THE UNIVERSITY OF NEW SOUTH WALES
SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

Walking Nao
Omnidirectional Bipedal Locomotion

Aaron James Soon Beng Tay

Bachelor of Science
(Computer Science, Honours)

August 2009

Supervisor
A. Prof. Maurice Pagnucco

Assessor
Prof. Claude Sammut

Abstract

Bipedal locomotion is an area of robotics which has risen to the research and commercial spotlight where there is an increasing push for their integration within human society. However, in order for successful integration with society, there is a need for these robots to manoeuvre around obstacles in a stable and effective manner. One field of application is through an internationally recognised competition of robotic soccer - RoboCup. Soccer, like most interactive environments, is constantly changing, such that manoeuvrability and stability must be considered in order for a player to successfully approach, control and manoeuvre a ball around opponents to effectively dominate the match. This thesis documents the locomotion module of the rUNSWift architecture including the development stages of its first bipedal, omnidirectional walk.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

"By the year 2050, develop a team of fully autonomous humanoid robots that can win against the human world soccer champion team"

- Robocup Federation 1993

1.1 Robocup and rUNSWift

Robocup is an international competition amongst universities around the world with the aim of furthering research and development towards the ultimate goal - developing a winning team of humanoid soccer playing robots by the year 2050. In 2008, the Standard Platform League bid farewell to the Sony AIBO dogs and greeted a new robot, the Nao. With the introduction of the Nao, the league was faced with many new challenges, particularly in the field of bipedal locomotion. rUNSWift participated in the 2009 world competition held in Graz, Austria, and emerged with its first bipedal omnidirectional walk.

1.2 Motivation

The Standard Platform League is still in its infancy in terms of development and as such, these years provide ample opportunity for new avenues into bipedal locomotion:

- Many teams relied on the use of the provided 'Aldebaran Gait' which executes walks in the static form of 'forward', 'turn' and 'sideways'. These movements make manoeuvrability a difficult task as the transitions between movements are discontinuous and require the robot to 'stop' briefly, potentially losing the ability to reach the ball before an opposing robot.

- Last year, due to the wide use of the ‘Aldebaran Gait’, most games were relatively slow and evenly matched in terms of speed. Like most competitive sports however, speed is often the key to dominance and control over a game. As such, novel and faster methods of moving around the field were necessary.

- The move from quadrupeds to bipeds is a huge transition and the problem of stability and control becomes of dire importance. For every fall, there is a costly penalty of being sent to the sidelines or having to execute get up routines, both of which hinder the robot’s ability to play for a substantial amount of time.

- Finally, in order to play a game of soccer sufficiently, a player needs to be able to perform a variety of actions given different circumstances. For example, when an opposing goal is completely open, the player should generally kick the ball towards the goal as opposed to dribbling towards it, giving the opposition team time to defend. Thus, actions such as such as kicking must be available as opposed to just walking.

1.3 Objectives

Due to the issues highlighted previously, a set of objectives were devised in order to allow the Nao to engage in a game of soccer:

- **Manoeuvrability:** The gait must be omni-directional. That is, be able to move in any direction on a 2D plane.

- **Swiftness:** The gait must be faster than the provided ‘Aldebaran Gait’.

- **Stability:** The gait must also be stable and maintain balance throughout the game and be able to accommodate for external disturbances, such as pushes and uneven floors.

- **Adaptability:** There must exist the functionality to execute other actions (such as kicking) as opposed to just walking.

1.4 Outline

This thesis report documents the processes undertaken from idea formulation to design decisions and finally implementation results:

- **Background:** Introduces the NAO Robotic Platform covering the core hardware and software features of the Nao. The rUNSWift Robotic Software Architecture will also be introduced briefly.

- **Bipedal Locomotion:** Contains the core of the thesis report, beginning with an introduction to bipedal locomotion. The remainder of the chapter will discuss details of formulated ideas, design decisions, evaluations and extensive analysis of the implementations, including three major prototypes produced.

- **Special Actions:** Introduces a new module to the rUNSWift system responsible for the creation and execution of user created actions such as kicking, recovery routines and goalie guarding skills.

- **Results:** This chapter highlights the final results of the gait under controlled and competition conditions and discusses important lessons learnt from the experience.

- **Conclusion:** This chapter closes the thesis with concluding remarks regarding observations on the project offering potential future work.

Chapter 2

Background

2.1 NAO Robotic Platform

Adopted in 2008 as the new robot for the Robocup Standard Platform League, the Nao is a bipedal robot developed by Aldebaran Robotics.

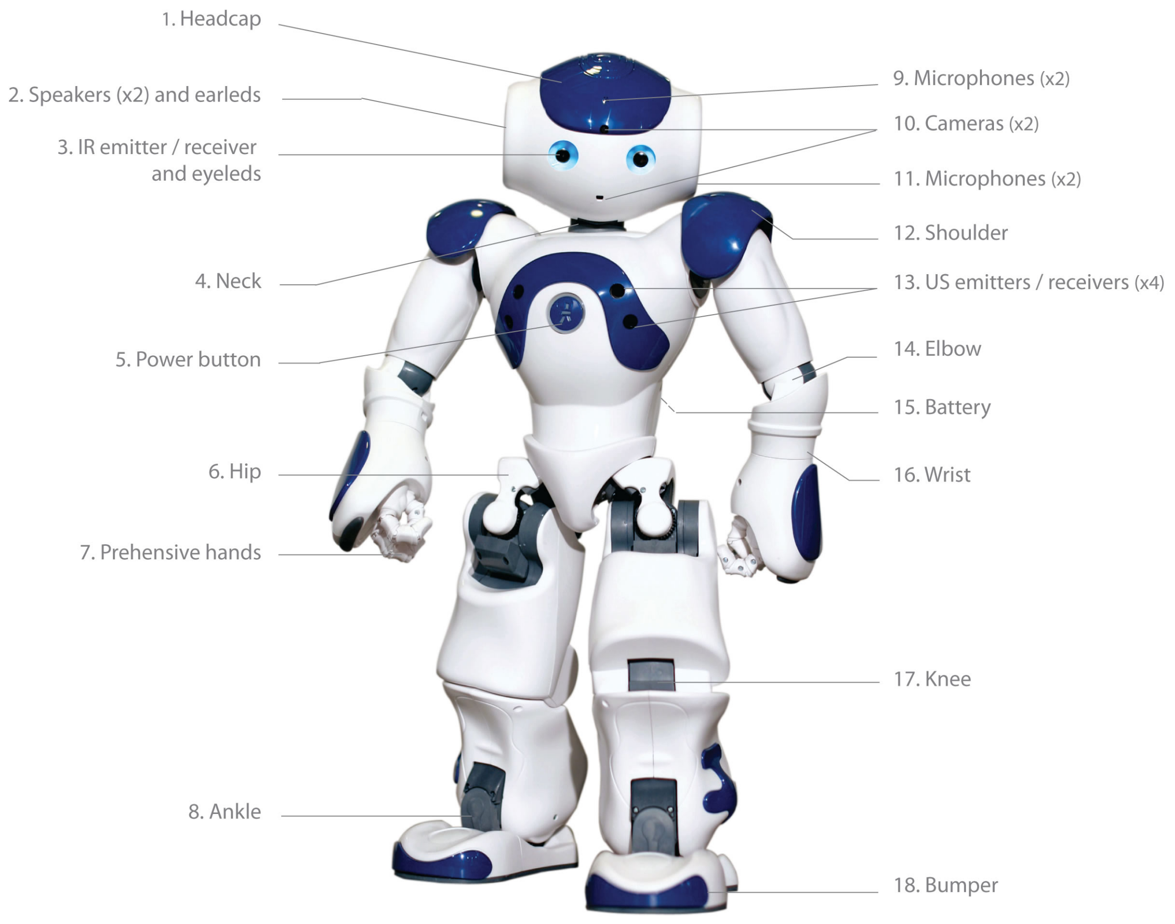


Figure 2.1: *NAO[5]*

2.1.1 Hardware

Standing on two legs, Nao has 21 degrees of freedom (DOF). The joint motors are allocated as per figure 2.2. One major difference which sets the Nao's joint configuration apart from many other bipedal robots is the unique design of the coupled, inclined rotary axis joint. Usage of this joint affects both legs, but enables the Nao to perform actions such as turning and angled kicks using one less joint compared to most bipedal robots[14]. However, this combined joint also has its drawbacks which will be explained in chapter 3.

In addition to the motor joints, Nao is equipped with an array of sensory devices including ultrasound sensors, an Inertial Measurement Unit (IMU) in the chest (containing a 3-axis accelerometer and a 2-axis gyro-meter), 4 Force Sensitive Resistors (FSR) each located in the corner of each foot, 2 cameras (on the face) and microphones. The head of the Nao is the home of its main CPU, the AMD Geode, with speakers located in the ears. Lastly, there is an abundance of Light Emitting Devices (LEDs) in the eyes, ears, chest and feet. The location of the devices can be seen in figure 2.1. In the area of bipedal locomotion and for the scope of this thesis, we are only concerned with the inertial measurement unit and force sensitive resistors.

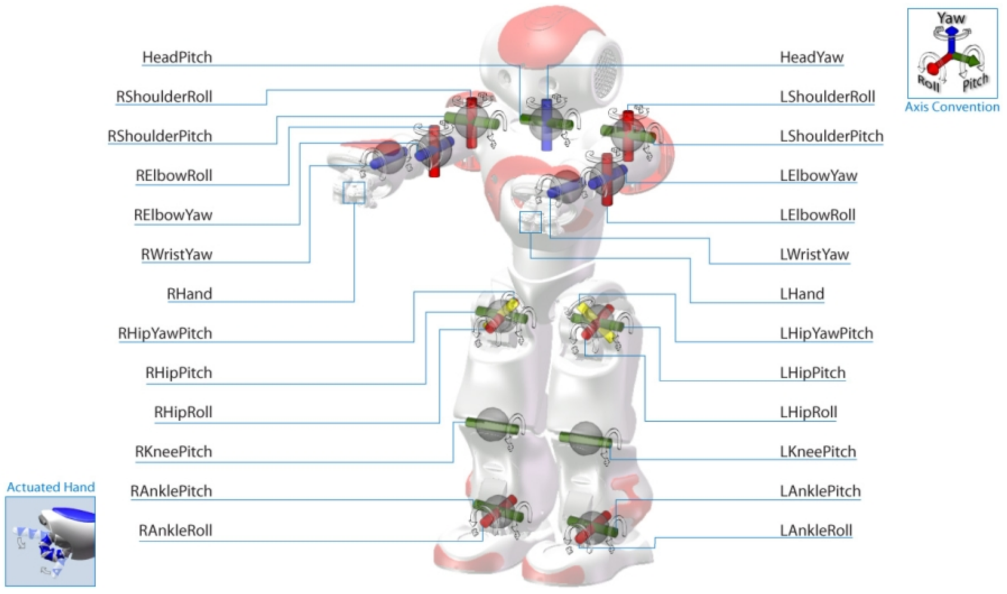


Figure 2.2: *Joint Configuration of the Nao (Robocup Edition had no actuated hand joints)[5]*

2.1.2 Software

In order to interface and make use of the hardware features provided by the Nao, Aldebaran Robotics developed NaoQi, a proprietary robotic middle-ware which interfaces with the Linux-Geode operating system available on the Nao. This middle-ware provides a safety layer between software executed on the robot interfacing with the motor joints in order to preserve their longevity. NaoQi also is equipped with a variety of preloaded modules to perform tasks such as motion actions (`ALMotion`), extract images from the camera, text-to-speech etc... Each module interacts with what is known as the *Main Broker* which sits on top of the operating system. Modules are then loaded and connected to the main broker in order to gain access to the hardware as well as access to other modules[5]. The rUNSWift system is designed to be a module which connects to the main broker.

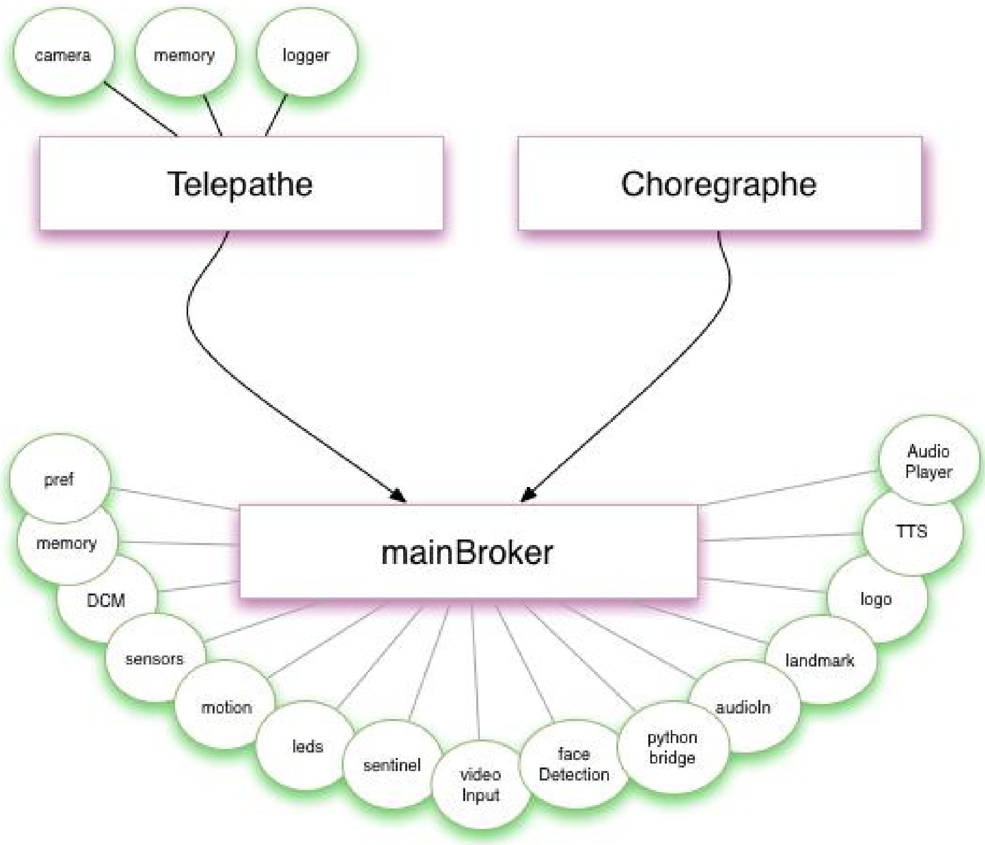


Figure 2.3: *NaoQi Module Interaction[5]. Refer to appendix B.3.1 for Choreographe and Telepahe*

2.2 rUNSWift Architecture

Although the general rUNSWift robotic software architecture has maintained a similar structure to last year, the modules responsible for interfacing with and management of the hardware of the robot has undergone a complete rework. Details are beyond the scope of the thesis, for more details refer to [28]. Figure 2.4 shows the overall architecture and how locomotion relates to the entire system. The two modules of interest are Actuation and Sensory. The core of this thesis relates to the Locomotion component of the Actuation Module.

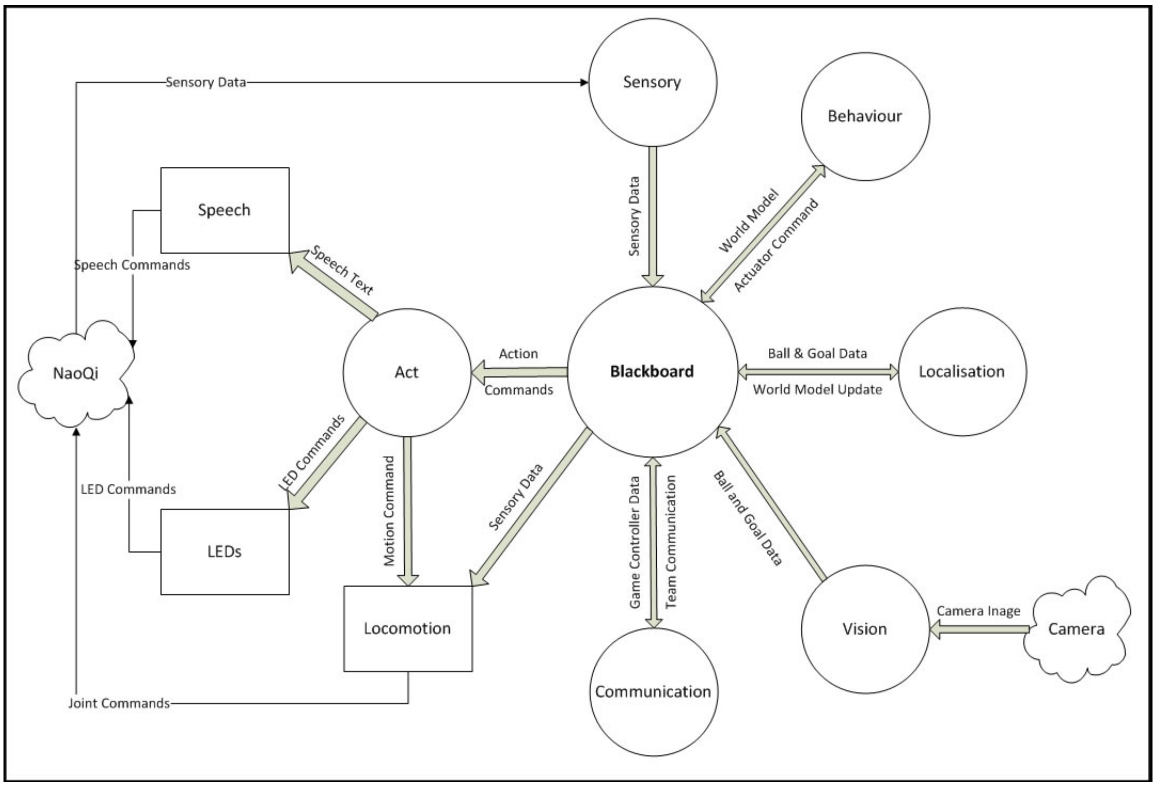


Figure 2.4: *rUNSWift Architecture. Circles represent main modules. Rectangles denote components managed by a module.*

2.2.1 Actuation Module

The Actuation module is responsible for robot actions in response to commands issued by the Behaviours. It is comprised of three main components; Locomotion, LEDs and Speech. The latter two are beyond the scope of this thesis and the reader is referred to the NaoWare Documentation[5] as usage is similar. The Locomotion component itself is comprised of two parts, `GaitEngine` and `MotionModule`. The `GaitEngine` is responsible for calculating and generating trajectories of the walk based on the motion commands. Once the trajectories for the gait have been determined, they are then committed to the `MotionModule`, which handles the joint requests through the interaction with NaoQi via the `ALMotion` proxy. The Locomotion component manages the interpolation between walks and special actions, explained in chapter 4.

2.2.2 Sensory Module

The purpose of the Sensory Module is to extract and calibrate sensory data from the IMU and FSRs. One technique adopted was the use of a specialise memory proxy, `ALMemoryFastAccess` which was made available specifically for RoboCup in 2009 by Aldebaran Robotics. This proxy allowed unrestricted access to all low-level sensors as detailed in the DCM documentation**??**, as well as optimised retrieval of values stored in NaoQi memory. Calibration of sensory data in this module prior to their exposure and usage creates a system which is robot independent as there are slight variances in the raw sensor readings, particularly the FSRs.

Chapter 3

Bipedal Locomotion

3.1 Background

Bipedal locomotion is far from a trivial challenge. Various factors ranging from core mechanical design of the robot to the more complicated system dynamics affect how to get a robot to execute its first step. In most research areas, the robot is designed from the bottom up with design decisions relating to the application of the robot taken into consideration when developing the mechanical structure of the robot[18][8][1]. Unfortunately in the standard platform league, the Nao's hardware is unable to be modified and as such, understanding and exploiting the robot's dynamics are necessary prior to the commencement of developing a dynamically walking gait.

3.1.1 Walking Classes

Within the field of locomotion, there has been much research undertaken relating to the different types of walking robots[17]. Most walking robots can be classified into one of three classes: Static, Dynamic and Passive. Static walkers have existed in the robotics field for quite some time and most of the early bipedal robots were static walkers. These robots have the property of always being stable throughout their walks. Generally, these walkers are very slow in order for inertia forces to be negligible. Passive walkers on the other hand are systems which require minimal or no active forces to react. Such systems arise from the mechanical design of the robots themselves as opposed to special software solutions and react only to the forces of nature such as gravity[26][19]. Finally, dynamic walkers are a class of walking robots between the two. These walkers plan, account for and exploit the forces of gravity and its dynamics in order to achieve a walking gait[13][3][16]. Most of these robots plan their footsteps with high precision to achieve a type of 'controlled instability'. This means that the system, although in a period of instability (the centre of mass outside the support polygon), is still able to continue walking by planning future actions to prevent the system from violating what is known as the Zero Moment Point (ZMP) requirement[29].

3.1.2 Trajectory Generation Techniques

There are numerous ways to generate trajectories in order to enable a robot to walk and various research has gone into bipedal locomotion with the aim of achieving 'human-like' motion[10][19]. There are two ways of generating trajectories; on-line and offline. Offline generation (also known as open-loop) is where an entire sequence of actions is planned prior to execution. The sequence of actions can range from a single step to an entire walking motion like the Aldebaran Gait[14]. One disadvantage with offline trajectory generation is that any disturbances to the system cannot be accounted for or reacted to immediately. On the other hand, systems which utilise on-line generation (closed-loop) incorporates a feedback controller to accommodate for its environment. These system are usually more robust to disturbances but have the added complexity of potentially heavy computation. As such, most static walkers are open-looped and dynamic walkers closed-looped. Examples of on-line generation techniques include Central Pattern Generators[4] where a system of neuron-like structures are interconnected through, and react, to inhibitory and exhibitory states[21]. Others include a typical feedback (and/or feed-forward) control loop where a system has to fulfil certain conditions such as the Zero Moment Point[1][2].

3.1.3 Walk Phases and Patterns

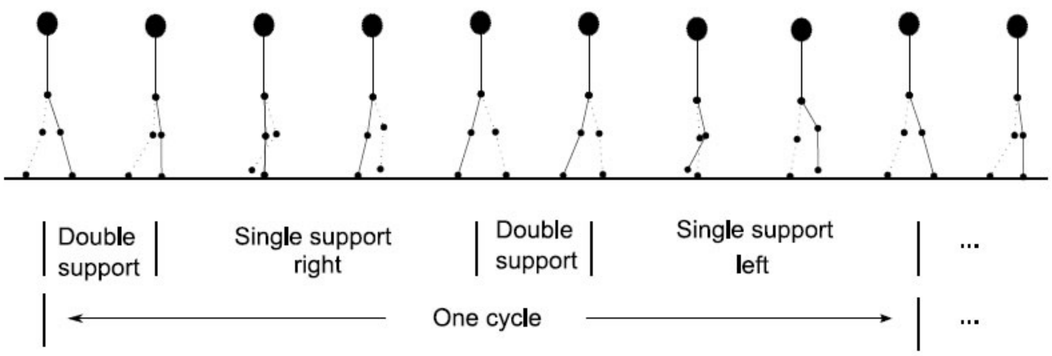


Figure 3.1: *Phases of a Walk[31]*

There exists a variety of literature regarding the various types of phases experienced throughout a walk[29][20]. However, at its core, walking consists of two distinct phases; the single support phase and the double support phase (running is a separate problem and, by definition, consists of a flight phase). Before the commencement of a walk, most bipedal systems begin in the double support phase, where both feet are in contact with the ground. They then transition into the single support phase which consists of one leg in contact with the ground (support leg) and the other in the air (mobile leg). The mobile leg is then moved to the desired location on the ground and the cycle repeats as both feet are now in contact with the ground (Figure 3.1). A common problem encountered throughout this cycle during the single support phase is a highly unbalanced system due to a small support polygon, inertial forces of the mobile leg, momentum from prior movements etc... [30] In attempts to control the unstable nature of the bipedal robots during the single support phase, there has been recent formulations of other phases such as shifting (the moving of double to single), raising (bringing the mobile leg into the air) and lowering (bringing the mobile leg back to the ground) in order to allow more fine grain solutions.

The types of walks determine the manoeuvrability of the system. A unidirectional walk (like the Aldebaran Gait) allows moving in only one direction at any point in time. Where as an omnidirectional walk is able to combine movements in various directions to achieve a more complicated motion.

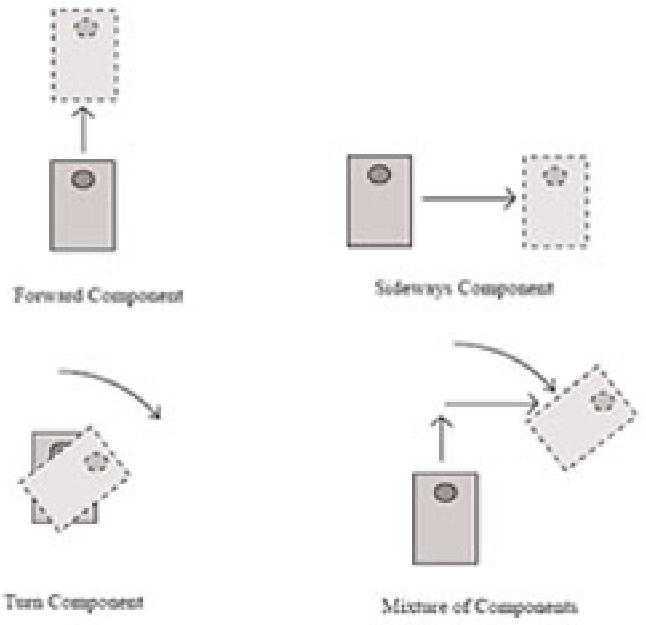


Figure 3.2: *Walk Patterns[7]*

3.2 Locomotive Requirements

Upon commencement of the project, the main outcome was an omnidirectional, dynamic, closed-loop gait for the Nao Robot. In order to reach this objective, a series of milestones were devised:

- Ability to manipulate the robot like a remote controlled toy to walk around the field (manoeuvrability requirement).
- Resultant walks must be faster than provided Aldebaran Gait (swiftness requirement).
- Feedback control to maintain stability of the upper torso and the movement trajectory of the Nao robot (stability requirement).
- Automated calibration methods of robots including joint angles and sensory data.
- Parametrised walk for future optimisation.

However, due to circumstances such as malfunctioning robots, changing priorities and work allocation and time restrictions, the requirements were adjusted slightly over time as the competition deadline loomed ahead. Thus, for competition purposes, the objectives were redefined, to exclude the requirement of a feedback loop for torso stability to:

- Ability to control the robot like a remote controlled toy to walk around the field (manoeuvrability requirement).
- Resultant walks must be faster than provided Aldebaran Gait (swiftness requirement).
- Minor feedback control to maintain movement trajectory of the Nao robot (stability requirement).
- Static open-looped gait.
- Prevent harmonic build-up (see 3.3.3).
- Limited Parametrised walk.

3.3 Development Phases

3.3.1 Initial Prototype

The development of the initial prototype model was carried on from work conducted over the summer[28]. The model's inverse kinematics system was defined and theoretical basis for the gait engine designed. This model latter formed the basis of what is known as the *Body Coordinate System*. Using this model, the leg end effectors (feet) were able to be manipulated in 3D space. Some restrictions and assumptions were introduced in order to simplify the model:

- The ankles of the robot were always adjusted to be parallel to the ground plane. This assumption ensured a flat region of contact with the ground at any point in time.
- The robot's stability is ignored when moving end effectors in order to make the system more predictable when changing joint angles. The initial proposal was to develop a kinematic model capable of manipulating the robot such that a higher level stability control system would use the model to maintain its stability.
- The provided documentation and specifications of the NAO were assumed correct (later to be proved otherwise) and thus a point on the robot which maximised repeat calculations was chosen; the centre of the robot at the intersection of the two legs, where the hipYawPitch joint is located.
- Arm kinematics were ignored at this stage of development.

Adjusting the feet

There exists a variety of step trajectory generation methods such as [2][12][19]. At this stage of the project, the primary focus was getting a simple onmidirectional gait to be utilised by the entire robotic system. As such, approaches more suited for static walkers were adopted, resulting in offline trajectory generation[6].

In this cycle of the development process, the initial formulation of the robot representation for parametrisation was introduced. These included the adjustment of footsteps and the body stance. As mentioned previously, there are a variety of factors which can affect the trajectory of a step. In 2000, UNSW United introduced the *'Parametrised Walk'* for the league's former platform, the AIBOs, which revolutionised the locomotive capabilities of the league[15]. At this stage of development however, only two parameters were deemed vital to the robot's ability to walk with the remainder kept as static assumptions. Two parameters which affect the magnitude of an individual step are its height and length. The Bezier curve approach was adopted[28], and an assumption was made where the two air control points remained fixed in relation to the step height and length. This maintained the half-ellipse shape produced by the walk as shown in Figure 3.3.

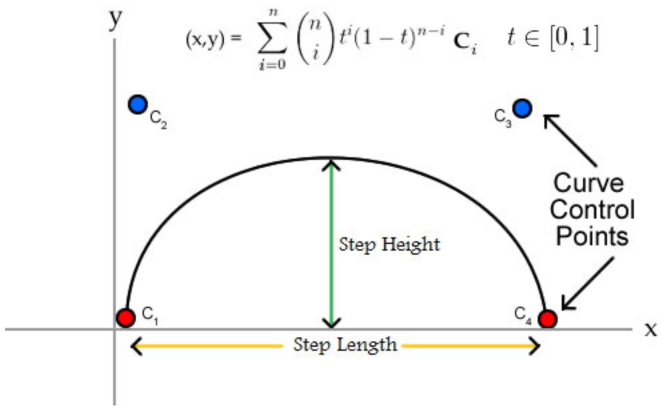

$(x,y) = \sum_{i=0}^{n} \binom{n}{i} t^i (1-t)^{n-i} \mathbf{C}_i \quad t \in [0,1]$
y
x
C_1
C_2
C_3
C_4
Curve Control Points
Step Height
Step Length

Figure 3.3: *Footstep Trajectory*

The body stance is another aspect of the system which plays an important role on the walk, and relates to how the body is positioned during the walk. These included the height of the robot's centre from the ground and angle of the upper body (relative to the upright position in both the x and y axis). These variables all affect the position of the robot's centre of mass. Changing the angle of the upper body effectively shifts the robot's centre of mass in the x and/or y axis. However as the robot's upper body is angled in the specified direction, it indirectly adds a 'falling effect' which can be likened to passive walkers. A property of static walkers is that due to their low height and slow speeds, such effects are almost negligible, although attempts have been made to explore and exploit some of these dynamics in later prototypes. Following from the findings of last year's competition[9], we aimed for a statically stable walk with a low height.

That First Step

By far the most difficult challenge was making the first step with this gait. Indeed, for static walkers, this can only be achieved if the robot system is balanced and the centre of mass is within the support polygon of the supporting leg. The next issue was moving the robot's centre of mass in such as way to allow sufficient time for the swing leg to be propelled forward, come in contact with the ground, and to move into the transition phase, thus producing the walk. In order to achieve this, a periodic sinusoidal function was induced to change the position of the robot's centre of mass in the coronal plane. The use of a periodic function allows us to scale and adjust the amplitude and period to achieve the desired effect as in figure 3.4.

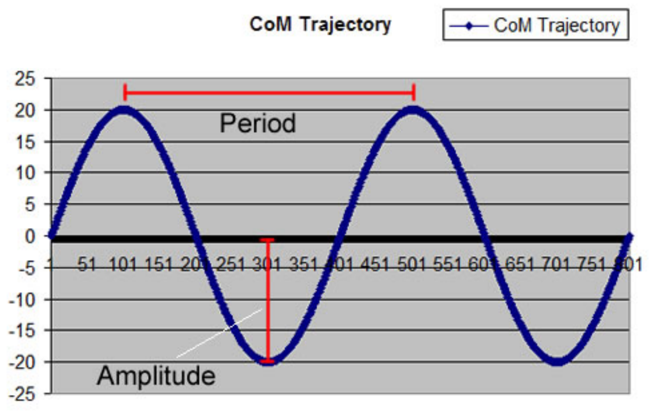


Figure 3.4: *Variables affecting the trajectory of the centre of mass*

Whilst swaying onto the support leg, the swing leg's foot was lifted up for brief moments in order to achieve a mark-time motion. The purpose of this was to tune the amplitude and period such that sufficient time was given for the swing foot to be lifted and replaced. Once this was achieved, the swing foot's position in the sagittal plane was moved forward slightly resulting in a successful first step. However, this did not result in a walking motion as the robot's centre of mass needed to travel in the same direction in order to 'move' the robot completely. Thus, the centre was moved an equal distance forwards, immediately after the swing foot has made contact with the ground. The resulting action when repeated continuously resulted in the first steps towards a new gait.

The low step height of the walk meant its feet were likely to remain in contact with the ground. Whilst this actually hindered the walking capabilities of the robot it allowed us to test concepts, maintain stability, and more importantly, begin integration with the whole architecture.

At this stage, the rUNSWift behaviours were interacting and using the provided Aldebaran Gait in order to achieve their purpose. However, in preparation for the new gait, minor adjustments were required to the interface between behaviours and locomotion. This involved the dynamic transitions between Aldebaran's gait and the new omnidirectional gait to achieve the best of both worlds;

- Aldebaran's Gait was substantially faster than the omnidirectional gait developed at this stage peaking at around 10cm/s as optimised from the previous year's competition. This speed was essential to approaching the ball before opponents.

- The omnidirectional gait, despite its slow speed (approx. 3cm/s), allowed behaviours to execute finely tuned ball aligning skills in order to position ourselves for an offensive attack as it required no explicit transitioning between changes in direction.
- One drawback from this approach was that transitions between Aldebaran's Gait and the omnidirectional gait still required the Aldebaran gait to cleanly finish its footsteps.

Overview

Although just an initial prototype, the resultant gait showed some potential that the pursuit of an omnidirectional walking is a feasible venture. Below are some of the highlights of the achievements.

Pros:

- Omnidirectional walk.
- Dynamic switching between Aldebaran Gait and omnidirectional gait to achieve the *'best of both worlds'*
- Extremely stable and resilient to external forces.

Cons:

- Omnidirectional walk was too slow.
- Transitioning between the two gaits required some overheads.
- Slightly cumbersome for behaviours.
- Wide centre of mass and swaying caused strain on hip joints.
- Low centre of mass caused strain on ankle joints.

3.3.2 Mid-Semester Milestone

The 'Mid-Semester Milestone' required collection, interpretation and use of sensory data from the robot as the intended outcome from this stage was a system which was able to react to periods of instability.

The robot's centre of mass was increase slightly to reduce the load on the hip and knee joints. However, this increase meant the system would be more unstable, facilitating our need for a stability controller. As such, a relatively low step height was used to compensate for this, effectively sliding on the ground. However, it was found that this behaviour resulted in occasional drifting (mainly veering to the right) of the robot from its desired path. As such, compensation methods were adopted to correct this drifting, which involved the addition of a minor turn in the opposite direction. The resultant walk was found to follow the desired walking direction more accurately but if the robot was left to walk forwards the entire length of the field, the effects of the drifting become apparent.

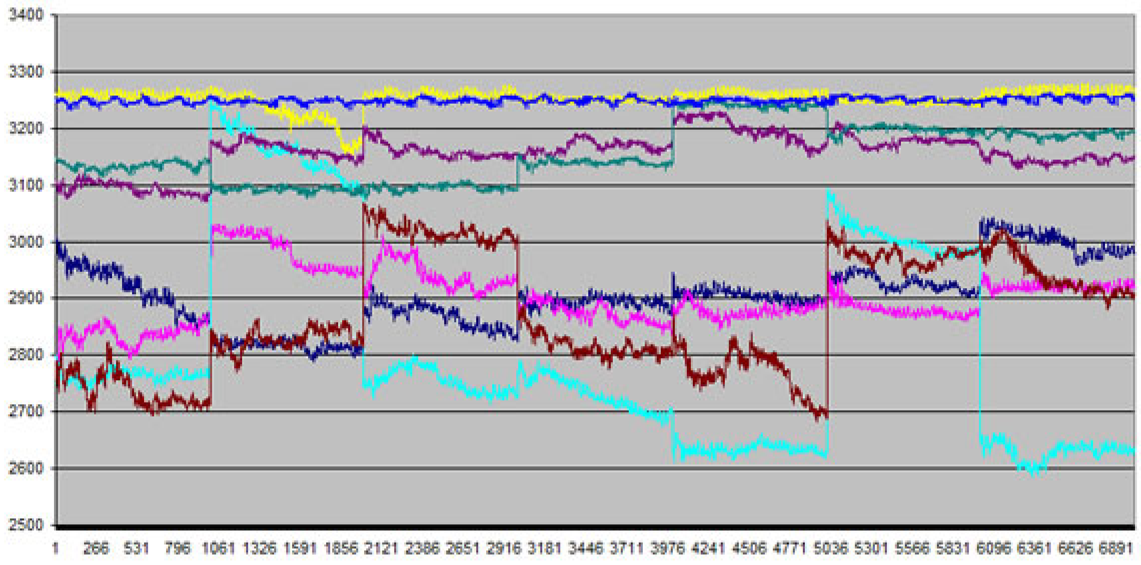
Another minor issue encountered was that due to the low step height, the robot would trip over the field line tape. In most cases, the robot would only jerk slightly upon contact with the lines, only to recover shortly afterwards. In the small other cases, the system would become unstable and thus, precautionary methods needed to be taken.

Nao Sensing Falls

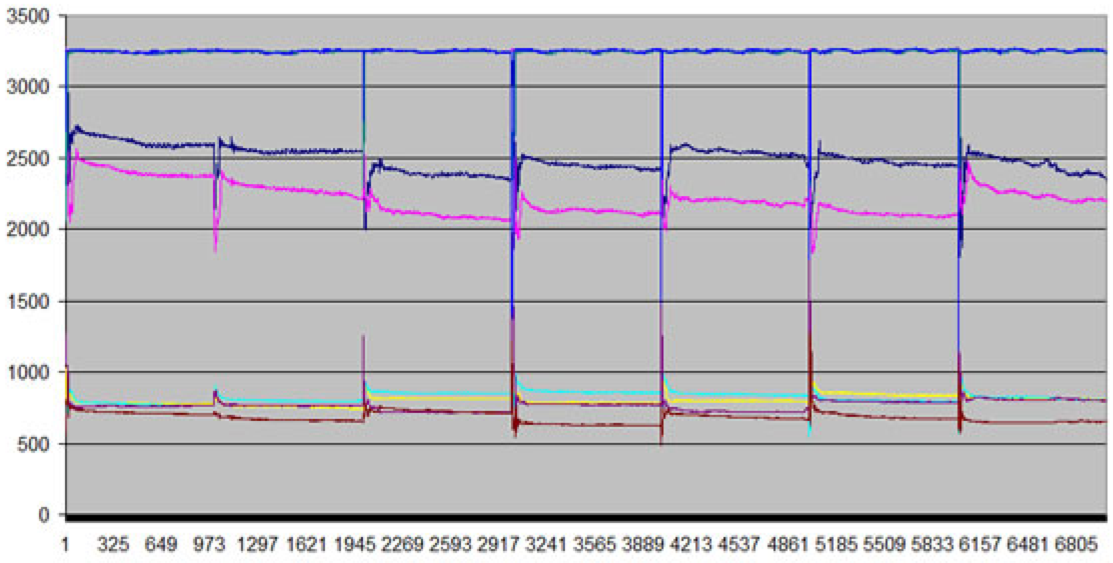
Data collection was carried out through the extraction of unfiltered IMU and FSR data of the robot whilst walking (under normal game conditions), standing still and whilst in the air. This allowed for a variety of data to be collected under numerous circumstances with the cases of standing still and in the air as reference points. In addition, the extraction process was carried out on all robots in order to determine the reliability of the sensors.

An interesting encounter worth mentioning is the reliability of the FSRs. In most cases, they seemed to be working properly, however there have been occasions where a single FSR on a foot fails to register, resulting in *'no force'* values. That is, there exists the possibility of only three of the four FSRs on a foot worked correctly. Usually a reboot (or two) of the robot rectified the sensors, however methods have been developed to estimate the ZMP using only three FSRs[23].

Such an occurrence did not occur with the IMU sensor and as such, they were the first of the sensors to be trialed for fall prevention. In this phase of testing, the accelerometers were trialled for their resolution (rate of update), because if the sensors are too slow to react, it may only be possible to detect falls after they occur.



(a) *FSR with robot in air*



(b) *FSR with robot in home*

Figure 3.5: *Force Sensitive Resistor Data. The blue line (front left FSR on the left foot) in both air and ground cases registered 'no force' for the duration of the test. In all other cases, there is a distinction between 'on the ground' and 'in the air'.*

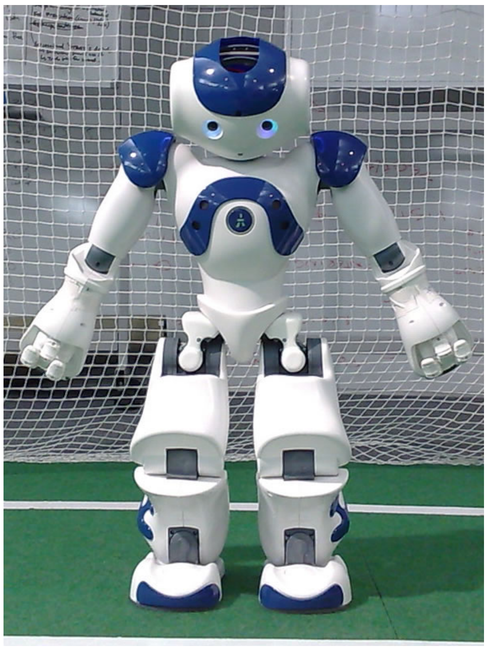


Figure 3.6: *Nao's Home (Initial) Position*

The robot was placed in the home position (figure 3.6) and a force (manual pushing) was induced from various directions in order to make the robot unstable. Through this, a rough range of values were devised to determine to overall stability of the system. The system was deemed *stable balanced*(SB) if pushing it results in both feet remaining in clean contact with the ground. The *unstable balanced*(USB) state is the case when after a push, a robot's foot leaves the ground momentarily, but the system returns into the stable balanced state immediately. Finally, *unstable unbalanced*(USUB) refers to when the robot's foot leaves the ground and a rocking motion is experienced before the robot eventually returns to the stable state. Beyond these three regions, and the robot falls over.

When displayed diagrammatically, as in Figure 3.7, it is clear that there is high level of tolerance in the forwards direction in relation to the robot's home position. This may be attributed to the location of the robot's centre of mass leaning slightly towards the ankles. As such, it is more difficult to fall forwards than in other direction, which support observed results.

System Stability	Accel. X	Accel. Y	Accel. Z
Stable Balanced	[-5, 10]	[-10, 10]	[-60, -55]
Unstable Balanced	[-10, 25]	[-15, 15]	[-75, -50]
Unstable Unbalanced	[-12, 30]	[-20, 20]	[-100, -45]

Table 3.1: *System Stability based on Accelerometer*

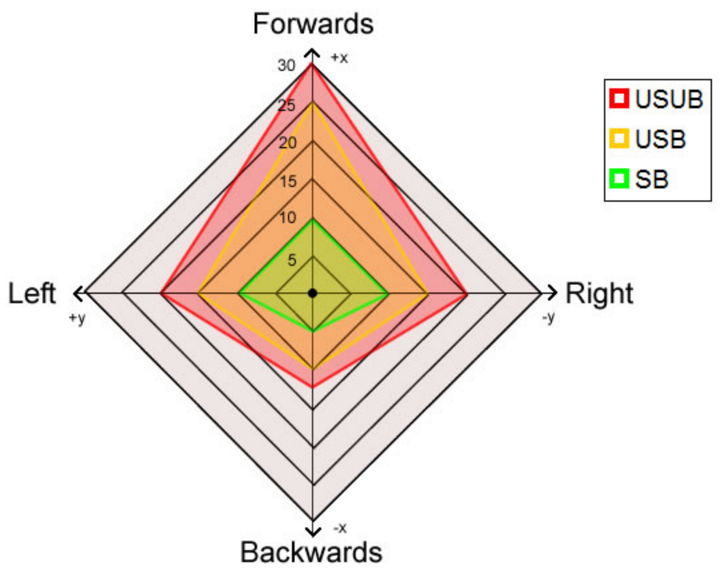
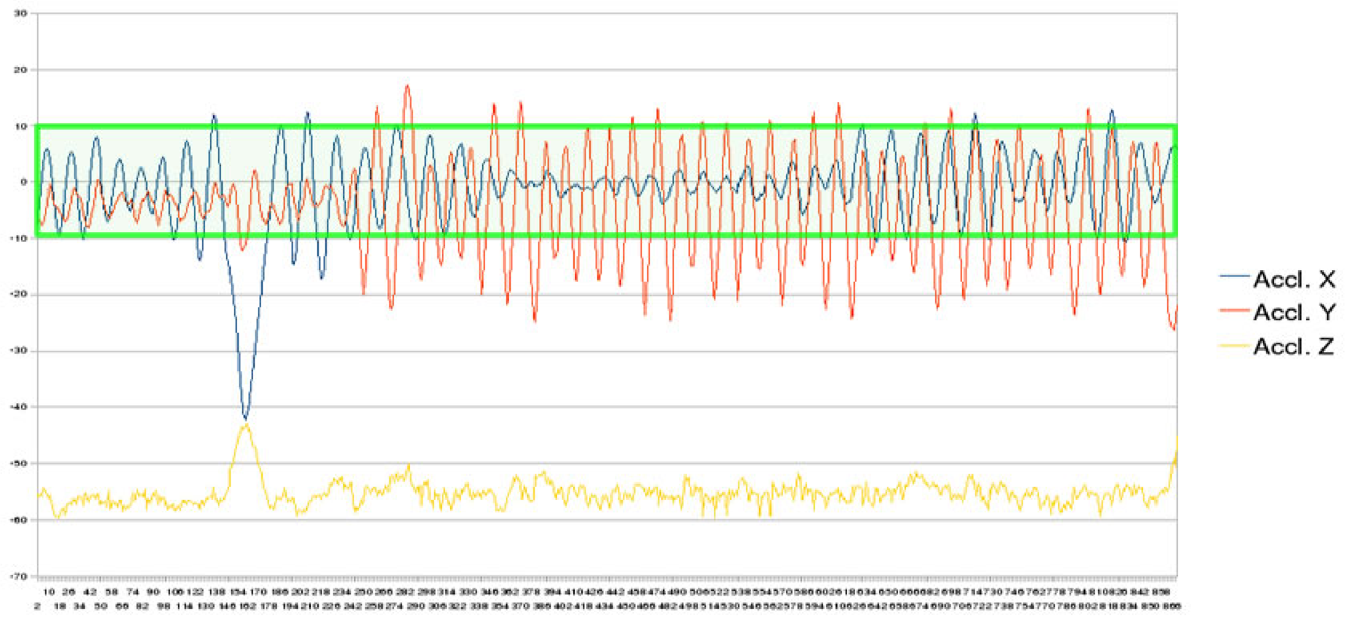
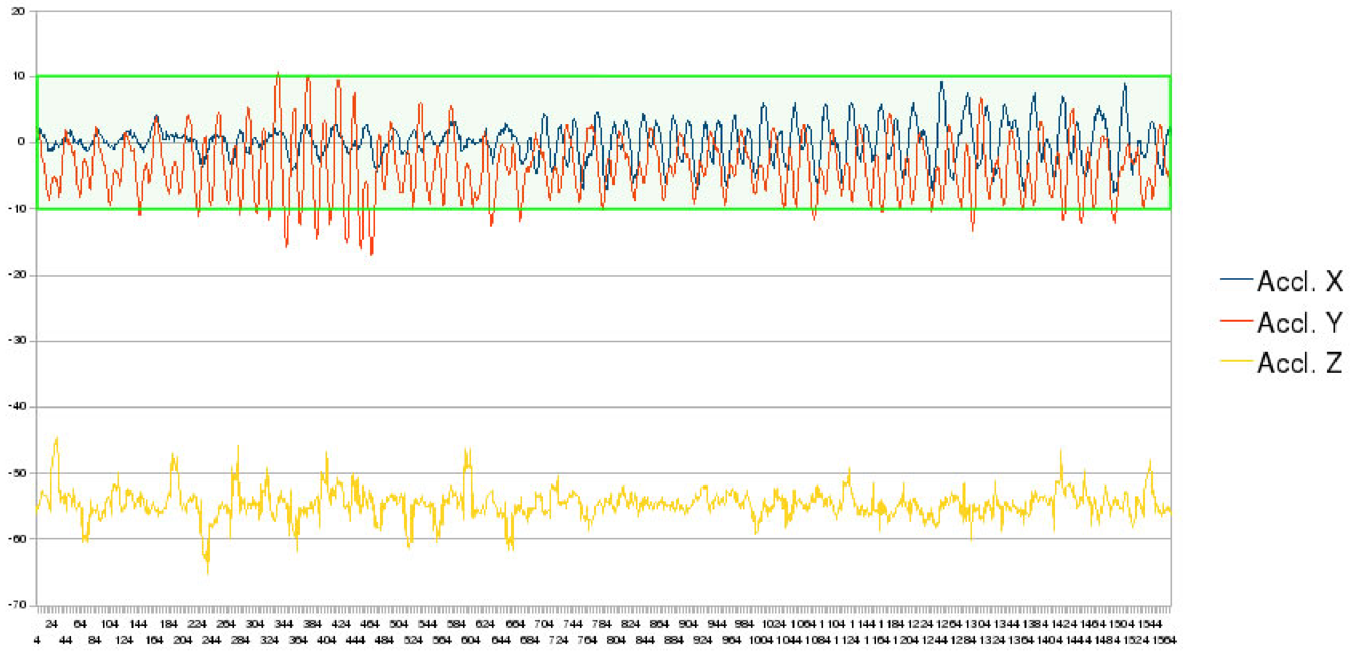


Figure 3.7: *Stability Regions from Accelerometer in X-Y Axis*

Using this gathered data and regions, the next phase was to develop some policy for the robot in order to react appropriately, particularly when in the USUB state. In the USB state, the robot's severity of the instability is classified as mild and as such, only minor adjustments are required. These included changes to the upper torso angle. For example, if the robot experience a high acceleration backwards despite moving forwards, it is highly probably that the robot is reaching an unstable point and is beginning to fall backwards. As such, tilting the upper torso forwards will hopefully counter this acceleration and re-stabilise the system. In addition to the actions encoded for the mild case, a more drastic cause of action was developed when the robot reaches the USUB state (severe). Here, simply tilting the torso will indeed reduce the acceleration, but will most likely be unable to stabilise the system completely. Thus, modifiers were introduced to the robot's gait to change its motion (whilst attempting to maintain commands from the strategy level).



(a) *Walking without adjustments*



(b) *Walking with adjustments*

Figure 3.8: *Stability while walking in accordance to our experiments. The green region denotes the range which was declared as 'stable balanced' for the Y-axis of the accelerometer (as majority of falls were too the sides). 3.8a actually experienced a period of what is known as 'harmonic build-up', up to sharp spike, after which the system reached its 'natural rhythm' and began to sway sideways in a stable, uncontrolled manner. However, once trapped there was no way to control this behaviour and as such, precautionary actions were required to prevent this, resulting in 3.8b*

A robot falling results in high acceleration in the falling direction (as can be seen in the 3.8a). In order to counter this, the walk was modified the walk in a number of ways:

- **Ignore:** Continue to walk hoping for the system to re-stabilise eventually.
- **Abuse:** The fall direction is the same as the walk direction, increase the speed (step period) of the walk to gain an 'extra' speed up, slowing down shortly afterwards.
- **Resist:** Similar to the abuse approach, but move immediately in the opposite direction to the fall direction.
- **Accept:** Attempt to change the walk direction slightly by moving towards the direction of the fall (in some proportion).

The accept approach was adopted as it caused the least disruption to the entire system. The ignore policy compromised stability, the abuse policy only works if the fall direction is the same and the resist policy was found to be too rash. The direction of the fall was calculated via the `atan2(y,x)` of the accelerometer values and its magnitude obtained through a difference in the acceleration from the home position with a scaling factor of 0.75, derived empirically. This vector is then combined with the current walk command from behaviours to result in a new walk pattern. When the fall is in the same direction, the robot's walking speed increases slightly to abuse the effect. On the other hand, if the fall direction is opposite to the robot's walk direction, it will slow down and attempt to walk in the direction of the fall until the system is deemed balanced. This behaviour is particularly evident in the forward/backwards motion.

Our Gait

Although slightly slower than the standard Aldebaran gait, the development of behaviours utilising the two gaits seemed to be causing unnecessary complexities at the strategy level, particularly in the transitioning between the walks. Coupled with the similar speed obtained from an omnidirectional walk, the dynamic switching between the two was phased out in this stage of development.

There were some slight drawbacks of completely changing over to the use of the developed gait. These included the slow sideways and turning motions. The slow sideways motion can be explained by the wide centre of mass trajectory. This lead to a system which swayed side to side heavily and coupled with a low body height, the legs had hardly any room to move in the coronal plane. Turning had a similar reasoning, but the main problem was incorrect calculations of the hipYawPitch joint angle. This caused the robot's upper body to lean forwards too much and thus tilt its centre of mass to the edge of the support polygon making it highly unstable. Thus, initial compensation methods were to reduce the angle of the turn. Both issues was addressed in the final version of the gait.

Overview

This stage of the project served as a checkpoint for progress and to allow for any adjustments to the scope of the project given the remaining time and current rate of progress. By this stage, the gait was capable which was capable of walking faster and at a rate almost comparable to the Aldebaran Gait with the addition of some fall avoidance. Below are some of the highlights:

Pros:

- Substantially faster than the initial prototype (approx. 8cm/s).
- Compensation methods for drifting.
- Interpretation of accelerometer sensory data for fall preventative measures.

Cons:

- Low step height caused some slippage and tripping especially on field line tape.
- Low step height lead to drifting (but was accounted for).
- When caught in its natural rhythm, the robot could be stuck in periods of ‘falling’, resulting in the robot rocking side-to-side whilst moving backwards in a type of ‘stable uncontrolled’ state.
- Slow in-place turns and sideways motion.

3.3.3 Final Implementation

Prior to the competition, a final version of the gait was developed. This involved experimentation with arm movements, a higher centre of mass and step height and more interpretations of sensory data. Some of the core kinematics were adjusted to accommodate for the asymmetric leg lengths and to improve the in place turning speed. Unfortunately, the initial aims of this thesis were unable to be met and thus were adjusted slightly to suit the current pace of work.

Arms doing the walking

Idea of using arms in walking came about through the observations of humans walking. The opposite arm moves in synchronisation with the mobile leg as a way of counter balancing the system. Initial experimentation with the static motion of arms on the Nao have shown that their motion has some effect on the stability of the system. This involved the generic swinging of the arms back and forth similar to a military march. The swinging arm would move in unison with the mobile leg such that the arm will be fully extended forwards when the leg strikes the ground. The support arm moves in the opposite direction (i.e. backwards). This approach was intended to explore the use of arms, with the length of extensions of the arms derived empirically through trial and error.

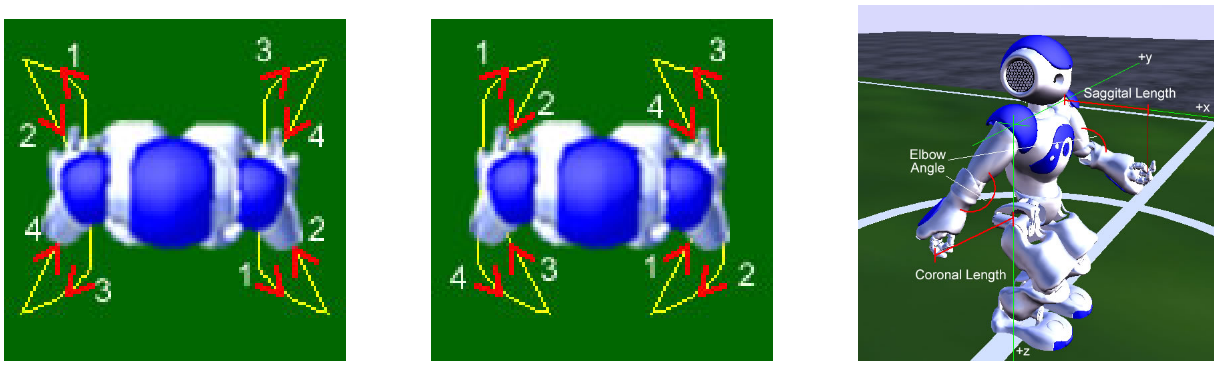
Building upon the previous walk trajectory, the aim to use the arms to account for some of the drifting of the robot by using the arms as extra weights. Two types of motions were developed; one where the arms circle outwards, and the other inwards. The former provided some evidence that the use of arms can affect the overall stability of the system. When the arms followed the trajectory as in Figure 3.9a, a type of outward rotational force was induced, causing the robot to become more unstable, possibly due to the robot's centre of mass being extended beyond the support polygon due to the arms. In contrast to this, when arms were circled inwards as in Figure 3.9b, the robot was more stable due to inward rotational forces.

The trajectory of the arms were directly related to the speed of the walk - faster walks lead to faster arm motions (figure 3.9c). Although the arms were statically moved in unison with the legs, the shape of the arm trajectories were encoded into the `GaitEngine` and controlled by adjusting various parameters based on the placement of the hands:

- **Sagittal lengths:** How far forward/backwards to swing the arms (length in sagittal plane).
- **Coronal lengths:** How far inward/outwards to move the arms (length in coronal plane).
- **Bend/Stretch angles:** Elbow angles at extremities.

Accounting for Discrepancies

Throughout the duration of the project, the special actions module was being developed (see chapter 4). It was during its development that the asymmetric nature of the robot's limbs was discovered. There exists, approximately, a 10mm discrepancy between the left and right thigh lengths. This is evident when the robot was walking forwards as there is a slight 'falling'



(a) *Outward Trajectory* **(b)** *Inward Trajectory* **(c)** *Arm Parameters*

Figure 3.9: *Arm Trajectories*

motion when the robot was landing on its right leg. As such, the kinematic model needed to be adjusted to account for this small discrepancy and assumed the right leg was shorter than the left (thereby increasing the left leg's length to 105mm).

In the world of static walkers, everything must be accounted for exactly for the system to perform 'perfectly' as this type of walker is unable to adjust adapt to its environment. Dynamic walkers on the other hand, would not find this minor discrepancy a problem as they can accommodate for the slight changes throughout the walk, which may be interpreted by their feedback controller.

Going up

The robot's centre of mass was raised to a height of 175mm from the ground (the maximum possible is 200mm), which was much higher than the initial prototype and Aldebaran Gait (150mm). This increase meant as a dynamic system, the robot is more unbalanced due to inverted pendulum dynamics. This was particular in motion in the coronal plane due to the side-to-side swaying motion which lead to a harmonic build up due to the use of the sinusoidal trajectory to induce the sway. In order to counter this, we reduced the amplitude and flattened the trajectory to produce a flat sinusoidal function (figure 3.10). This essentially had the effect of pausing the robot's sway for a brief period (40ms) to remove any excess momentum force, but resulted in a less fluid walk.

The next height increase was the step height. In order to reduce the effects slippage, the foot needed ample clearance from the ground. However, the drawback of doing so means the system is even more unbalanced as the robot's support polygon is reduced to the single support phase's for an extended period of time. This can be accounted for by reducing the period of the sway trajectory, that is, by quickly moving the centre of mass to the support foot.

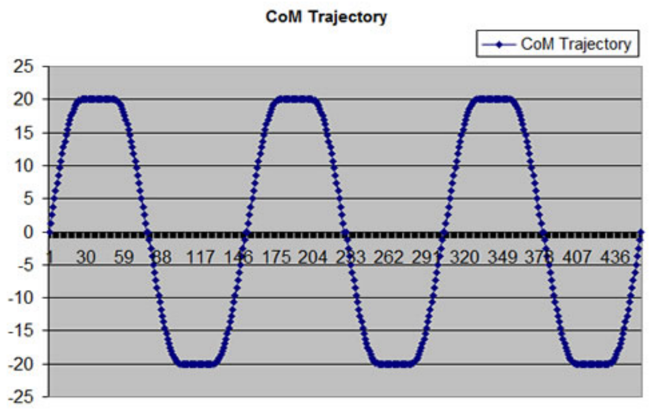


Figure 3.10: *Modified centre of mass trajectory. Amplitude in this example is set to 20cm*

By raising both the step height and robot's height, we were able to walk with larger stride lengths, covering more ground, despite a lower step frequency than previous prototypes. This effectively improved our walk's speed by around 50% bringing it to nearly 12cm/s.

Overview

This final gait before additional parameter tuning was by far the most successful gait in terms of both speed and stability. There were some minor issues such as new methods of interfacing with behaviours to maximise its capabilities but due to short time this was unable to be realised.
Pros:

- Faster than Aldebaran's Gait (approx. 12cm/s).
- Arm motions to reduce drifting effects and improve stability.
- Minor fall prevention measures carried on from previous prototype.

Cons:

- Still a static, open-looped walk.
- Sidestepping was still slower than forward motion.
- Unable to update behaviour interface for maximum utilisation (time constraint).
- There existed a point of discontinuity at the transition phase evident during turning, resulting in slightly jerky motions.
- The fall prevention measures sometimes hindered the walk speed in order to prevent rapid accelerations.

3.4 Results and Post Mortem

In order to determine the effectiveness of the gait, in terms of swiftness, a quantitative analysis was undertaken. Similar to last year's test setup configuration; the robot was made to walk in a straight line across the width of the field (approx. 4.0 metres)[27]. This allowed us to evaluate the speed of our forward (and sideways) walking. When turning on the spot, the robot was placed in the centre of the field and made to rotate at its maximum turning angle for five complete revolutions. Five trials were conducted in order to obtain a variety of samples. The results of the experiments can be seen in table 3.2.

Prototype	Forward Speed	Sideway Speed	Turning Speed	Max. Turn Angle
Initial	3.0 cm/s	0.75 cm/s	7.9 deg/s	16.0 deg
Mid-Semester	8.2 cm/s	4.04 cm/s	8.02 deg/s	16.0 deg
Final	11.06 cm/s	8.04 cm/s	13.71 deg/s	18.0 deg
Hyper	15.92 cm/s	9.10 cm/s	22.15 deg/s	18.0 deg

Table 3.2: *Overview of Prototypes Developed*

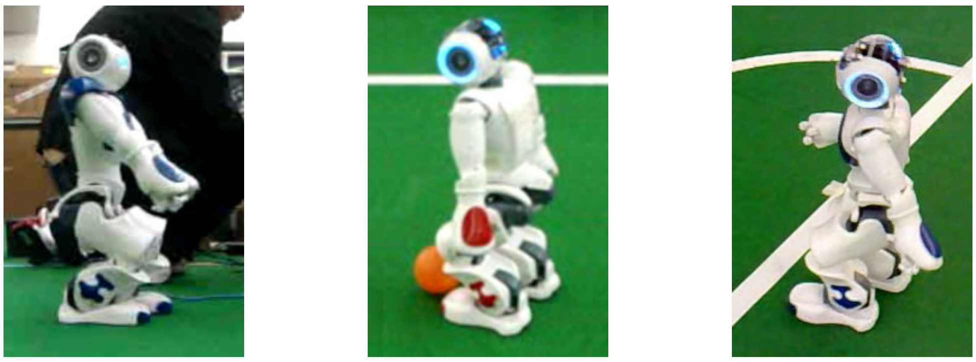

(a) *Initial Prototype* **(b)** *Mid-Semester* **(c)** *Final Prototype*

Figure 3.11: *Various Gaits*

Although our gaits produced up to and including the mid-semester milestone were substantially slower than the Aldebaran Gait, our final (hyper) product was 50% faster in speed. This is a particular achievement given that the resulting gait was a static, open-looped walk. In comparison with some teams which achieved omnidirectional gaits, despite having dynamic closed-loop walks, mildly comparable results were obtained; Northern Bites(10cm/s) and Nao Devils(25cm/s)[24][22].

The developed gait at the conclusion of this project was parametrised and further optimised upon arrival at the world competition. However this set of parameters were only used on the a few of the robots as there was insufficient time to manually adjust all robot.

Chapter 4

Special Actions

4.1 Requirements

A new module to the rUNSWift architecture was the addition of special actions. These special actions allow the robot to execute a set of preplanned joint poses to create a desired action. The main requirements of this module were:

- **Creation:** The actions should be able to be created in a manner which allowed for effective use of time. Time should not be spent on developing actions which could be used to develop other modules.

- **Adaption:** The actions must be able to be changed (modified) easily. Creating actions themselves may be time consuming, so more time should not be wasted in having to recreate actions if a single pose was incorrect or required modifications.

- **Transitions:** Interpolating between walking and the actions are important to prevent the robot from suddenly falling over due to sudden movements. Likewise, transitioning back into walks is vital to reduce unnecessary calculations for the `GaitEngine`.

- **Modular:** From developer's perspective, this module must be designed such that future RoboCup members can make any necessary modifications with minimal effort. In addition, the actions created and executed should be able to work on all robots despite the minor differences between them.

- **Actions:** The minimum set of actions required were the recover (get-up) routines so the robots can continue to play the game without being penalised to the sideline.

4.2 Development

4.2.1 Action Creation and Playback

Actions were created via joint captures to create poses. When combined over time, the interpolation between poses resulted in an action. More information regarding action creation can be found in Appendix A.2.

Execution of an action was done via the `ALMotion` proxy of the NaoQi interface due to its convenience. This allowed an entire action to be passed into the system and executed without the need for fine grain control over the joint angles. The capability of cancelling actions throughout their execution allowed for actions to be switched quickly, however, doing so results in an unstable robotic system. Although there exists literature regarding balancing whilst executing such specialised actions[11], time restrictions limited us into this pursuit.

Some adjustments which were necessary however, were the transitions between walks and actions and vice-versa. In order to achieve smooth transitioning between the two, the robot must finish its step and cleanly leave the gait to prevent momentum forces. This drawback meant the robot would execute an additional step, upon reception of the action, to move the gait into the home position. Once in the home position, actions are free to be executed as the robot is simply standing in this state. Once an action has been completed, the robot's stance must be moved towards the gait's home position before recommencement of the walk as interpolation between awkward positions and the gait resulted in sudden jerks due to the motor joints attempting to move to positions too quickly. This approach, although slightly cumbersome, was sufficient given the limited set of actions developed.

4.3 Actions developed

4.3.1 Recover Routines

Aldebaran Robotics released, at their Nao Design Workshop, two standard recovery routines, one for recovering when the Nao is face down, and another for when the Nao is face up. Initial experiments were carried on the usability of these routines to assess their efficiency on:

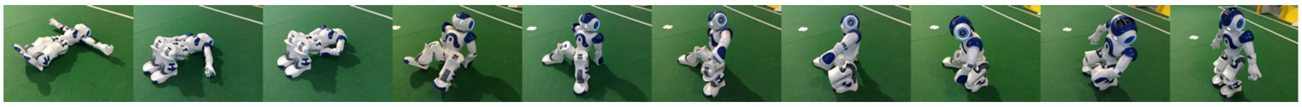
- Time required to execute (move from lying down to standing)
- Portability of routine to other robots (can other robots use the same routine)

Before the existence of the Action Creator, these recovery routines were executed via Choregraphe (see appendix B.3.1). It was discovered that the initial recovery routines provided, when executed on all functioning Nao robots on a smooth surface such as a table or bench, successfully brought the robot from a lying down position back to the standing position. However, when trialled on a rough, surface with sufficient spring such as carpet, most robots failed to recover to the standing position when lying on their backs. Finally, when trialled on our field, a flat surface with slight spring due to the felt, the majority of robots were able to recover from either position on the ground. It is worth noting that on all the robots which failed, there was a common segment in the routine which lead to them falling over - this correction will be explained later in this chapter. These initial test results were encouraging and demonstrated to us that the provided get up routines indeed function properly but with some minor adjustments required.

Upon completion of the Action Creator (and Playback) the recover routines were converted manually into the necessary format required. Manual entry was chosen as it was sufficient given our needs and the small number of routines. Once this was completed, the robots on which the recover routines failed on were trialled again to determine which segment of the poses caused the robots to fall over. The Action Playback allows us to control which segments of a routine to playback. In order to determine what caused the robot to fall over, the combined segments were iteratively played back in order. Once the bad segment was discovered, an extensive analysis was undertaken to determine why it fell over:

- In which direction did the robot fall over?
- Was this due to the robot reducing the size of its support polygon before appropriately shifting its weight to the next support polygon?
- Is the transition into (or out of) the segment discontinuous?

One of the key reasons the robots failed to recover from lying on their back was due to the arm supporting the robot during the transition of the weight to the legs was released prematurely, resulting in a reduction the support polygon[25]. This lead to the robot's centre of mass extending beyond the requirements for stability, thus the robot returned to the ground, failing to complete the recover routine. This problem was overcome by adding momentum (by increasing the speed) to the transitioning segment effectively moving the centre of mass quickly into the support polygon. An alternative solution trialed was the slowing down of the transitioning segment. Although this method was also successful, it increased the total time required to execute the recover routine substantially. Figure 4.1 shows the recovery routines.


(a) *Recover from face up*

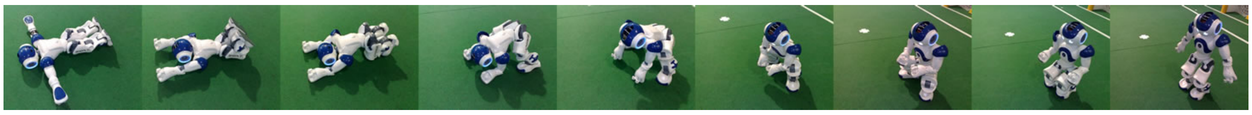

(b) *Recover from face down*

Figure 4.1: *Recover Routines (left to right)*

4.3.2 Kicking

To show ease and speed of developing actions, a quick bunt action was developed. This action was intended to be a supporting action to the main 'Powerhouse kick'. Due to its swiftness and weak strength, it allowed the robot to quickly push the ball ahead whilst still being able to chase after it shortly after due short execution time (four seconds against the nine seconds required for the 'Powerhouse Kick'). Figure 4.2 shows the bunt in action. Under optimal condition, this quick bunt is able to kick the ball approximately 120cm with mean angle error of ten degrees from the immediate straight line (see Figure 4.3).

Despite behaviours deciding against the adoption of the bunt in the actual competitions, the creation of this quick bunt lead to the discovery of a discrepancy within the Nao Robot's design specification - the leg limb lengths are **asymmetric**. This was discovered due to the *mirror swapping* of joint angles from left to right. Actions which work on one side failed to do so on the other side. For example, kicking with the right foot was successful, but after the joint swap, kicks with the left foot resulted in the kicking foot's toe colliding with the floor, producing some instabilities. This was an important issue as it also allowed us to accommodate for minor instability issues experienced within the Gait Engine.

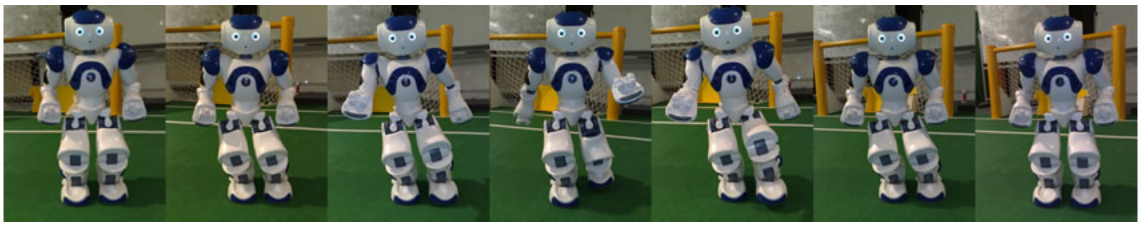


Figure 4.2: *Quick Bunt (left to right)*

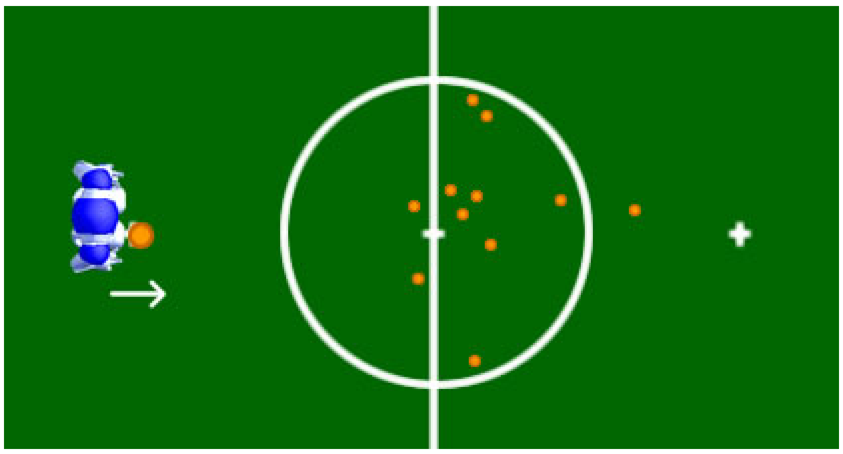


Figure 4.3: *Quick Bunt Range*

4.3.3 Goalie Guarding

The goalie required some special actions to be able to maximise its cover of the goal in order to reduce the chances of an opponent from kicking a goal. As such, a 'crouch-and-recover' routine was developed. This action was broken down into two segments `GoalieCrouch` and `GoalieRecover`. The crouch was actually a few segments of a recover routine played in reverse as this allowed us to bring the robot down to the crouch position in a quick and stable fashion. The leg spread was widened and hip lowered slightly such that we cover as much area as possible as well as preventing any balls slipping between the robot's pelvis and ground. The robot's hip joint motors were taken into consideration when lowering the robot as the bulk of strain is placed on the hip joints, which can greatly impede a robot's ability to walk if damaged. `GoalieRecover` was simply the crouch action played in reverse. Figure 4.4 shows the crouching action. Finally, minor adjustments to the Locomotion framework were required in order to prevent the robot from walking after a `GoalieCrouch` and only to react to the recover (including `GoalieRecover`) routines. This requirement was largely due to the requirements of transitioning between actions and walks previously stated and highlighted the inflexibility of the system.

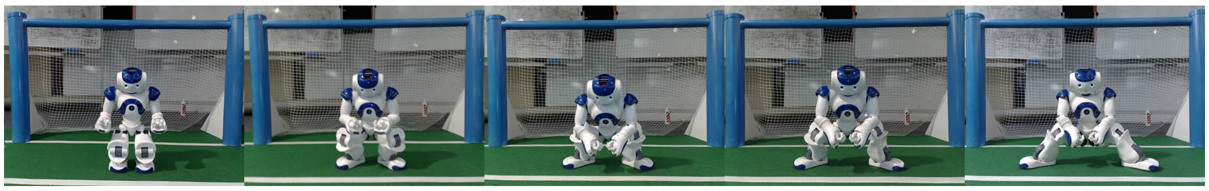


Figure 4.4: *Goalie Crouch (left to right) & Recover (right to left)*

4.4 Results and Post Mortem

Special actions were an integral part of the rUNSWift architecture and allowed the higher control levels such as Behaviours to manipulate the robot's body like a real soccer player.

One common pitfall encountered when developing the recovery routines was that as individual poses, executed one by one, the robot would often fail to recover to the standing position. However, when executed in its entirety, it was able to successfully recover. Although the reasoning behind this is obviously due to momentum it is worth mentioning for future teams.

In terms of the initial requirements stated, the project successfully met these requirements. However, some which issues arose included the restrictions on executing the actions. These included the precondition that the gait must be reset and the post condition that all actions must be terminated with the home position pose. The biggest issue was that the current framework does not allow for the easy addition of other 'locking' actions such as the goalie crouch and recover pair, nor the transitioning between different actions.

On the other hand, there were many accomplishments from this project. The discovery of the robot's asymmetric leg limb length provided some reasoning for minor faults of the gait engine and the accommodation of these hardware issues contributed to a more stable walk. The creation and playback features allowed rapid prototyping of actions which greatly reduced development time. The interfacing of the actions allowed them to be played on the robot independently of the rUNSWift code base and potentially be used for demonstrations.

Chapter 5

Results

5.1 The Competition

Despite the achievements of an omnidirectional gait, rUNSWift suffered a substantial blow in the world competition in 2009. The gait however, proved to be a great asset against a number of our competitors which allowed us to manoeuvre around opposition robots in order to approach the ball and dominate the game on ball control. However, the execution of our strategies were greatly flawed, which lead to numerous opportunities for our oppositions.

5.2 Lessons Learnt

Dynamic walking is much more difficult than initially envisioned. Starting with the initial approach of a static walk and attempting to add a stability controller on top seemed to interfere more with the walking as opposed to assisting it. Dynamic walking should be developed from the ground up incorporating and utilising appropriate sensory data from the beginning. This approach however, might result in a system which may or may not work which is extremely high risk given the short project life cycle.

Robotics, particularly in the real world, has an almost infinite number of variables, and, unlike a simulated world is not perfect. As such, any changes which are committed, must be tested and trialled extensively in the real world - as this was our field of application. These included the adjustment of walk parameters, bug fixes, optimisations etc... Even if a change makes one robot act better, it must be trialled on all other robots and under strict game condition to ensure the changes are valid and not just random luck.

Chapter 6

Conclusion

This project resulted in rUNSWift's first bipedal omnidirectional walk capable of approaching the ball faster than the Aldebaran Gait and most of our competitors. Despite failing to achieve the initial aims of the stability requirement, the completion of a statically open-loop gait was sufficiently stable under normal game condition. Finally, we presented a method of executing numerous actions such as kicking and recovery routines via pose interpolation. Despite these achievements, rUNSWift is still far from a dynamic walk. However, it is hoped that the ideas generated from this thesis provide some insight into the field for future years.

6.1 Future Work

It would be worthwhile spending a substantial amount of time interpreting sensory data in order to understand the dynamics of the Nao, particularly if a dynamic walk is to be achieved in the future. A self calibration method of all joints and sensors may prove to be useful as manual calibration is tedious and prone to human error. This would allow the robots to be used on any surface at anytime resulting in a surface independent system.

The current approach with pose interpolation could be improved (by removing the pose dependancies) and in doing so may require the transition from `ALMotion` to the `DCM`. The `DCM` allows direct access to the motors and its usage throughout the league is beginning to become apparent in showing its effectiveness.

The ankles in this project were statically defined and remained always parallel to the ground via the kinematic model. It may be necessary to allow manipulation of these ankles for a dynamic walk.

Bibliography

[1] *The Developmenet of Honda Humanoid Robot*, 1998.

[2] *Biped Walking Pattern Generation by using Preview Control of Zero Moment Point*, 2003.

[3] *Humanoid robot HanSaRam: Schemes for ZMP compensation*, 2003.

[4] *Programmable Central Pattern Generators: an application to biped locomotion control*, 2006.

[5] Aldebaran Robotics. *NaoWare Documentation*, 2009. Technical Specifications Document.

[6] J. Baltes and P. Lam. Design of Walking Gaits for Tao-Pie-Pie, a Small Humanoid Robot. Technical report, University of Manitoba, 2003.

[7] S. Chen, M. Siu, T. Vogelgesang, T. F. Yik, B. Hengst, S. B. Pham, and C. Saummut. The UNSW RoboCup 2001 Song Legged League Team. Unpublished Report, 2001.

[8] J. Christensen, J. L. Nielsen, M. S. Svendsen, and P. F. Orts. Development, Modeling And Control of A Humanoid Robot. Master's thesis, Aalborg University, 2007.

[9] D. Collien and G. Huynh. From AIBO to Nao: The Transition from 4-legged to 2-legged Robot Soccer. Unpublished Thesis, 2008.

[10] S. Collins and A. Ruina. A Bipedal Walking Robot with Efficient and Human-like Gait. Technical report, University of Michigan, Cornell University, 2003.

[11] S. Czarnetzki, S. Kerner, and O. Urbann. Applying Dynamic Walking Control for Biped Robots. Technical report, 2009.

[12] G. Endo, J. Morimoto, T. Matsubara, J. Nakanishi, and G. Cheng. Learning CPG-based Biped Locomotion with a Policy Gradient Method: Application to a Humanoid Robot. *Int. J. Rob. Res.*, 27(2):213–228, 2008.

[13] T. R. et al. B-Human Team Report and Code Release 2008. Technical report, University of Bremen, 2008.

[14] D. Gouaillier, V. Hugel, P. Blazevic, C. Kilner, J. Monceaux, P. Lafourcade, B. Marnier, J. Serre, and B. Maisonnier. The NAO humanoid: a combination of performance and affordability. *CoRR*, abs/0807.3223, 2008. informal publication.

[15] B. Hengst, D. Ibbotson, S. B. Pham, and C. Sammut. The UNSW United 2000 Sony Legged Robot Software System. Unpublished Report, 2000.

[16] K. Hirai. The Honda Humanoid robot: development and future perspective. *Industrial Robot: An International Journal*, 26,, 1999.

[17] J. Liu, M. Tay, and X. Zhao. Legged Robots - and Overview. *Transactions of the Institute of Measurement and control*, 2007.

[18] K. Loffler, M. Gienger, and F. Pfeiffer. Study of ZMP Measurement for Biped Robot Using FSR Sensor. *The International Journal of Robotics Research*, 22, 2003.

[19] P. Manoonpong, T. Geng, T. Kulvicius, B. Porr, and F. Worgotter. Adaptive, fast walking in a biped robot under neuronal control and learning. *PLoS Computational Biology*, 3, July 2007.

[20] E. Nicholls. Bipedal Dynamic Walking in Robotics. Master's thesis, 1998.

[21] J. Nicolas. Artificial evolution of controllers based on non-linear oscillators for bipedal locomotion. Master's thesis, Biologically Inspired Robotics Group, 2005.

[22] Northern Bites. Northern Bites Blog. `http://robocup.bowdoin.edu/blog/2009/07/02/northern-bites-beat-nao-devils-2-0/`.

[23] B. G. Shim, E. H. Lee, H. K. Min, and S. H. Hong. Study of ZMP Measurement for Biped Robot Using FSR Sensor. Technical report, Korea Polytechnic University & University of Incheon.

[24] J. Strom, G. Slavov, and E. Chown. Omnidirectional Walking using ZMP and Preview Control for the NAO Humanoid Robot. Technical report, Bowdoin College, 2009.

[25] J. Stuckler, J. Schwenk, and S. Behnke. Getting Back on Two Feet: Reliable Standing-up Routines for a humanoid Robot. Technical report, University of Freiburg, 2003.

[26] M. Tad. Passive Dynamic Walking. Technical report, School of Engineering Science, Simon Fraser University, 1988.

[27] A. Tay. Exploration of Nao and its Locomotive Abilities. Unpublished Report, 2008.

[28] A. Tay. Teaching the Nao to Walk: One step for Nao. Unpublished Report, 2008.

[29] M. Vukobratovic, B. Borovac, and V. Potkonjak. Towards a unified understand of basic notions and terms in humanoid robotics. *Robotica*, 25, 2007.

[30] F. Yamasaki, T. Matsui1, T. Miyashita1, and H. Kitano. PINO The Humanoid: A Basic Architecture. Technical report, ERATO, JST, Osaka University, 2001.

[31] T. F. Yik. *Locomotion of Bipedal Humanoid Robots: Planning and Learning to Walk*. PhD thesis, University of New South Wales, 2007.

Finally, a special thanks to everyone on the 2009 rUNSWift Team

Appendix A

Locomotion Documentation

A.1 Walking

From behaviours, the only method to execute as walk via the blackboard through the issuing of an `ActionCommand::Body`:

```
#include ``act/locomotion/ActionCommand.hpp''
#include ``blackboard/Blackboard.hpp''
...
ActionCommand::Body body(forward, left, turn);
valueToBlackboard(bodyCommand, body);
```

Listing A.1: *Issuing a Walk Command*

The walk parameters define how the robot moves and is found in `/etc/runswift/cfg/walk.cfg`. By default the standard walk will be executed in case this cannot be located. There are 11 main parameters and 6 adjustment parameters. Description of parameters and tuning follows. Below is the best set of parameters for the walks as used in Graz, Austria:

```
//Fastest walk achieved (~15cm/s)
20 60 175 20 21 -5 0 0 0 6 10 0.1 0.1 0.05 0.05 8.8 11.0 4.3

//Standard walk
15 60 175 25 27 -5 0 0 0 10 10 0.1 0.1 0.05 0.05 8.8 11.0 4.3
```

Listing A.2: *Walk Parameters*

A.1.1 Walk Parameter Description

Main Parameters:

- **Step height:** How high off the ground to lift the foot (mm)
- **Step length:** How far forward to step (mm)
- **Body height:** How high above the ground is the torso (mm)
- **Torso Left:** How much to shift the torso to the left (mm)
- **Torso Right:** How much to shift the torso to the right (mm)
- **Torso lean:** How far left/right to tilt the upper torso ('roll') (degrees)
- **Torso pitch:** How far forwards/backward to tilt the upper torso (degrees)
- **Length of air control points:** Affects swing leg's forward/backward 'kick-out'
- **Angle between air control points:** Affects angle of swing leg's lift
- **Shift duration:** How many updates required to shift the body onto the support leg (Positive integer)
- **Swing duration:** How many updates required to move the swing leg (Positive interger)

Adjustment Parameters:

- **Forward and Left Drift:** How much extra turn to induce when walking forwards and left (radians)
- **Forward and Right Drift:** How much extra turn to induce when walking forwards and right (radians)
- **Sidesteps Left Drift:** How much extra turn to induce when sidestepping left (radians)
- **Sidesteps Right Drift:** How much extra turn to induce when sidestepping right (radians)
- **Forward Odometer:** Factor to scale forward odometer (Real number)
- **Left Odometer:** Factor to scale leftward odometer (Real number)
- **Turn Odometer:** Factor to scale turning odometer (Real number)

A.1.2 Parameter Tuning

Below are some tips on parameter tuning for the walks

- If swaying to much to the left, decrease torsoLeft (or increase torsoRight). Similarly, if swaying to much to the right, decrease torsoRight (or increase torsoLeft)
- Higher body height means more instability and higher center of mass. As such, the robot does not need to sway left/right as much and the torsoLeft/Right parameters may be reduced.
- The maximum step height is related to and affected by how much the robot sways. More sway can often lead to a higher step height. Adding a torso lean can also improve the step height.
- The length of the step is largely dependent on how 'balanced' the robot is. Higher step heights usually allow for this to be larger but be prepared as stability will decrease.
- If the robot looks like it is falling in the forwards/backwards direction, adjusting the torsoPitch value will often reduce this.
- The control points affect the step trajectory. If the robot's feet are dragging or slipping on the floor, use this to bring the mobile foot faster. A slowly returning foot to the ground reduces the force of impact.
- Quicker swing durations can lead to the left/right swaying also being reduced.
- If drifting right on left movement, increase value turn adjustment value. Similarly for drifting left on right movement.
- If odometry is larger, increase the factor
- If odometry is smaller, decrease the factor

A.2 Actions

From behaviours, controlling of the head is similar to walking and it achieved via the blackboard through the issuing of an `ActionCommand::Head`:

```
#include ``act/locomotion/ActionCommand.hpp''
#include ``blackboard/Blackboard.hpp''
...
ActionCommand::Head head(pitch, yaw, [pitchRelative], [yawRelative]);
valueToBlackboard(headCommand, head);
```

Listing A.3: *Issuing a Head Command*

Special Actions however, are a type of body command and should be used as follows:

```
#include ``act/locomotion/ActionCommand.hpp''
#include ``blackboard/Blackboard.hpp''
...
ActionCommand::Body body(ActionCommand::Body::KICK_LEFT);
valueToBlackboard(bodyCommand, body);
```

Listing A.4: *Issuing a Head Command*

Notice that when creating a body command, the type of action to be invoked is passed in directly. Consult (and update if necessary) `act/locomotion/ActionCommand.hpp` for other available actions. Appendix B.2 has some information on how to generate actions.

Appendix B

Software and Tools

B.1 Remote Control

This tool allows the robot to be controlled like a toy via manual entry or keyboard arrow interface. In order for this to work, the code base must be compiled with the follow `CMAKE` flags:

```
ENABLE_ACTUATOR            ON
ENABLE_BEHAVIOUR           OFF
ENABLE_COMMUNICATION       ON
ENABLE_PROFILING           OFF
ENABLE_REMOTE_CONTROL      ON
```

Listing B.1: *CMake flags for compiling a Remote Controlled Nao*

The resulting dynamic library generated should be placed on the Nao and execute normally. Next, execute the remote control program: `./remote_control ip_of_nao`. Remote control is not limited to locomotion. If a localisation module is unavailable or if behaviours need testing, the values in the blackboard can be overwritten. Same for vision for switching cameras.

B.2 Action Creation and Representation

The creation of actions is made fairly simple through this tool which allows the user to *'capture'* (or freeze frame) all the joint angles of the robot. It also has the capability to playback actions created providing they conform to the format specified henceforth. Of course, nothing prevents you from entering joint angles manually. However, the duration of interpolation between poses needs to be entered manually. The standard layout for the `.pos` files is:

```
[number of poses] [head flag] [interpolation flag] [blocking flag]
[joint angles ..] [duration]
...
[timestamp]
```

Listing B.2: *Typical layout of an action*

- **Number of poses:** This denotes how many entries within the file
- **Head Flag:** If the action uses the head, this is set to 1. Otherwise 0. By not using the head (i.e. set to 0) the head is able to be controlled whilst the action is being executed.
- **Interpolation Flag:** Consult `act/locomotion/MotionModule.hpp`.
- **Blocking Flag:** If the action is blocking, this is set to 1. Otherwise 0. If an action is blocking, the locomotion module effective stops at the point of execution and will ignore all future commands until the action has completed. The head is not affected by this if the head is not used in the action.
- **Joint Angles:** These are the angles of the joints in the pose. The order of joints must be in the order specified in sharejoints.hpp. If generated via the action creator, this will be done automatically. Note: Joint angles are in degrees (not the usual radians).
- **Duration:** How long (in milliseconds) to interpolate from the current pose into the next pose.
- **Timestamp:** This is automatically generated if the action was generated via the action creator.

B.3 Aldebaran

B.3.1 Choregraphe

Choregraphe is a software solution developed by Aldebaran Robotics with the goal of providing a method of editing Nao's behaviours and movements. Although the user is referred to the Choregraphe documentation for execution procedures, some key features of Choregraphe will be highlighted.

By far the most useful feature is that Choregraphe has a visualiser which can show the pose of the robot. It is also able to be connected to any system which has the NaoQi software executing (including simulation programs and the physical Nao). This makes it a highly valuable resource.

We mainly use Choregraphe to test the joint angles of the robot and to ensure it was in a working condition, particularly after receiving one from repairs. Choregraphe was used as it allowed sufficient control over each joint through the `ALMotion` proxy (which was used by our system). There is also another method of testing the robot's joints which is the 'Tai Chi Test'. When the robot is started up, place both arms upright into the air. Then with a single push of the chest button, the robot will execute this routine. If at any stage the robot falls, then there are possibly some issues related to its legs.

Choregraphe, as stated previously, allows motions to be created. This can be achieved via the time line interface. This works in a similar manner to our action creator where you can set joints of the robot, capture the angles and adjust the time durations. Similarly, Choregraphe can be use to set the joint angles of the robot and then our action creator used to freeze frame and generate our action files.

B.3.2 Telepathe

Telepathe also developed by Aldebaran Robotics enables the user to extract and view all sensory information regarding the Nao. Again, refer to the Telepathe documentation for proper usage. Some camera settings which can be trial including the resolution, white balances, horizontal flips etc... It also provides live data views of sensors including the IMU, FSRs, joint angles, battery status, motor loads etc... This is an extremely effective and efficient way of viewing live data without having the need to modify code in order to extract data. It also provided a quick means of checking the camera state for the 'green image bug' experienced the previous year[9].