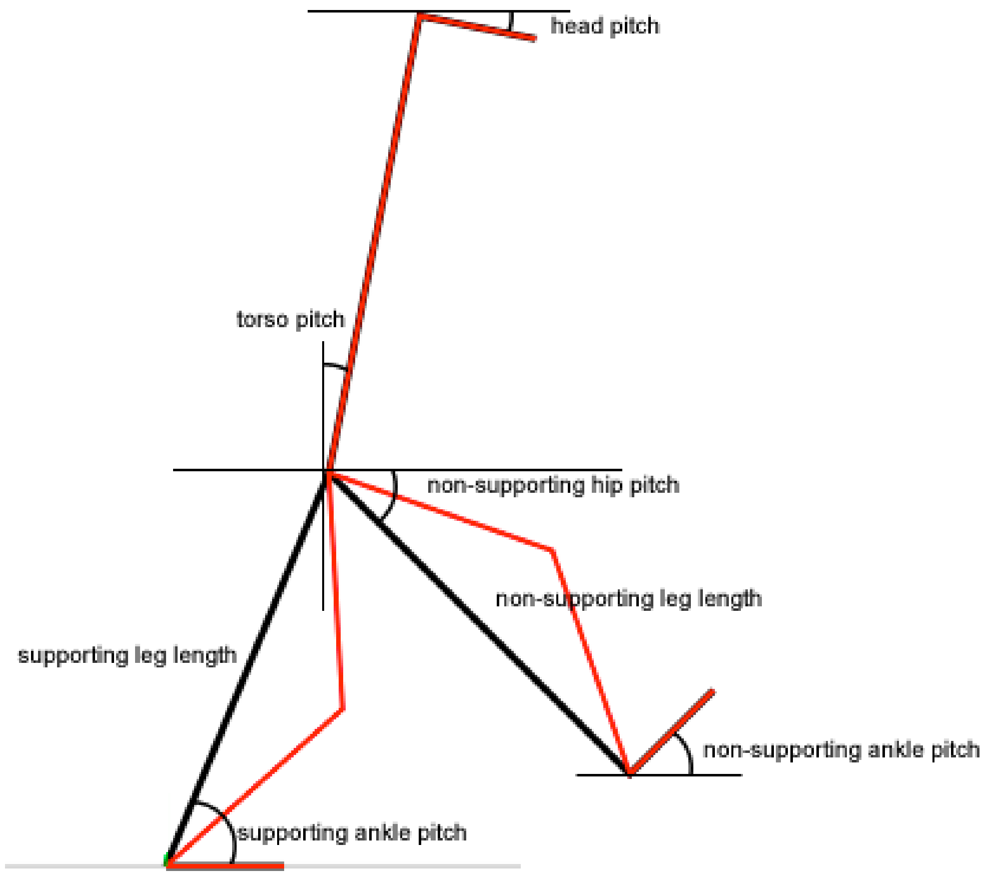

Figure 4.5: Simplified pitch parameters, relative to the ground plane (supporting foot), replacing the knee joint with a linear 'virtual leg':

- supporting foot: left or right (the foot assumed to be on the ground)
- supporting ankle pitch (angle between ground plane and supporting virtual leg)
- supporting virtual leg length
- non-supporting hip pitch (angle between ground plane and non-supporting virtual leg
- non-supporting virtual leg length
- non-supporting ankle pitch (relative to ground plane)
- head pitch (relative to ground plane)
- torso pitch (relative to the perpendicular of the ground plane)

The physical joints of the robot can be represented as a function of these virtual joints.

4.5 Gaits and Walking

4.5.1 Phases of Walking

The gait, or walking pattern, of a biped robot consists of a single-support and a double-support phase. i.e. the phases when one foot, or both feet are in contact with the ground.

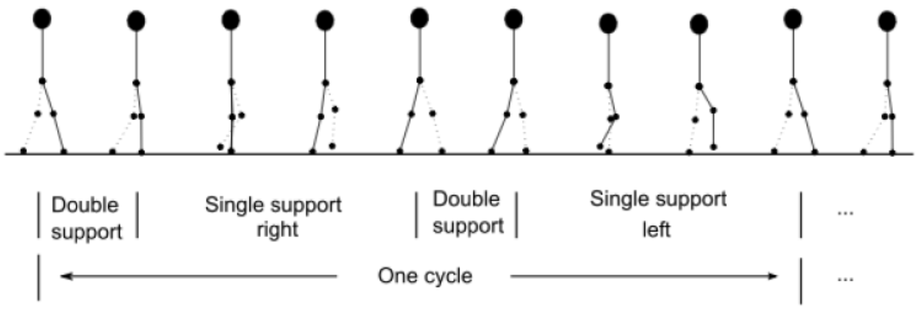

Figure 4.6: Phases of Walking.
[16]

These phases are split into further sub-phases, defining the lift, swing and weight acceptance of alternating legs. These sub-phases are often named and defined differently, depending on the approach.

4.5.2 Statically Balanced Walking

A robot is statically stable if it remains indefinitely stable if all motion of the robot is stopped (frozen) at any point in time. Statically balanced walking is achieved by projecting the Centre of Gravity (CoG) on the ground and ensuring that it is always inside the foot support area. The foot support area is defined as the the foot surface of the supporting leg in a single support phase or as the minimum convex area containing both foot surfaces in a double support phase.

As the dynamics of the robot are not taken into account, static walking speeds must remain slow enough that inertial forces are negligible. This usually results in static walkers having have large feet and strong ankle joints. This initial approach to walking has been largely abandoned due its slow speed and lack of realistic and agile movements.

4.5.3 Dynamically Balanced Walking

Dynamically balanced walking allows for the CoG of the robot to be outside of the support region whilst remaining dynamically stable. There is no single clear criterion to determine if a gait is dynamically stable, however most dynamically balanced walking algorithms use two stability criteria: the Zero Moment Point and/or the angular momentum of the robot, treating the robot as an inverted pendulum pivoting on the support area of the foot.

4.5.4 The Zero Moment Point

The Zero Moment Point (ZMP) is “the point on the ground at which the moment of the ‘ground reaction force’ is zero” [Shimojo Araki Ming & Ishikawa], or as Tak Fai explains:

> “A robot’s walk is considered balanced as long as the zero moment point remains inside the contact area during the process... The zero moment point is closely related to the position of the centre of mass. When the ground projection of the centre of mass of an object is within the contact area, the gravitational force and the normal force from the contact surface cancels each other out and thus the object remains stationary. The zero moment point extends this concept and includes the forces acting on the object other than gravity. It is the projection of the sum of gravity and forces acting on an object on to the contact surface. When the zero moment point lies within the contact area, the torques generated by gravity, force acting on the object and the normal force of the contact surface zero out. There are no resultant tipping moment acting on the object and thus it is balanced. If there is no force acting on an object other than the gravity, the zero moment point is equal to the ground projection of the centre of mass.” [16]

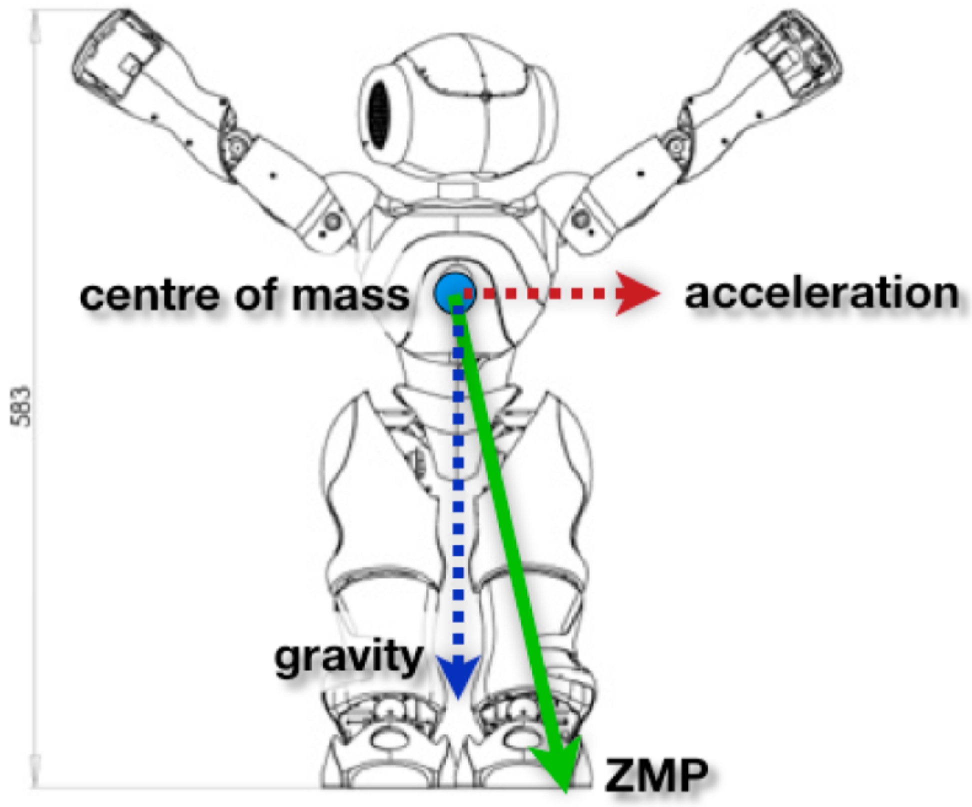

Figure 4.7: The relationship between the centre of gravity and the ZMP.

4.5.5 Calculating the ZMP on the Nao

The moment of the ground reaction force, M, can be calculated from the force perpendicular to the ground, on the foot's surface. This moment is zero at the ZMP, thus $M_x = M_y = 0$ is satisfied at the ZMP with coordinates (x_{zmp}, y_{zmp}). This gives [4]:

$$M_x = \int\int_D (x - x_{zmp})\ f_z(x,y)\ \mathrm{d}x\ \mathrm{d}y\ = 0$$

$$M_y = \int\int_D (y - y_{zmp})\ f_z(x,y)\ \mathrm{d}x\ \mathrm{d}y\ = 0$$

where $f_z(x,y)$ is this force on the foot at any point $(x,y)$and domain D indicates the foot area. The ZMP (x_{zmp}, y_{zmp}) is calculated as follows [4]:

$$x_{zmp} = \frac{\int\int_D\ x\ f_z(x,y)\ \mathrm{d}x\ \mathrm{d}y}{\int\int_D\ f_z(x,y)\ \mathrm{d}x\ \mathrm{d}y}$$

$$y_{zmp} = \frac{\int\int_D\ y\ f_z(x,y)\ \mathrm{d}x\ \mathrm{d}y}{\int\int_D\ f_z(x,y)\ \mathrm{d}x\ \mathrm{d}y}$$

Since the Nao only has four foot force sensors (FSRs) per foot, this can be simplified to:

$$M_x = \sum_{i=1}^{4}\sum_{j=1}^{4}(x_i - x_{zmp})\ f_z(x_i, y_j) = 0$$

$$M_y = \sum_{i=1}^{4}\sum_{j=1}^{4}\ (y_j - y_{zmp})\ f_z(x_i, y_j) = 0$$

and

$$x_{zmp} = \frac{\sum_{i=1}^{4}\sum_{j=1}^{4} x_i \; f_z(x_i, y_j)}{\sum_{i=1}^{4}\sum_{j=1}^{4} f_z(x_i, y_j)}$$

$$y_{zmp} = \frac{\sum_{i=1}^{4}\sum_{j=1}^{4} y_j \; f_z(x_i, y_j)}{\sum_{i=1}^{4}\sum_{j=1}^{4} f_z(x_i, y_j)}$$

where $(x_1, y_1), (x_2, y_2), (x_3, y_3)$ and (x_4, y_4) are the coordinates of the four FSRs per foot.

As stated in Chapter 4.5.5, this is difficult to achieve with the current foot sensors due to the FSR sensors not correctly correlating to an independent point on the sole of the foot.

The ZMP can also be defined in terms of the dynamics of the robot:

$$x_{zmp} = \frac{\sum_{i=0}^{n-1} \; m_i(\ddot{z}_i + g)\, x_i - \sum_{i=0}^{n-1} m_i \; \ddot{x}_i \; \ddot{z}_i}{\sum m_i(\ddot{z}_i + g)}$$

$$y_{zmp} = \frac{\sum_{i=0}^{n-1} \; m_i(\ddot{z}_i + g)\, y_i - \sum_{i=0}^{n-1} m_i \; \ddot{y}_i \; \ddot{z}_i}{\sum m_i(\ddot{z}_i + g)}$$

Where g is gravity, x axis is the walking direction and y is to the left and z is upwards. n is the number of body parts, m_i is the mass of body part i. $\ddot{x}_i$, $\ddot{y}_i$, $\ddot{z}_i$ are acceleration of body part i along those axes. x_{zmp} and y_{zmp} are the coordinates of the zero moment point. [16]

The acceleration of the robot in each axis can only be sensed by readings from the IMU, and the velocity and position calculated with greater error accumulated on each integration. As such, the ZMP sensed using this method would be very rough.

4.5.6 Gait Generation

An open-loop dynamically stable gait for a particular robot can be devised offline. This gait generation forms the core of a walking algorithm on top of which sensor input provides an extra layer of control to react to environmental variations. One approach to generating a gait models the single-support phase of a walk as an inverted pendulum. An open-loop gait is then generated which satisfies the condition of keeping the pendulum upright whilst moving in a forward motion (see Figure 4.8) until a double support phase is realised. The sensed ZMP is then used as an auxiliary control to absorb external forces, adapt to uneven ground and control posture, thus closing the control loop. [5]

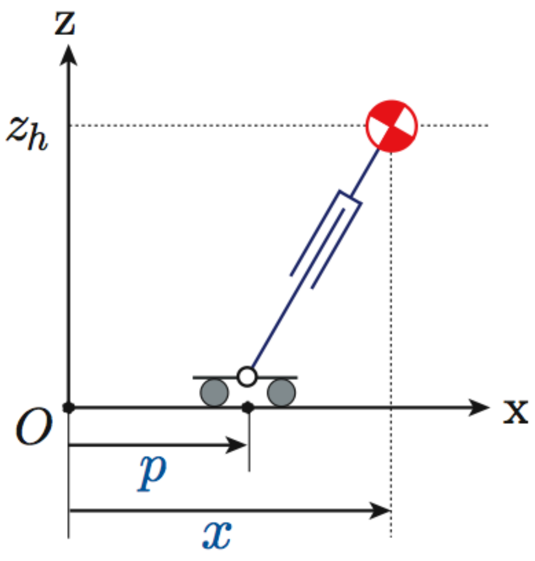

Figure 4.8: A moving cart controls its acceleration to balance an inverted pendulum. [5]

Other approaches involve machine learning parameterised attributes such as stride length, lift height and locus formed [16] or learning functions that transform ZMP targets into ideal centre of mass targets (which can then be achieved by calculating joint trajectories) [10].

4.5.7 Walk Monitoring and Governing

A gait generated offline represents the robot performing a dynamically balanced walk under perfect circumstances. Attributes of the dynamics such as the angular momentum and ZMP trajectory can be modelled under these perfect circumstances, producing the ideal stability criteria. When a walk is performed by a robot in an environment that is not ideal, the sensed ZMP, angular momentum, or other stability criteria can be profiled against their ideal counterparts.

A walking algorithm may already be able to react to small external forces, and maintain stability, however maintaining stability when reacting to larger forces may require additional behaviours. Such behaviours include slowing down the walk speed when the system becomes unstable and stopping to assume a more stable posture to react to sudden external forces (as shown in 4.9). Figure below shows a proposed layered approach to stability control.

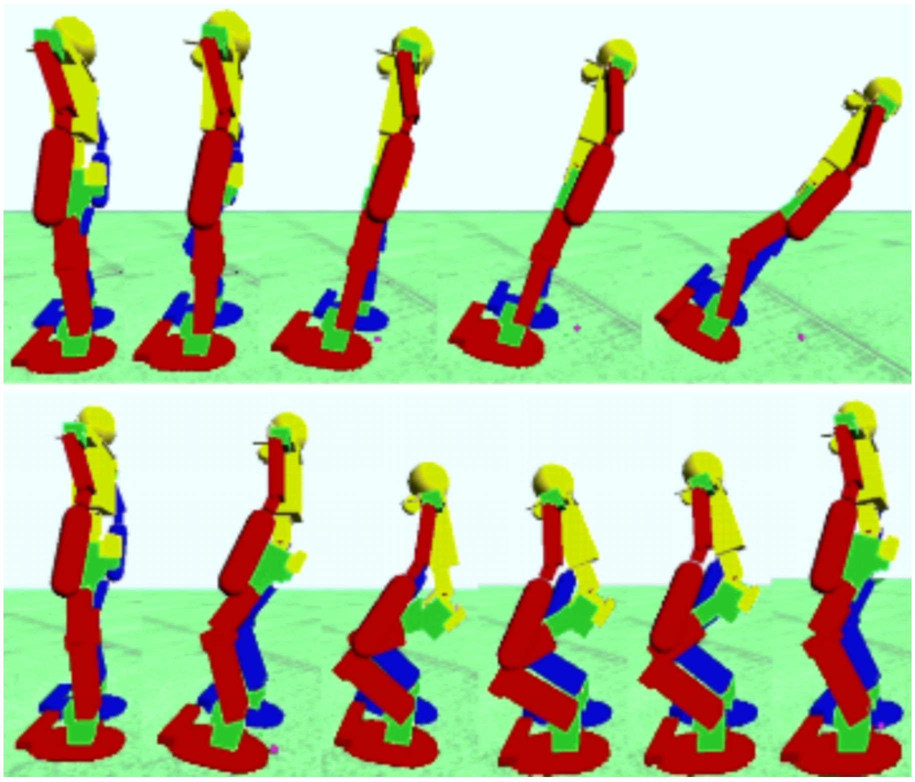

Figure 4.9: Assuming a more stable posture as a reaction to an external force. [13].

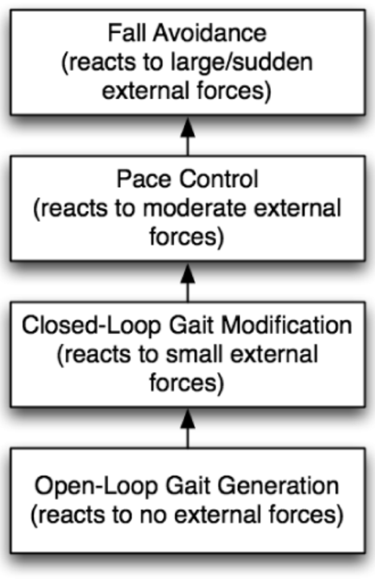
Fall Avoidance
(reacts to large/sudden external forces)
Pace Control
(reacts to moderate external forces)
Closed-Loop Gait Modification
(reacts to small external forces)
Open-Loop Gait Generation
(reacts to no external forces)

Figure 4.10: Proposed approach to stability control.

4.6 Conclusion

This chapter:

- derived optimised functions that perform all required forward kinematics:
  - calculating the position of end-effectors from each joint in a chain
  - calculating the centre of mass of the robot
  - finding the head height from the ground
  - finding the orientation of the ground/horizon relative to the head
- identified and addressed possible causes for inaccuracy in the forward kinematics calculations
- explored two different methods of calculating inverse kinematics - analytical and iterative methods
- proposed a human-friendly re-paramaterisation of the robot, identifying those necessary to control and placing all joints relative to the ground plane.

- explored theories behind gait generation and stabilisation
- showed how the Zero Moment Point can be calculated from four foot-force sensors, and
- proposed a layered approach to walk stabilisation

Chapter 5

Implementation of the Locomotion Module

5.1 Module Structure

The locomotion module was designed to reflect a simplified version of the stability control model mentioned in chapter 4.5.7, with a 'Walk Governor' object responsible for profiling the sensor data, halting the walk if it was thought to be unstable and re-continuing the walk once stability was achieved. See Figure 5.1.

The profiling functionality was abandoned after too much time was wasted concentrating on the dysfunctional foot-force sensors when the robots were already running a stable and well-performing open-loop walk. The Walk Governor became a convenient wrapper for setting the required parameters for the walk. Due to the rushed nature of the Robocup competition and some confusion over the purpose of the Walk Governor object, this object eventually contained most of the locomotion code.

Developing a custom walking engine is not a trivial task, especially in the first experimental competition of a new league and platform. The Aldebaran-provided walk (part of NaoQi's ALMotion module), although open-loop, was stable enough to work with when we were lucky enough to have working hardware.

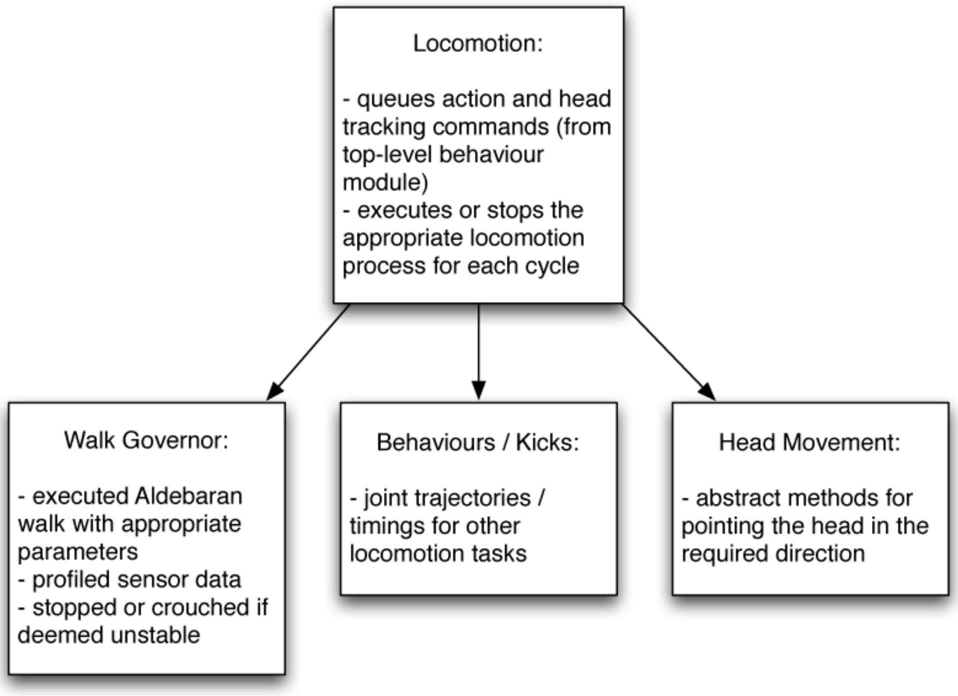
Locomotion:
- queues action and head tracking commands (from top-level behaviour module)
- executes or stops the appropriate locomotion process for each cycle
Walk Governor:
- executed Aldebaran walk with appropriate parameters
- profiled sensor data
- stopped or crouched if deemed unstable
Behaviours / Kicks:
- joint trajectories / timings for other locomotion tasks
Head Movement:
- abstract methods for pointing the head in the required direction

Figure 5.1: Proposed structure of Locomotion Module.

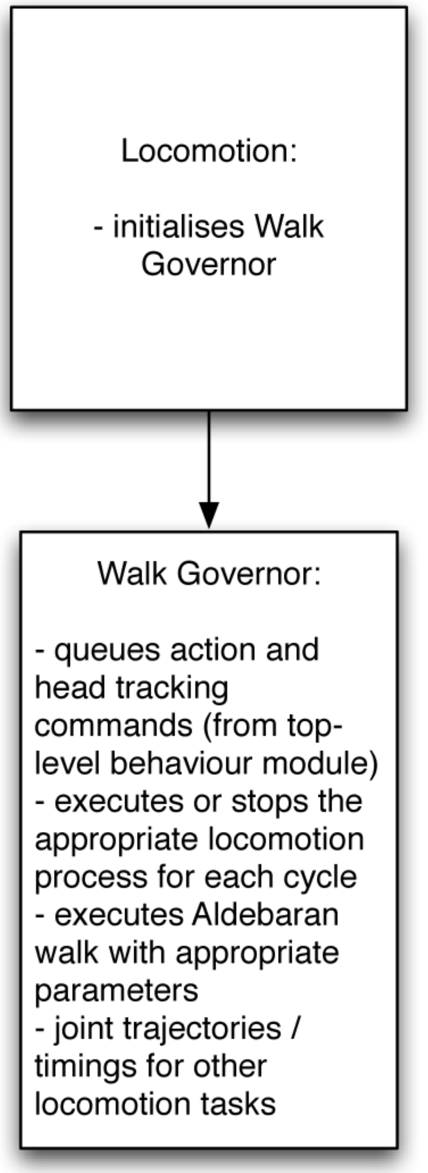
Locomotion:
- initialises Walk Governor
Walk Governor:
- queues action and head tracking commands (from top-level behaviour module)
- executes or stops the appropriate locomotion process for each cycle
- executes Aldebaran walk with appropriate parameters
- joint trajectories / timings for other locomotion tasks

Figure 5.2: Eventual implementation of Locomotion Module

Chapter 6

Locomotion Systems

This chapter describes the details of various locomotion walking systems that were developed for the Aldebaran Nao. Multiple locomotion systems have been tried on the Nao in 2008, both in simulation and on the robot itself. This section provides the basis of theory for further chapters of this thesis and presents the various experiments with locomotion methods.

Since this year has been the first year that the standard platform league has been run most locomotion systems in the league consisted of very basic movements, based solely on the standard Aldebaran walk that was provided. These basic movements consisted of a set of atomic walk actions that included walking (both forward and backwards), turning, sidestepping, and an arc walk.

The standard tuning of the final Aldebaran-provided walk was somewhat sufficient within the robocup domain, however this walk lacked the agility and speed that would be advantageous in the competition. We were able to optimise the provided walk and improve its performance in both speed and stability.

6.1 Urbi Walk

This section gives an overview of the Urbi Nao walk that was supplied with the Webots Nao simulator. This walk became the basis of all our initial work to create our own gait engine. The walk consisted of an open-loop gait that was based on individually tuned joint angles on the robot; this gait was a basic static walk that didn't allow for easy adjustments. It is important to note that this walk is only feasible within the Webots Nao simulator due to an un-realistic physics engine and differences with the real robot dimensions within the Webots simulator. The gait is achieved by the adjustment and movement of 22 parameters these parameters are namely the degree of freedom within the robot. This description will describe this gait in three sections Parameters, Phases and Discussion.

6.1.1 Parameters

The following contains descriptions of all the 22 joints that are adjusted within the Urbi walk. However only some joints actually affect the gait of the robot, the other parameters within the Urbi walk simply provide the gait with a cosmetic appeal. Factors such as arm movement and head movement have shown little influence on the balance or speed of the gait.

6.1.2 Phases

The Urbi walk is divided into 11 phases with each phase representing different trajectories of movement that is required for the robot to achieve a walk. The Urbi walk phases mimics that of a human gait with different single support and double support phases, there is always at least one foot on the ground supporting the robot.

6.1.3 Performance

The Urbi walk wasn't feasible on the actual robot but it provided a well defined starting point for development of our own gait engine.

The Urbi walk presents a hand- tuned and very basic approach to walking with all foot trajectories, inverse kinematic and ZMP calculated offline in a set of static joint angles. This walk lacked the ability to be easily tuned and optimised, hence it lacked the ability to be optimised and tuned.

The Urbi walked presented a set of static hand tuned joint angles, it lacked a parameterisation (ie the ability to adjust CoG, step height, step length, ZMP offset and frequency of steps). Hence we were able to tune or optimise this gait. Speed and timing data wasn't accurately obtainable either through webots as different computers ran the simulator at different speeds. Despite the flaws we started developing on top of this walking gait which will be discussed in further detail in the next chapter.

6.2 Aldebaran Walk v1

With our first experiments on the Nao it was soon realised that the Urbi gait was not feasible on the physical robot. Hence we began experimenting with the provided Aldebaran walk and this became the first gait, which successfully worked on the robot.

Aldebaran provided a gait in the NaoQi package that was based on the Kajita algorithm [5]. It is based upon a dynamic walk that models an ideal ZMP in which the foot must land, the foot trajectory is calculated using a cycloid locus based upon parameters within the walk. This gait is closed loop and parameterised by 8 parameters, which control different aspects of the walk. (See Figure 6.1)

6.2.1 Parameters

The Aldebaran gait presents 8 parameters that can be adjusted to affect the walk of the robot. Further sections of this thesis will present how to optimise and adjust these parameters for stability and speed, however this section will provide an overview of this gait and its parameters.

- Max Step Length : The maximum length in meters of a footstep (0.0 to 0.09 meters)
- Max Step Height : The maximum height of a footstep cycloid (0.0 to 0.08 meters)
- Max Step Side : The maximum side length in meters of a footstep (0.0 to 0.06 meters)

- Max Step Turn : The maximum change in z orientation in radians of a footstep (0.0 to 1.0 radians)
- Zmp Offset X : The zmp offset in the forwards direction in meters (0.0 to 0.10 meters)
- Zmp Offset Y : The zmp offset in the sideways direction in meters (-0.05 to 0.05 meters)
- Distance : The distance in meters e.g. 0.1f
- Number of Samples Per Step : The number of 20ms cycles per step. e.g. 60

6.2.2 Phases

- The Aldebaran walked is quite different from that of the urbi walk while it follows the similar double support, single support alternation its walk stride varies and it achieves a different type of walk. The main phases of the Aldebaran walk are:
- Double Support (starting base)
- Single Support (right leg short stride)
- Single Support (left leg long stride)
- Single Support (right leg long stride)
- Double Support (left leg short stride)

6.2.3 Performance

The Aldebaran gait gave us an insight into what a Nao could be capable of, it was the first time we saw the robot actually walking. From this walk were able to see the mechanical limitations of the robot and gauge what was possible within the robot. This gait was however highly unstable and since it was open-loop, lacked the ability to balance or deal

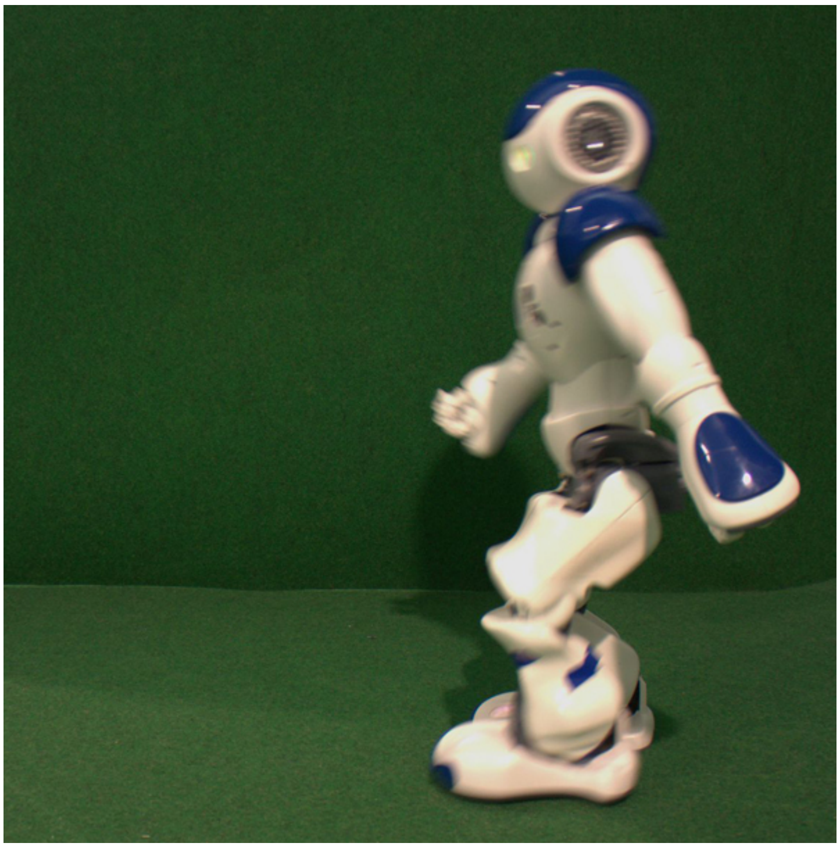

Figure 6.1: Aldebaran Walk v1

with uneven surfaces. We noticed that this gait had a very high centre of gravity (CoG), this meant that robot was prone to small external forces also a higher centre of gravity gave rise to greater momentum causing it to easily tip. The faster the robot walked the more momentum it carried hence the faster the robot walked the more likely it was to tip. Speed is a critical issue within this supplied gait, the ability to quickly move to the ball is a critical skill within the robocup domain. Initially a stable gait took 6 minutes to cross from one goal to another, with our game only lasting 20minutes this simply was too slow. While we were able to neglect some of these issues, this walk still lacked the the ability to change course in the middle of a walking task (an omni-directional walk).

6.3 Aldebaran Walk v2

With the advent of the Nao version 2 came a new gait engine that was supplied with the robots. This new gait engine was based on the previous gait engine (Aldebaran Walk version 1), however it now included new parameters that allowed for it to be highly optimised in both terms of speed and stability. Flaws that were seen in the Aldebaran Walk version 1 were improved and this led to a walk that had allowed for more optimisation and tuning. While this new gait engine was still open loop it was able to stabilise and balance more effectively then Aldebaran walk version 1.

6.3.1 Parameters

Aldebaran walk version 2 added 4 new parameters to the original gait, these new parameters mainly allowed for hip adjustments within the gait of the robot. The most significant change to the Aldebaran walk was the ability for a variable CoG within the gait, effectively increasing the stability of the robot and also increasing the speed of the gait. Overall Aldebaran walk version 2 has 12 parameters, however 8 of these are common with the original walk.

6.3.2 Phases

This new version of the Aldebaran walk followed the same walk phases as that of the original Aldebaran walk (version 1).

6.3.3 Performance

Most noticeably this new gait engine allowed for a lower CoG through the lowering of the knee joints, through this we were able to optimise the gait drastically. Lowering the CoG allows for a lot of momentum to be contained within the gait, since the tipping moment of the robot is much lower, this added momentum translates directly to an increase in the speed of the gait on the Nao. While this gait was still opened looped we noticed that there was a form of passive balancing within the ankles that was not noticed with the original Aldebaran walk. The ankles would adjust according to the CoG of the robot within the coronal plane, this form of passive balancing allowed for more stability but only to a limited degree. Experimentation lead to the conclusion that this passive balancing was attributed to the stiffness within the ankles of the robot, setting a stiffness that was stiff enough to walk but soft enough to adjust according to shifts in the CoG of the robot. This new gait was widely adopted throughout the SPL and became the main gait engine for the majority of teams.

6.4 Gait used in Competition

The second version of the Aldebaran gait became the walking gait which was utilised during the Robocup competition. The urbi walk was helpful to provide insights into the development of our own gait engine.

Chapter 7

Building our own Gait Engine

Bipedal locomotion is a large and open research area within robotics, a focus derived from most classical forms of robot locomotion being un-suited to many human habitats (designed for human locomotion). Robots with human-like biped locomotion would have the ability to navigate through any areas that humans inhabit - a great advantage for many applications of robotics.

7.1 Motivation and Aim

The motivation of this work is to generate a gait engine that will allow for the robot to compete in the robocup SPL competition. However in addition to this main motivation the gait engine must also exhibit:

- Stability
- Agility
- Speed
- Omni-directional capabilities
- Self-correcting behaviour to external forces (closed-loop)

7.2 Stability

This work was started by Steven Wong (rUNSWift 2008 team member) and then further investigated. Active balancing became the critical factor which we focussed on. A balancer was created to allow the robot to stabilise with 1 foot, with an aim of developing our own walk thereafter. We investigated a PID controller using the FSR (Foot Sensor Resistors) values for the adjustment calculation to balance the foot. We calculate the Centre of Gravity in each foot based on these FSR values. A weighted average was then calculated from each CoG based on their distance to the pelvis (to get a common reference point). This CoG is then used in the PD controller with a target of (0, 0) which is point projected right under the pelvis. The proportional component inside the PID controller moves the system towards the target and the derivative component dampens the oscillation (overshooting). The integral component was not used because it measures the cumulative offset, however in a moving robot the foot would always be in a stepping motion and hence cumulative offset over time does not apply because the system is not static. Adding an integral term will massively overshoot the controller. There are 2 PD controllers, 1 on the ankle pitch and 1 on ankle roll. The system was designed to operate in both single support phase and double support phase. However there was a design fault because transition between phases is not smooth and often results in inaccurate/bias FSR readings being passed to the PD controller and cause an over reaction. For example, after the robot takes a step, it places its foot down however if the foot does not touch the ground at the same time then it will produce incorrect reading for the FSRs. Leading the PD controller to think that there is a larger force on 1 side of the robot. There are workarounds, such as filtering the sensor readings or only using the PD controller when all 4 FSR on one foot record a reading etc. The system was demonstrated to be able to stand on one foot and actively balance. The parameters of the controller were to be tuned using reinforcement learning by the downhill simplex method where the cumulative error between actual CoG and target CoG as the robot move swirl around was used as the cost.

7.3 Conclusion

This method was later abandoned because of a software issues in the webots simulator where if the robot moved to a different location in the simulated world, the IMU reading would be skewed and hence the robot could not determine the correct direction to get up when it falls. This prohibits the learning process because after the robot has fallen over, there's no way of getting the robot to stand again without restarting the simulator.

Chapter 8

Walk Optimisation

For robots to be effective in society they must be efficient in movement. Efficient movement differs depending on the objective of the robot [12]. Efficiency is defined different depending on the context and work environment of the robot, in the robocup domain speed is one of the most significant objectives. The ability to approach the ball with speed was the main focus within our research, however due to the nature of the competition this year namely the switch from a quadapedal to a bipedal robot, stability became another important factor. Hence within our optimisations we focussed on speed and stability, this presented a much more difficult task compared to previous work in the 4-legged robocup domain.

Walk optimisation focussed on the tuning of Aldebarans walk, as they presented a well parameterised walk that was devised to allow for performance tuning and hence optimisations. This chapter focus on the optimisation of the Aldebaran walk version 1 and 2 and describes how our experiments were conducted and our resulting gait.

8.1 Summary of Related work

As stated above Speed is a key issue within the robocup competition domain, with the advent of PWalk [2], this was clearly seen within the competition. The need for a parameterised walk was necessary as it allows for easy and clearly defined optimisations. PWalk allowed for rUNSWift to achieve the fastest gait within the 2000 robocup competition hence the team name rUNSWift.

PWalk became a basis to which the gait of the Sony Aibos gait could be tuned and optimised. Further years hand tuned and adjust the loci shape of the gait of the robot and were able to achieve relatively fast speeds. However in 2003 rUNSWift became the first team to utilise an automated gait optimisation process this allowed for significant gains with speed and agility.

8.2 Possible approaches

There are various techniques in which optimisation can be achieved through, both have been utilised by previous years of rUNSWift. An manual or automated approach are the main ways in which optimisation can be achieved, both methods have their relative advantages and disadvantages.

8.2.1 Machine Learning

This entails an automated approach to the robot learning the optimal walking gait by itself. By setting the parameters to tune and a goal the robot can optimise itself through repetitive trials. This represents a good model of optimisation as it requires no human input and is systematic. However depending on the scope of tuning , this form of optimisation can take a long period of time and result in wear and tear of the robot.

8.2.2 Hand tuned

A hand tuned approach requires the knowledge of all the parameters and theory behind the walk before one can begin optimising the gait. A human can quickly notice tends and correlations between tuning parameters and tune accordingly to those trends rather than an exhaustive approach. A hand tuned approach also caters towards the robot with the limitations of the robot and wear and tear in mind.

With hardware unreliable and robot preservation in mind we decided upon a hand tuned approach to the gait optimisation of the robot.

8.3 Motivation and Aim

The aim of this work is to take robot that is already walking and to optimise its speed with regards to stability. In addition to the main motivation of speed there are other influences that motivated this solution:

The solution should not consider hardware or the physical dimensions of the robot, as theoretically each and every robot should be similar. The number of trials required to optimise the walk should be kept to a minimal to reduce the wear and tear on the robot. Optimisations need to solely run on the robot and no off-system processing should be required as this robot is autonomous.

Deviation within the gait was noticed especially whilst walking long distances, this deviation must be kept to a minimal. The gait must be as straight as possible.

The ability to dribble the ball within a gait would be highly advantageous, the final solution should try to cater for the possibility of dribbling the ball.

8.4 Experiment Set up

The experiments to optimise the gait of the Naos were conducted on our full-size playing field. That consisted of a 4m x 6m field that had a green carpet overlayed on the top. With a blue and yellow goals at alternate ends of the field, and all required field lines marked.

Our experiments were conducted along the half-way field line of the field, 10cm increments were measured along this line and marked them out. This allowed for us to test for speed, deviation and odometery. The robot would walk along the half-way line and we would time and mark the end position of the robot on the field. From this we were able to obtain speed, degrees of deviation from the beginning position and odometry data.

8.5 Experimentation (Aldebaran walk version 1)

This sections details the experimentation results from the Aldebaran walk gait version 1.

8.5.1 Detailed Discussion of Aldebaran's Walk

The Aldebaran walk version 1 is based on a cycloid locus that is calculated in order to generate a foot trajectory that the robot will follow. The motion engine of the walk generates an idea ZMP target which each foot should be at in order for the robot to walk with stability. This target ZMP is then mapped to the centre of mass (COM) of the robot, the robot tries to bring its COM within the targeted ZMP area in order to generate the walk. This forms the basis of the Aldebaran open loop walk, however we have noticed that this form of a motion engine utilising only a COM target is inaccurate and causes issues within the walk.

Despite the issues within the walk we have been able to tune and optimise the walk and negate issues that arise from implementation of the walk. Within this walk there are 8 parameters that can be optimised however only 6 of these affect the walk. Through experimentation we were able to observe how each parameters affects the walk, they will now be discussed in detail:

Max Step Length

The maximum step length dictates the maximum length of a footstep the robot is allowed to walk within one stride. The larger the distance the faster the robot is able to move, since less strides are required. However with larger step strides the robot is required to support itself in the single support phase of a walk much longer, since the single support phases of a walk is inherently the most unstable phase of a walk, the robot is prone to instability as the step length increases.

The foot of the Nao compared to that of a human is relatively inelastic hence the momentum of the robot is either absorbed by the surface it is walking or transferred to the robot's body. With a large step length the robot has less contact with the floor and retains most of the momentum and energy that is generated this combined with the fact that robot is in single support phases more often adds a sway on the coronal plane creating instability.

Max Step Height

The maximum step height sets the maximum height in which the foot trajectory cycloid locus can reach. A raised step height adds more area within the cycloid locus hence with an increase step height each foot of the robot must travel more distance.

A raised step height also increases the time in which the robot is in a single support phase, heny ce increasing the instability of the robot much like that of the step length.

Ideally a low step height should be set however this set height needs to be high enough to reduce dragging the foot on the floor and allow for the movement over uneven surfaces.

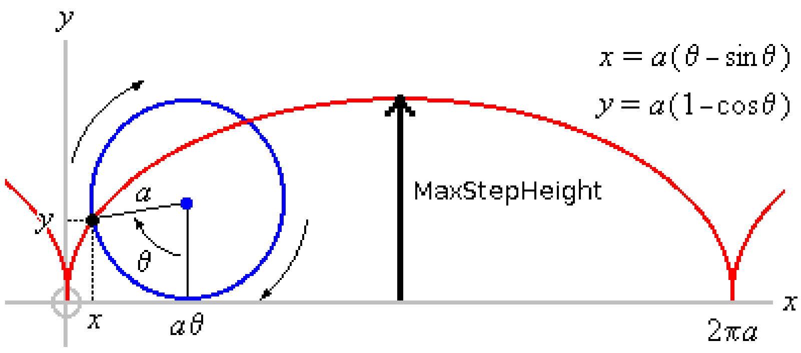

Figure 8.1: Locus of Cycloid [14]

Zmp Offset X and Zmp Offset Y

The ZMP model within the walk approximates the balancing point within the robot however realistically it differs depending on the hardware (wear and tear) and the surface in which the robot is walking upon. In order to optimise the walking gait of the robot the ZMP must be optimised to all environmental factors the robot will face. By offsetting the ZMP we are able to cater for this inconsistency within the environmental factors.

ZMP offset allows for momentum to be increased or decreased within the walk allowing for optimisation to different playing surfaces i.e in an elastic field more momentum is required to propel the robot hence the offset will be greater.

Overall the ZMP offset along the X axis affects the forward momentum of the robot, whereas the ZMP offset along the Y axis affects the sway within the robots gait. The Zmp Offset is a very sensitive parameter and requires careful fine turning.

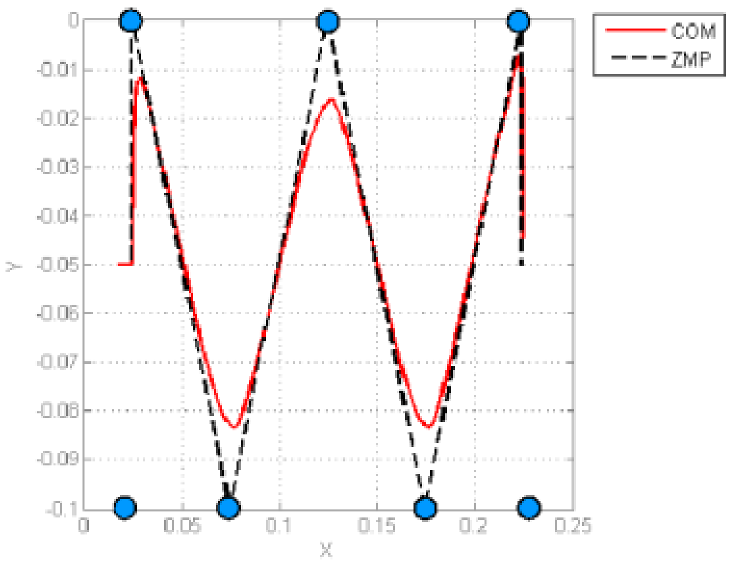

Figure 8.2: Mapping Target ZMP to CoM [14]

Distance

The distance has a relatively low impact on the walk,however with increased walking distances the deviation of the robot from its original intended path also increases however we were able to optimise this and reduce the noticed deviation.

Num Samples Per Step

This parameter dictates the step frequency of each step of the robot. The lower the value the faster the robot walks, this parameter details how many 20ms cycles it takes the robot to take a step.

The faster the step movement the more ZMP offset that is required as the robot is carrying a lot of momentum. However as the speed increases the ZMP area also becomes smaller hence the robot is more prone to tip and fall especially with a high CoG.

8.5.2 Optimisation of Aldebara's version 1 walk

Through experimentation we were able to observe how different parameters within the walk influenced the gait of the robot. We based our optimisation on these findings with an emphasis on speed and stability.

Optimisation began with the default parameters that were supplied by Aldebaran, we noticed correlations between multiple parameters and trends.

Optimisations to the walk followed the following process:

- Optimisation of step frequency, ZMP offset
- Optimisation of Step length and height

The following details some important trends that we noticed(all values use the default Aldebaran parameters unless specified) :

Step Height vs Speed

Step height has a noticeable correlation to the speed of the Aldebaran walking gait simply as it varies the foot trajectory distance of the gait. The higher the step height the more time the foot is in motion, this also leads to the robot being in a single support phase longer.

Step height vs Deviation

As the step height increases we have noticed that the deviates of the walk dramatically deviations from the straight path it is meant to follow. After close examination and experimentation we attributed this to the body momentum of the walk as the step height increases the foot is held for longer periods in the air. During this single support phase

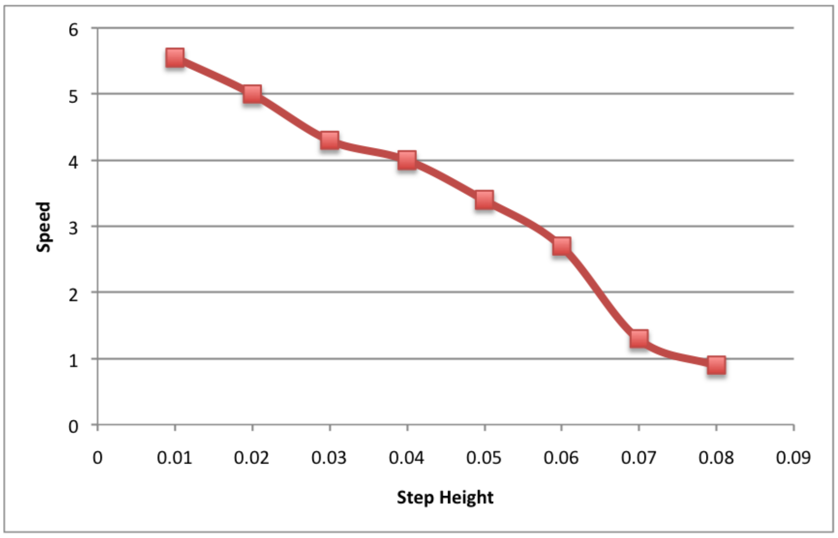

Figure 8.3: Step Height vs Speed

the body sways on the coronal plane more than in a double support phase, this sway led to the deviation in the walking gait of the robot.

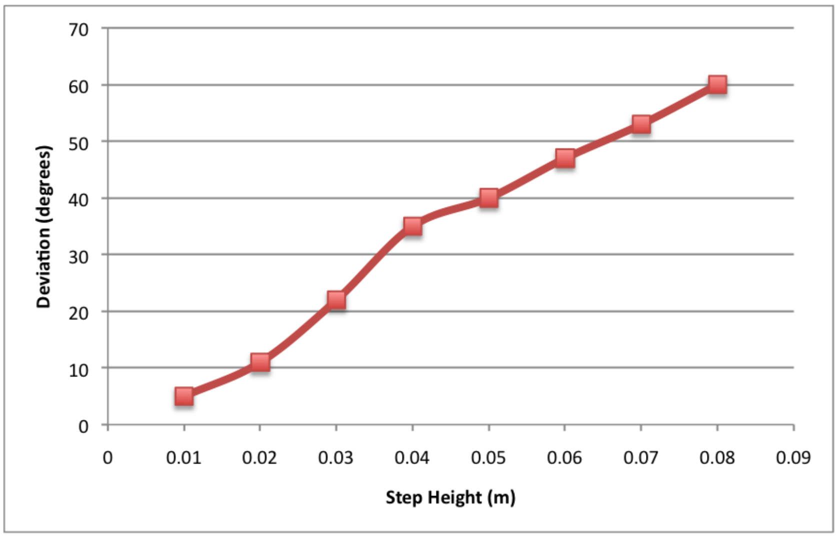

Figure 8.4: Deviation vs Step Height

Through analysing and experimenting with the Aldebaran gait we were able to hypothese the best solution within the gait, through tuning around this ideology we were able to find a well balanced niche between speed and stability. Our ideal solution would have a low step height, fast step frequency.

8.5.3 Results

The final gait that was developed from the Aldebaran Version 1 walk gait had a short stride length to keep the robot stable, a relatively high step height and slow step frequency. We tried to reduce the step height to reduce the distance the foot trajectory was required to travel, however due to hardware limitations (as all our experiments were conducted on this gait were conducted on the Nao version 1) we weren't able to reduce this height and step frequency was slowed due to the high CoG the robot tended to tilt and lose a its stability.

The following are the optimal parameters we resulted with:

- Step Frequency 30
- Step Height 0.06
- Step Length 0.02
- Zmp offset X 0.02
- Zmp offset Y 0.018

The experimental results in the following table represent the speed of a walk and the lateral deviation from its expected path.

Table 8.1: Timing vs Deviation Data

Distance	0.5	1.0	1.5	2.0	2.5	3.0	3.5
Aldebaran gait v1	9.0s / 5	17.1s/9	24.0/20	31.5/5	40.0/10	47.0/-10	Fell

The walking gait was able to achieve a maximum speed of 5.55cm/second opposed to the default Aldebaran gait speed of 1.67cm/second.

8.5.4 Conclusion

Overall we were able to optimise the robot to an extent, there were various flaws within the gait engine that limited our ability to optimise the walk any further. Namely the hardware limitations of the robot, and the high CoG of the gait that wasn't variable.

8.6 Experimentation (Aldebaran walk version 2)

With the introduction of the new Nao (the Aldebaran Nao version 2), the Nao middleware that became the interface between our software and the robot was also upgrade (Naoqi 0.1.18). With the introduction of this new Naoqi came a new walk that introduced new parameters into the Aldebaran walk.

This section details how we optimised this new walk gait.

8.6.1 Detailed discussion of Aldebaran walk parameters

Through further experimentation with the new Aldebaran walk we were able to analyse and understand the theory of the new gait, the following is a detailed description of the new parameters. Understanding and analysing these new parameters we were able to optimise and tune the new Aldebaran gait efficiently.

Left and Right Hip Roll Backlash Compensator

This parameter varies the outwards movement of the Hip roll during the walking gait of the robot, it allows for issues within the plastic casing of the robot to be negated. A major cause of instability is due to the robot walking into its own feet, with this parameter it became possible to avoid this. The adjustment of the Hip roll backlash compensator also affects the amount of step time of the gait. Using this parameter it is also possible to adjust the ratio of power input to the hip roll of the robot.

This parameter mainly affects gaits that require sideways movement, hence it had little influence on the walk.

Hip height

This is the most significant parameter within the new Aldebaran walk it allows for the CoG within the robot to be dramatically lower, reducing the tipping moment within the robot and allowing for more speed and momentum to be contained within the walk. The hip height is lowered through the hip pitch and knee pitch, the advent of this parameter we were able to optimise the walk much more dramatically.

Torso Y Orientation

The torso orientation dictates the amount of body lean during the walk along the transverse plane. The walk engine uses the ankle joint to compensate for this lean in the body of the robot, as the body is leant more forward during the walk the ankle joints are set higher to try to keep the body from tipping over. The ankle joints are lower when the robot leans its body back. This parameter presents a method to vary the momentum of the robot dramatically, however as the momentum of the body varies the ZMP also varies dramatically.

8.6.2 Optimisation of Aldebaran's version 2 walk

Optimisation of the new Aldebaran walk followed a very similar process to that of the original Aldebaran walk, again the focus was on stability and speed.

We focused our optimisations around a low CoG approach. As this allowed for Aibo P-Walk to speed up its gait. We also tuned for a shorter step stride so that we were in a double support phase more often then a single support phase so that the walk was more stable. We adjusted for this short stride length by increasing the step frequency so that the robot covered a greater distance within a short time. The following details some important results that we concluded from our experimentations.

The lower the hip height the lower the centre of gravity of the robot, hence the lower the hip height is the more stable the robot is.

Hip height vs speed

The hip height vs speed analysis provided some interesting results, the hip height correlates to an increase in speed to a certain point where it would then correlate to a decrease in speed. This was due to the fact that the hip height would provide a lower CoG and the step frequency on the robot can be increased, however as the hip height passes a certain threshold the step stride of the robot must be reduced as the low CoG limits the stride of each step. With this reduction in stride length comes the reduction of distance travelled per step and hence the overall speed of the robot.

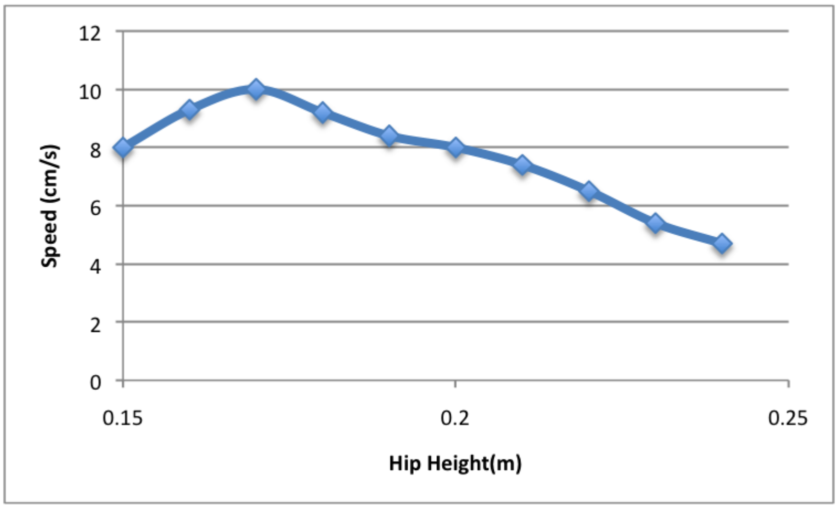

Figure 8.5: Hip Height vs Speed

Torso orientation vs Speed

We noticed that the torso orientation has little correlation to the speed of the walking gait within the robot. A small degree in the forward lean of the body results in some speed gains but the torso of the body must remain upright in order for the walking gait to be stable.

8.6.3 Results

¿From the optimisation of this walk we were able to improve all the deficiency of the Aldebaran version 1 walking gait. Due to the improvement of the hardware on the robot we were able to reduce the step height of the robot, hence reducing the trajectory of the foot cycloid locus, resulting in a faster walk. The lowering of the step height also meant that the robot spent less time in a single support phase as opposed to a double support phase, giving it more stability.

The CoG within the robot was also lowered within this new gait engine, this allowed for the step frequency to be dramatically increased allowing for a faster walk. This CoG also meant that the robot was much more stable within its walking gait as the robot's tipping moment now is much lower.

We also concluded that if a short step stride length was taken with a high frequency of steps the robot is able to walk over uneven surfaces with a high degree of stability. We attributed this to the fact that with this set up the robots gait has a minimal time in the single support phase.

Our resultant gait from the optimisation of the Aldebaran walking gait version 2 results in a walk with a low CoG, short stride lengths, low step height and a high step frequency.

The following are the optimised parameters we used:

- Step Frequency 20
- Step Height 0.010
- Step Length 0.04
- Zmp offset X 0.018
- Zmp offset Y 0.02
- Left Hip Roll Backlash Compensator 4.5
- Right Hip Roll Backlash Compensator -4.5

- Hip Height 0.17
- Torso Orientation 5.0

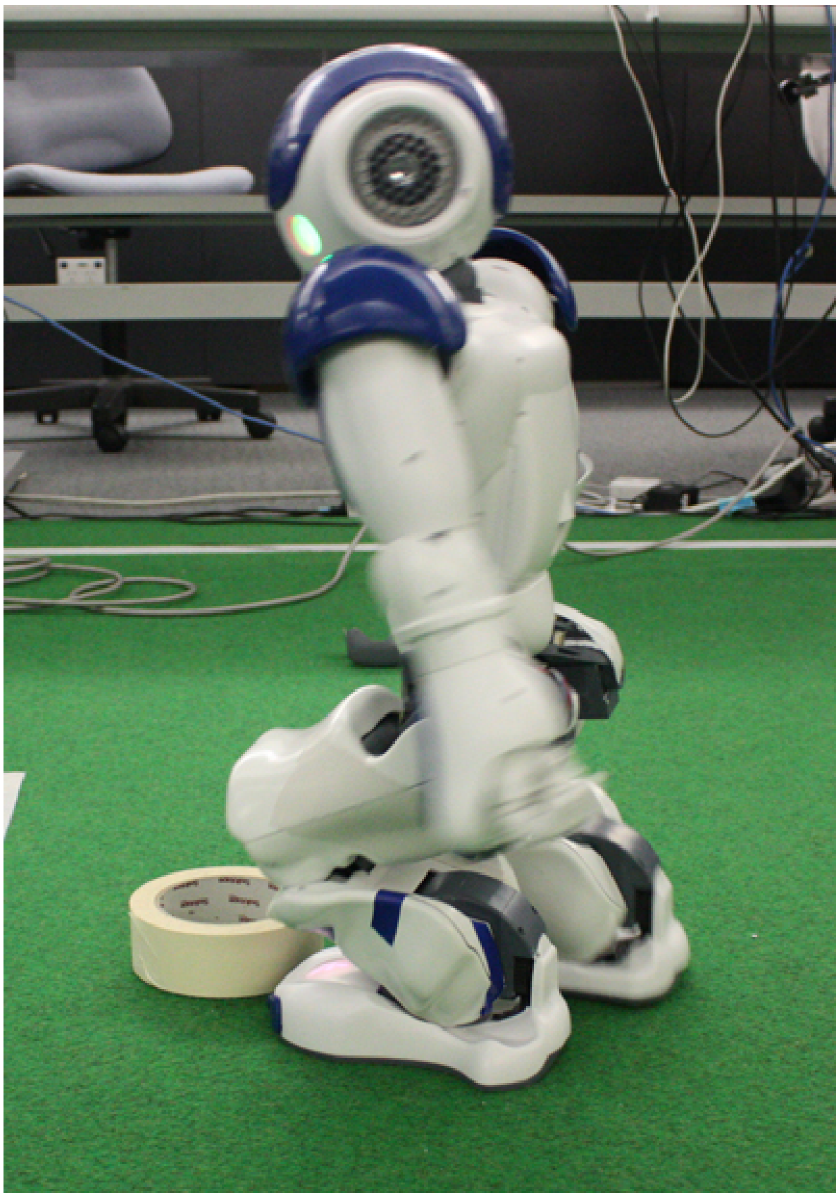

Figure 8.6: Our optimised walk

The experimental results in Table 8.2 represent the speed of a walk and the lateral deviation from its expected path.

Overall we were able to achieve a maximum walking speed of 10.0cm/s with a high degree of stability over uneven surfaces, this compared to the default Aldebaran version 2

Table 8.2: Timing vs Deviation Data

Distance	0.5	1.0	1.5	2.0	2.5	3.0	3.5
Aldebaran gait v2	5.7s /0.0	10.7s/5.0	15.4/10	19.3/15.0	24.3/21.0	29.7/24.0	36.1s/30.0

walking gait parameters which achieved a maximum walking speed of 7.0cm/s with some instability over uneven surfaces.

8.6.4 Conclusion

Overall we were able to achieve a highly stable and fast walking gait that proved to be one of the best within the robocup competition. Despite this result there are still issues within the walk that we wanted to improve but weren't able to due to hardware and the closed source nature of Naoqi.

One main issue within the walk was that it was an open loop, the walk relied on no sensor feedback to stabilise or adjust it gait. The walk was able to deal with uneven surfaces because of optimisations made, however the need for a active stabiliser is still critical to the walk for speed and stability.

Another issues within the walk was the lack of an omni-directional walk (a agile walk that was able to change direction during the walk). The current gait we have has to stop, go to an initial position and then change the direction the walk. This results in lost of time and agility. In conclusion the gait sufficed for this years competition further years will require a much more agile and faster walking gait. This will require the development of our own gait engine.

8.7 Other possible approaches

Through the competition this year we noticed different approaches to the optimisation of Aldebaran's walking gait. The following provides an overview of these different approaches:

8.7.1 High CoG and Wide Stride Length

A high CoG presents a much less stable walking base for the robot, however it gives a greater vision depth as a robot who is standing higher can see further. Through experimentation the higher the robot is the less stable it base is, however as the height of the robot increases its stride length can increase and hence the distance that each step can take. While the step frequency of the robot must be kept relatively low because of the high CoG we can still maintain a high speed gait through a long stride length. This long stride also provides a good mechanism of kicking the ball with will be discussed in this thesis.

8.7.2 Low stiffness Walk

We utilised a medium stiffness walking gait throughout the competition but experimentation has showed us that a low stiffness within the gait can be beneficial with uneven surfaces. The low stiffness allows the joints of the robot to absorb the instability caused within the uneven surface and allow for the robot to stay upright and remain walking. While our approach to uneven surfaces worked extremely well this approach is notable.

Chapter 9

Other Gait Optimisations

Within the robot domain a steady and fast walk is key, but similar to any real game of soccer other forms of gaits are also required in order for the robot to be able to perform and play a game of soccer. Hence a focus on other forms of gaits is also required namely a sideways walk and turn. These other forms of gaits allow the robot to manoeuvre around the field and to the ball, change it's heading direction and allow it to line up to a ball. This chapter will detail all the other forms of gaits that were possible on the robot and how they were optimised for speed and stability. Our optimisations are based on the Sideway Walk and Turn provided with Naoqi 0.1.18 written by Aldebaran.

9.1 Related work

These forms of gaits presented much more of a stability issue than those of the Sony Aibos. Whilst in essence they are similar movements the basis in which they are conducted is entirely different. The shift from an inherently stable quadapedal system to a bipedal system magnifies the issue of stability. Trying to optimise for speed is a much more difficult problem with this lack of stability within the robot.

9.2 Approach

Again with these other forms of gaits we had the ability to optimise using two approaches an automated machine learnt approach or a hand tuned approach. Because of reasons stated above such as hardware unreliability and fragility, we decided upon a hand tuned approach.

9.3 Motivation and Aim

The motivation for these other forms of gait movements are similar to that of the motivation and aim for that of the walk. We optimised these gaits based on speed and stability, with an emphasis to avoid wear and tear towards the robots by minimising the number of experiments we conducted.

9.4 Experiment Set up

Each gait had a different experimental setup that was required for optimisation. This was due to the fact that each gait achieved a different purpose and movement. The following details different experimental set up of each gait:

9.4.1 Sideway walk

The sideways walk utilised the same experiment set up as that of the walk gait optimisation. The only difference was the fact the robot was positioned to the sideways to allow it to walk down the half-way line. Measurements were taken from the leading foot depending on which direction the robot was moving.

9.4.2 Turn

The turn was optimised by standing the robot on a circle with 10 degrees increments marked out on it. The robot would turn a given degree and the time, accuracy and stability of the turn would be analysed and tuned accordingly.

9.5 Sideway Walk

A sideways walk is a walk the allows the robot to move from along its coronal plane without the requirement of movement on the transverse plane. It allows the robot to move form side to side it is especially useful to use whilst trying to line up to the ball or get behind the ball. A simulator version of this gait was developed by our team however as stated earlier work in the simulator did not translate directly to the robot, due to time constraints we decided to use the sideways walk that was provided by Aldebaran. The section will detail the results from our experiments and will discuss our optimised sideway walking gait.

9.5.1 Parameters

Much like the walking gait there are two versions of the Sideways walk, however this thesis focussed on the second version of this gait. This second version of the gait was provided with Naoqi 0.1.18 and contained the following parameters within the gait that could be optimised and tuned:

Max Step Height

The Step height correlated to the amount of swing generated within the torso during the Sideway walking gait, we noticed that the torso should have a minimal amount of sway in order to make a efficient and speedy sideways walk, hence step height should be kept to a minimal.

Max Step Side (The maximum side length in meters of a footstep)

Max step side dictates the length of each stride, as the stride increase so does the distance covered by each step. However we noticed that as the stride length increased so did the sway of the torso causing the instability and the robot to tip during the gait.

ZMP Offset X and Y

The ZMP offset along the X axis has little influence on the sideway walking gait, however we noticed that the Y offset must be set and fined tuned in order for the gait to be stable, the faster the gait moved the more ZMP offset required along the Y axis.

Distance

The sidestep had a strong deviation as the distance increased. However it was always stable despite covering long distance over slightly uneven surfaces.

Num Samples Per Step

We noticed that the step frequency had little impact on the stability of the sideways walk if the torso of the robot didnt sway. We noticed that as the step frequency increased the ZMP along the Y axis had to be adjusted by the relative balance was not undermined.

Left and Right Hip Roll Backlash Compensator

The hip roll backlash compensator was utilised heavily within this gait, we noticed that the robot had slightly uneven motors within each of its hips. The right hip seemed to generate more power than that of the left hip, utilising the Hip roll backlash compensator we were able to tune and adjust this ratio of power. Utilising these parameters provided a even gait within the sideways walk.

Hip Height

The hip height lowered the CoG within the sideway gait, this dramatically affected the sway and swing of the torso of the robot and hence the stability of the robot. The lower the hip height the more stable the gait became, however if the gait was too low there not enough momentum to keep the robot moving sideways hence it was a balance between stability and functionality.

Torso Y Orientation

The torso orientation has little affect on the sideways walk, as the torso lean affected the transverse plane not the coronal plane.

9.5.2 Phases

The Aldebaran Sideway Walk consisted of four main phases this phases were quite similar to that of the walk, however they propelled along the coronal plane and not the transverse plane. The phases of the side step are:

- Double support phase (starting base)
- Single support phase (transition of one leg sideways)
- Double support
- Single Support (transition of remaining sideways)
- Double Support (base to starting base)

9.5.3 Experimentations

Our optimisation of the sideway walk was based on speed and stability, we began with the default Aldebaran parameters and followed the following methodology:

- Optimisation of Step frequency, Step height and length
- Optimisation of ZMP offset and CoG

Through initial experimentations we concluded that a sideway walking gait with a low step height, fast step frequency, low CoG and minimal torso movement presented the body model of the sideway walking gait. We optimised the sideways walking gait around this ideology.

9.5.4 Results and Conclusion

Our final sideways walk proved that our ideology was correct, minimal torso movement increases the stability of the gait while still allowing it to achieve a high performance. The following are the optimised parameter values that were utilised by our team within the robocup competition:

- Step Frequency 20
- Step Height 0.012
- Step Side 0.06
- Zmp offset X 0.02
- Zmp offset Y 0.018
- Left Hip Roll Backlash Compensator 4.5
- Right Hip Roll Backlash Compensator -4.5
- Hip Height 0.17
- Torso Orientation 5.0

Arguable we had the best sideways walk within the whole league.

9.6 Turn

The ability to walk forwards and sideways is critical within a game of soccer, but the ability to turn and change or direction of heading is crucial too. Hence in addition to our walk and sideway walk we optimised and tuned a turn for the Nao.

Again this turn was based on Aldebarans turn that was supplied with Naoqi 0.1.18, we optimised based on speed and stability.

9.6.1 Parameters

The turning gait provided by Aldebaran utilises similar parameters to that of the walk and sideway walk. The following details how the parameters affect the turn gait of the robot:

Max Step Height

The maximum step height of each foot affect the accuracy of the turn, the more height within a turn the more deviation from the intended turn angle. This is due to the fact that a higher step height adds more sway in the body affecting the placement of the foot when it touches the ground again. Since the trajectory generated by the foot is again a cycloid the step height also affect the distance of each foot step.

Max Step Turn The maximum change in z orientation in radians of a footstep

The step turn is a parameter that only affect the turn gait, it dictates the maximum angle a single foot is able to move in a step. While a large step turn increaes the speed of the turn it also increases the momentum in the turning body, a large large step turn leds to instability within the turning gait.

Zmp Offset X and Y

The ZMP offset affects the turning gait dramatically, both the X and Y axis offsets affect the turn. They must both be fined tuned.

Turn Angle

This is the equivalent of the distance parameter within the walk and sideway walking gait, it specifies how much the robot should turn by. It has no effect on the speed or stability within the turn.

Step Frequency

The step frequency of the turn has little effect on the stability of robot whilst it has a low CoG, hence we were able to utilise a high step frequency to have a high turning rate if we kept a low CoG.

Left and Right Hip Roll Backlash Compensator

This much like the sideway walk can be used to alter the power distribution of the hip roll joints, it is mainly used in the optimise the hips so that they generate an equal amount of torque within its turn, allowing it to keep it stability as body sway is minimal.

Hip Height

The hip height tuned the CoG within the robot, as stated above lower the CoG allow for a high step frequency and fast turning speed. The hip height however must allow the sufficient step height to be achieved or else the robot will become highly unstable.

Torso Y Orientation

The Torso orientation of the robot severely affects the turning movement stability.

9.6.2 Phases

The turn consist of five main phases that allow for one foot turn at all with stability and speed, the following details the phases within the turn:

- Double support phase (starting base)
- Single support phase (movement of one foot sideways to turn)
- Double support phase
- Single support phase (movement of other foot to turn)

- Double support (the robot then repeats until it has travelled the required amount of degress)

9.6.3 Experimentation

Optimisation of this gait followed the same methodology as that of the sideways walk. We began with the Aldebarans default values and then tuned based on our understanding of the parameters and the gait engine behind the turn.

Again stability and speed were the key motivations behind this optimisation, however accuracy was also vital and this became another variable that the turn was optimised towards.

9.6.4 Results and Conclusion

Through optimisation and tuning we concluded that the best turning gait utilised a high step frequency and a mid range step turn for a quick and responsive turn. A low step height and CoG increased the stability within this turning gait to allow for this quick and agile response. The following are the parameters we utilised in the robocup competition:

- Step Frequency 25
- Step Height 0.012
- Step Turn 0.314
- Zmp offset X 0.02
- Zmp offset Y 0.018
- Left Hip Roll Backlash Compensator 0.9
- Right Hip Roll Backlash Compensator -1.0
- Hip Height 0.17

- Torso Orientation 5.0

This turning gait allowed for a performance and stability within the robocup competition and was utilised heavily.

Chapter 10

Kicking

The most iconic element within a soccer game is the kick, it is also one of the most vital elements as it allows for teams to score a goal which is the main objective within a game of real or even robot soccer.

Hence it became vital that a kick or a mechanism for scoring goals with the Nao be developed. This chapter details the development of different varieties of kicks that were developed for the Nao.

The basis for the our kick came from a example kick that was supplied with the Nao, kick motions from the robocup humanoid league also influenced the development of our kick.

10.1 Analysis

¿From the robocup competition there were two main branches of kicks that were developed, each having their own advantages and disadvantages.

This section will analyse both these branches of kicks in detail:

10.2 Walk Kick

This kick was employed most successfully by the eventual league winners the Numanoids. The Numanoids developed a walking gait that carried a lot of momentum within the feet, allowing for the Nao to simply walk into the ball and push the ball forwards, with its walking movement. This form of a kick is advantageous as it is agile and quick, it doesn't require the robot position itself stringently as both feet can kick the ball. The robot simply walk into the ball without having to alter it movement, while this type of simplified kick has its advantages it is too overly simplified. The walk kick is based on random probability the ball can come off any part of the two feet. There is a general lack of direction and strength behind the kick, while this kick was effective in this year's competition this was due to a lack of skill within the league.

10.3 Conventional kick

A conventional kick is what most people associate as a kick in the context of soccer. A conventional kick involves selecting which foot to kick the ball with and then lining up behind the ball so that one is able to make contact with the ball with the desired foot.

Once behind the ball one must shift all body weight to the non-kicking side of the body, then move the kicking foot backwards and the body downwards to build momentum. One must then strike the ball with the kicking foot transferring all the momentum and kinematic energy to the ball.

The advantage of this kick is that it has the ability to be accurate, a lot of momentum is also transferred to the ball hence resulting in a more powerful kick.

10.4 Development of a kick

Since a walk kick requires little developed and the result of the walk kick is reliant on probability. We chose to develop a conventional kick for the Nao. The supplied Aldebaran kick provided a good basis for the development of a kick, while it lacked enough power and accuracy it presented a simplified model of a kick.

Through analysis of the humanoid league kicks and the supplied Aldebaran kick we noticed the following common phases and based our kick on those:

10.4.1 Phases

Single Support

This phase shifts all the weight to non kicking side of the body and allows for momentum to be built up to allow for a momentum power transfer to the ball through the kicking foot. It also provides clearance for the kicking foot to kick.

Raise kicking foot

The kicking foot needs to raise off the ground so that move backwards.

Lean torso forward and move kicking foot backwards

This phases builds the potential energy within the kick by shifting the body weight downwards and by moving the kicking foot backwards. This phases was important in the power and distance of the kick.

Shoot with non-supporting foot and move torso upwards

This phase involves moving the leg downwards to strike the ball, while simultaneously moving the body upwards to transfer the potential energy stored to kinetic energy.

Move back to double support

This phase involves going back to the initial kick position allowing the robot to gain back its stability. Picture of the phases we have

10.4.2 Parameters

By observing the phases within the conventional kick, there are common elements within the conventional kick across all the phases, through this we developed a parameterised kick and allow for easy adjustments to vary the kick for different circumstances. The kick was initially developed in Ruby and then ported to C++.

The following are the parameters we developed into the kick:

Torso Bend (along the Transverse (X) and Coronal (Y) plane)

The torso bend correlates directly to the power and distance of the kick, the more angle and downward torso movement within the torso lean phase adds more potential energy towards the kick. However this torso lean also severely affects the stability of the robot.

Movement backwards

The movement of the leg backwards has been parameterised it dictates the amount of swing the leg will have before it strikes the ball. The more movement backwards the more instability within the body after the kicks as the robot has a small support base of one foot.

Movement forwards

This parameter dictates how far the foot swings in front of the robot, this was important as it allowed for the robot to have a larger margin of error whilst lining up behind the ball. The more forward movement on the kick translates to the more distance the robot is able to be behind a ball and still make contact with it. However it is important to note that too much of a forward movement will cause the robot to surpass its tipping point.

Movement sideways

This parameters allows for the adjustment of sideways momentum applied to the ball, the left foot can only apply a right momentum to the ball and vice versa. This parameter is mainly utilised only in the sideways kick. Height of the foot lift

This parameter details the height of foot as it is lifted off the ground for a kick. This parameter must be set high enough not to hit the ground, but also must be low enough to make good contact with the ball.

Speed of Kick

This is an very important parameters as it sets the time that is required from in the torso lean phase to the shooting phase (the speed of the kick). The speed of the kick usually affects the power of the kick in conjunction with the body torso lean, we however have noticed that it has a minimal affect on the stability of the robot.

Ankle Yaw and Pitch

This parameter affect the Yaw and Pitch of the ankle, these parameters have little influence on the kick. However they are utilised to adjust for the form of contact made with the ball.

10.5 Developed Kicks

Through developing a generalised parameterised kick we were able to develop different kicks from this platform. This allowing for a diverse collection of kicks that the Nao were able to utilise, different kicks have their own advantages and disadvantages. The following details the four main kicks that were developed for the Nao competition:

10.5.1 Power Kick

The power kick was the main kick that was developed for the competition, it was a highly straight and accurate kick that was able to score from just beyond halfway across the field. This kick became one of the main focuses within our strategy to score within the robocup competition.

- Torso Lean: 10 Degrees

- Leg Movements
- Backwards: 100 Degrees
- Forwards: 20 Degrees
- Sideways: 0 Degrees
- Height: 1cm
- Speed: 1.5 seconds
- Ankle Yaw and Pitch: 0 Degrees
- Range: 3.5m+
- Accuracy: Very straight and Accurate

10.5.2 Full length Kick

The full-length kick was designed as a kick to be utilised by a goal keeper, it provided the ability to clear the ball and move it into the oppositions half of the field effectively. This kick however was slightly unstable due to the amount of swing required to generate enough power to travel across the length of the field. This kick was also highly inaccurate as it was easily effected by uneven surfaces and tended to deviate from a straight path.

- Torso Lean: 17Degrees
- Leg Movements
- Backwards: 100 Degrees
- Forwards: 20 Degrees
- Sideways: 0 Degrees
- Height: 1cm

- Speed: 1 second
- Ankle Yaw and Pitch: 0 Degree
- Range: 6m
- Accuracy: Highly inaccurate and the kick is not straight.

10.5.3 Pass Kick

This is a slow very stable kick, its main purpose is to push the ball up into the oppositions half of the field but not loose sight of the ball. It requires little body angle and most of the momentum of the kick is sourced from the foot alone. This kick represented an easy option to past the ball up the field and it also allowed the robot to kick and chase the ball if required.

- Torso Lean: 5 Degrees
- Leg Movements
- Backwards: 100 Degrees
- Forwards: 25 Degrees
- Sideways: 0 Degree
- Height: 1cm
- Speed: 3 seconds
- Ankle Yaw and Pitch: 0 Degrees
- Range: 1.0-2.0m
- Accuracy: Low accuracy depends on where the ball makes contact with the foot.

10.5.4 Side Kick

This kick was also known as a kick off kick, it provided the robot with the ability to kick the ball to its side to avoid obstacles or to simply kick the ball off from a starting position. It had a short range but was a fast and stable kick.

- Torso Lean: 5 Degrees
- Leg Movements
- Backwards: 100 Degrees
- Forwards: 20 Degrees
- Sideways: 30 Degrees
- Height: 1cm
- Speed: 3 seconds
- Ankle Yaw and Pitch: 5 Degress
- Range: 0.5m
- Accuracy: Always kicks to the side of the robot.

10.5.5 Issues with the kick

The main issue we faced with the kick was the inherent stability of the kick, kicks that were slower and lower powerful were stable other kicks that were able to travel a long distance were less stable.

This was due to the momentum of the body robot body during a single support phase, lowering the torso angle and forward movement of the kick allowed for a more stable kick, however this had its disadvantages.

The elasticity of the carpet also presented an issue within the kick the shift to a single support phase wasnt possible on elastic carpets, this resulted in the robot kicking its foot into the ground and then falling over. In over to resolve this we had to tune the CoG of the robot however as we shifted the CoG more towards one side of the robot it becomes more unstable.

Overall we developed match-winning and goal scoring potential with the Nao, however due to issues within our behaviour module these kicks were never utilised in a real game situation.

Chapter 11

Conclusion

This thesis provides a foundation for locomotion on the new Nao platform. It assesses the strengths and issues with the new hardware, software, sensor configuration and locomotion within the domain of this new league.

It also presents optimised gait movements that were effective and utilised within the competitive environment of Robocup soccer.

Whilst the new platform appears to be encouraging progress in the area of biped locomotion, there are some major drawbacks with:

Hardware Fragility and poor manufacture of the robot, unreliable readings from foot force sensors, defective camera device

rUNSWift Software Complicated and incoherent code structure

Aldebaran Software Lack of an efficient and independent interface to the hardware appropriate for robocup, incompatibility between simulator and hardware, relying on code to which there is no access to the source

Whilst the Aldebaran provided software, NaoQi, had its drawbacks - it did provide the league with a kick start.

Major progress was made with:

Locomotion Derivation of a forward kinematic model, very thorough optimisation of walking parameters to gain maximum competitive advantage in walking speed and stability, insights into maintaining stability, understanding of gait generation for future development

rUNSWift Software Identification and proposal of a simplified code structure, development of a locomotion module

Areas identified for future research include:

- Omnidirectional gait engine
- Stability research - ZMP, using the IMU as a stability criteria, making use of FSRs
- Walk profiling / pace control
- Fall avoidance / reactions to sudden external forces
- Inverse kinematics implementation
- Dynamic balance control during kicking
- Sensor filtering

A thorough investigation and experimentation with the different aspects of the new platform has allowed this thesis to produce a bipedal robot that can admirably perform all the basic locomotion tasks required to be competitive in the Standard Platform League in 2008.

Bibliography

[1] Yaniv Altshuler Alfred Bruckstein. Introduction to robotics, lecture ii, 2008.

[2] B. Hengst et al. Omnidirectional locomotion for quadruped robots. 2002.

[3] Kemalettin Erbatur et al. *A Study on the Zero Moment Point Measurement for Biped Walking Robots*. PhD thesis, Faculty of Engineering and Natural Sciences, Sabanci University, 2002.

[4] Makoto Shimojo et al. A zmp sensor for a biped robot. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, 2006.

[5] Shuuji Kajita et al. Biped walking pattern generator allowing auxiliary zmp control. In *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006.

[6] Yongxue Zhang et al. A new method of desired gait synthesis in biped robot. In *Proceedings of the 3rd World Congress on Intelligent Control and Automation*, 2000.

[7] Richard Fitzpatrick. Centre of mass, university of texas physics lecture.

[8] Enyang Huang. *rUNSWift 2006 Behaviors & Vision Optimizations*. PhD thesis, School of Computer Science & Engineering, UNSW, 2006.

[9] InvenSense. *Integrated Dual-Axis Gyro IDG-300 Datasheet*.

[10] Kouhei Onishi Kenji Sorao, Toshiyuki Murakami. A unified approach to zmp and gravity center control in biped dynamic stable walking. Technical report, Department of Electrical Engineering, Keio University.

[11] Jon Kieffer. *A Path Following Algorithm for Manipulator Inverse Kinematics*. PhD thesis, Department of Mechanical Engineering, University of Illinois at Chicago, 1990.

[12] Michael Quinlan. *Machine Learning on AIBO Robots*. PhD thesis, School of Electrical Engineering and Computer Science, University of Newcastle, Australia, 2006.

[13] Reimund Renner and Sven Behnke. Instability detection and fall avoidance for a humanoid using attitude sensors and reflexes. Technical report, University of Freiburg, 2006.

[14] Aldebaran Robotics. *Nao Ware Documentation*.

[15] Oliver J. Woodman. An introduction to inertial navigation. Technical report, University of Cambridge, London, 2007.

[16] Tak Fai Yik. *Locomotion of Bipedal Humanoid Robots: Planning and Learning to Walk*. PhD thesis, School of Computer Science and Engineering, UNSW, 2007.