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

# Forward and Inverse Kinematics for the NAO Humanoid Robot

Nikolaos Kofinas

Thesis Committee
Assistant Professor Michail G. Lagoudakis (ECE)
Professor Minos Garofalakis (ECE)
Assistant Professor Aggelos Bletsas (ECE)

Chania, July 2012

Πολυτεχνειο Κρητης

Τμημα Ηλεκτρονικων Μηχανικων και Μηχανικων Υπολογιστων

# Ευθεία και Αντίστροφη Κινηματική για το Ανθρωποειδές Ρομπότ NAO

Νικόλαος Κοφινάς

Εξεταστική Επιτροπή

Επικ. Καθ. Μιχαήλ Γ. Λαγουδάκης (ΗΜΜΥ)

Καθ. Μίνως Γαροφαλάκης (ΗΜΜΥ)

Επικ. Καθ. Άγγελος Μπλέτσας (ΗΜΜΥ)

Χανιά, Ιούλιος 2012

# Abstract

Articulated robots with multiple degrees of freedom, such as humanoid robots, have become popular research platforms in robotics and artificial intelligence. Such robots can perform complex motions, including the balancing, walking, and kicking skills required in the RoboCup robot soccer competition. The design of complex dynamic motions is achievable only through the use of robot kinematics, which is an application of geometry to the study of arbitrary robotic chains. This thesis studies the problems of forward and inverse kinematics for the Aldebaran NAO humanoid robot and presents for the first time a complete analytical solution to both problems with no approximations, including an implementation of a software library for real-time execution. The forward kinematics allow NAO developers to map any configuration of the robot from its own joint space to the three-dimensional physical space, whereas the inverse kinematics provide closed-form solutions to finding joint configurations that drive the end effectors of the robot to desired points in the three-dimensional space. The proposed solution was made feasible through a decomposition into five independent problems (head, two arms, two legs), the use of the Denavit-Hartenberg method, and the analytical solution of a non-linear system of equations. The main advantage of the proposed inverse kinematics compared to existing numerical approaches is its accuracy, its efficiency, and the elimination of singularities. The implemented NAO kinematics library, which additionally offers center-of-mass calculation, is demonstrated in two motion design tasks: pointing to the ball and basic balancing. The library has been integrated into the software architecture of the RoboCup team "Kouretes" and is currently being used in various motion design problems, such as dynamic balancing, trajectory following, dynamic kicking, and omnidirectional walking.

# Περίληψη

Τα αρθρωτά ρομπότ με πολλαπλούς βαθμούς ελευθερίας, όπως τα ανθρωποειδή ρομπότ, έχουν γίνει δημοφιλείς πλατφόρμες έρευνας στη ρομποτική και την τεχνητή νοημοσύνη. Τα εν λόγω ρομπότ μπορούν να εκτελέσουν σύνθετες κινήσεις, συμπεριλαμβανομένων και των δεξιοτήτων ισορροπίας, βαδίσματος και λακτίσματος που απαιτούνται στον διαγωνισμό ρομποτικού ποδοσφαίρου RoboCup. Ο σχεδιασμός πολύπλοκων δυναμικών κινήσεων μπορεί να επιτευχθεί μόνο μέσω της χρήσης ρομποτικής κινηματικής, που είναι η εφαρμογή της γεωμετρίας στη μελέτη αυθαίρετων ρομποτικών αλυσίδων. Η παρούσα διπλωματική εργασία μελετά τα προβλήματα της ευθείας και αντίστροφης κινηματικής για το ανθρωποειδές ρομπότ Aldebaran NAO και παρουσιάζει για πρώτη φορά μια πλήρη αναλυτική λύση και για τα δύο προβλήματα χωρίς προσεγγίσεις, συμπεριλαμβανομένης μιας υλοποίησης βιβλιοθήκης λογισμικού για εκτέλεση σε πραγματικό χρόνο. Η ευθεία κινηματική επιτρέπει στους προγραμματιστές του NAO να απεικονίσουν οποιαδήποτε διάταξη του ρομπότ από τον χώρο των αρθρώσεών του στον τρισδιάστατο φυσικό χώρο, ενώ η αντίστροφη κινηματική παρέχει λύσεις κλειστής μορφής για την εξεύρεση διατάξεων των αρθρώσεων που οδηγούν τα άκρα του ρομπότ σε επιθυμητά σημεία στον τρισδιάστατο χώρο. Η προτεινόμενη λύση κατέστη εφικτή χάρη στην αποσύνθεση σε πέντε ανεξάρτητα προβλήματα (κεφάλι, δύο χέρια, δύο πόδια), στη χρήση της μεθόδου Denavit-Hartenberg και στην αναλυτική επίλυση ενός μη-γραμμικού συστήματος εξισώσεων. Το κύριο πλεονέκτημα της προτεινόμενης αντίστροφης κινηματικής σε σύγκριση με υφιστάμενες αριθμητικές προσεγγίσεις είναι η ακρίβειά της, η αποδοτικότητά της και η εξάλειψη των ιδιόμορφων περιπτώσεων. Η υλοποιημένη βιβλιοθήκη κινηματικής για το NAO, η οποία περιλαμβάνει και υπολογισμό του κέντρου μάζας, επιδεικνύεται σε δύο προβλήματα σχεδιασμού κίνησης: κατάδειξη μπάλλας και βασική ισορροπία. Η βιβλιοθήκη έχει ενσωματωθεί στην αρχιτεκτονική λογισμικού της ομάδας RoboCup «Κουρήτες» και χρησιμοποιείται σε διάφορα προβλήματα σχεδιασμού κινήσεων, όπως δυναμική ισορροπία, παρακολούθηση τροχιάς, δυναμικά λακτίσματα και πολυκατευθυντικό βάδισμα.

# Acknowledgements

First of all, I would like to thank Manolis Orf (a.k.a. "re palikari") for his help and his great ideas as well as for the great fights we had.

Next, I would like to thank my advisor Michail G. Lagoudakis for his inspiration and the trust that he showed in me.

Fanoula is the next person that I would like to thank ◡̈. She helped me so much during this difficult period and I am so lucky that she still talks to me.

Team Kouretes (N. Pav, A. Top, M. Kounoupidi, D. Janetatou, Orf, Iris), I can't understand why you still talk to me after all the things we've been through together in "ypoga". After all, I like this team and our lab more than I had imagined; thank you for your help and all the fun in the team.

To sum up, I would like to thank my friends with whom I had the greatest five years of my life. N. Pavlakis (he paid five Euros for the second reference), E. Alimpertis, K. Perros, and E. Soulas, thank you for everything!

Of course, I don't have words to describe the help I received from my parents, so I ...

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Articulated robots with multiple degrees of freedom, such as humanoid robots, have become popular research platforms in robotics and artificial intelligence. Such robots can perform complex motions, including balancing, walking, standing up, etc. A challenging domain, where humanoid robots are called to demonstrate complex motion skills is the RoboCup (robot soccer) competition [**?**], whereby teams of autonomous robots compete against each other in various leagues. In this thesis we focus on the Aldebaran NAO humanoid robot, which is used exclusively by all teams competing in the Standard Platform League (SPL) of the RoboCup competition. NAO a mid-size humanoid robot with 21 degrees of freedom (independently-moving joints) divided in five kinematic chains (a head, two arms, two legs). NAO is capable of performing various complex motions and, in fact, many SPL teams have designed and implemented their own omni-directional walk algorithms [**?**, **?**], balancing methods [**?**], and kick engines [**?**] to be more competitive.

It is widely known that the design of complex dynamic motions is achievable only through the use of robot kinematics, which is an application of geometry to the study of arbitrary robotic chains [**?**]. Robot kinematics include forward and inverse kinematics. The forward kinematics provide the means to map any configuration of the robot from its own multi-dimensional joint space to the three-dimensional physical space in which the robot operates, whereas the inverse kinematics provide the means to finding joint configurations that drive the end effectors of the robot to desired points in the three-dimensional space. It is easy to see why kinematics are required in any kind of complex motion design. Stable walk gaits rely on the ability of the robot to follow planned trajectories with its feet; this is not possible without some mechanism that allows the

robot to set its joints to angles that drive the feet to points along such trajectories, an instance of inverse kinematics. Likewise, balancing methods rely on the ability to calculate the center of mass of the robot, which is constantly changing as the robot moves; finding the center of mass is made possible, only if the exact position and orientation of each part of the robot in the three-dimensional space is known, an instance of forward kinematics. It is also quite understandable that any kinematics computations must be performed in real-time to be useful in dynamic motions.

## 1.1 Thesis Contribution

This thesis studies the problems of forward and inverse kinematics for the Aldebaran NAO humanoid robot and contributes for the first time a complete analytical solution to both problems with no approximations. In addition, it contributes an implementation of the proposed NAO kinematics as a software library for real-time execution on the robot. This work enables NAO software developers to make transformations between configurations in the joint space and points in the three-dimensional physical space on-board in just microseconds.

The proposed solution was made possible through a decomposition into five independent problems (head, two arms, two legs), the use of the Denavit-Hartenberg method [**?**, **?**], and the analytical solution of a non-linear system of equations. Existing methods for NAO inverse kinematics either offer analytical solutions [**?**], but only under certain simplification assumptions, or offer approximate numerical solutions [**?**], which are nevertheless subject to singularities. The main advantage of the proposed inverse kinematics compared to existing approaches is its accuracy, its efficiency, and the elimination of assumptions and singularities.

This thesis additionally contributes two demonstrations of NAO kinematics: (a) a pointing-to-the-ball task, whereby the robot tracks a ball in the field and uses forward and inverse kinematics to point to the exact location of the ball with its stretched arm(s), and (b) a basic balancing method, whereby the robot calculates its current center of mass through the help of forward kinematics and drives one of its legs to the projection of the center of mass on the floor using inverse kinematics to maintain balance. The implemented NAO kinematics library has been integrated into the software architecture of the RoboCup team "Kouretes" and is currently being used in various motion design problems,

such as dynamic balancing, trajectory following, dynamic kicking, and omnidirectional walking.

## 1.2 Thesis Outline

Chapter 2 describes the RoboCup competition, the Standard Platform League (SPL), our SPL team Kouretes, and the Aldebaran NAO humanoid robot. Furthermore, it provides basic background information about generic robot kinematics, affine transformation matrices, and the Denavit-Hartenberg (DH) parameters. In Chapter 3 we provide a complete description of the NAO hardware and we define the problem of kinematics for the NAO robot. Furthermore, we discuss the related work about forward and mainly inverse kinematics for the NAO robot. In Chapter 4 we describe in detail our solutions to the problems of forward and inverse kinematics for the NAO robot. Additionally, we explain the implementation of kinematics and integration with our team's code. In Chapter 5 we present the real-time performance of our kinematics mechanism and a couple of scenarios to demonstrate its effectiveness. Finally, in Chapter 6 we discuss the results of this thesis and we compare it with other related approaches, pointing out at the same time possible future research directions.

# Chapter 2

# Background

## 2.1 RoboCup

The RoboCup competition was initially inspired by Hiroaki Kitano [**?**] in 1993 and his idea eventually led to the establishment of the RoboCup Federation. The RoboCup competition has a bold vision: "By the year 2050, to develop a team of fully autonomous humanoid robots that can win against the human world soccer champions". All the teams participating in RoboCup have to find real-time solutions to some of the most difficult problems in robotics (perception, cognition, action, coordination). All the divisions in RoboCup (soccer, RoboCup@Home, RoboRescue, etc.) are designed so as to test the proposed solutions by the various teams to the problems mentioned above. So far, the researchers participating in RoboCup have made a lot of progress in solving real-world problems that show up in the various RoboCup leagues within each division.

### 2.1.1 Standard Platform League

The Standard Platform League (SPL) is one of the many leagues in the soccer division of RoboCup. In this league all the teams use the same robot, the Aldebaran NAO humanoid robot, and they focus only on algorithm design and software development for this robot. For this reason, the teams are prohibited to make any changes to the hardware of the robot. The robots are completely autonomous and no human intervention from team members is allowed during the games. The only interaction of the robots with the "outer


Figure 2.1: Standard Platform League at RoboCup 2012 (from www.botsport.tv)

world" is the reception of data from the Game Controller, a computer that broadcasts information about the state of the game (score, time, penalties, etc.).

Currently, the SPL games are conducted on a field with dimensions $4m \times 6m$ [**?**]. The field consists of a green carpet marked with white lines and two yellow goals (Figure 2.1). The appearance of the field is similar to a real soccer field, but it is scaled to the size of the robots. The ball is an orange street hockey ball. Each team consists of four robots and each robot carries a colored waist band (blue or pink) that distinguishes the teams. The total game time is 20 minutes and is broken in two halves; each half lasts 10 minutes.

### 2.1.2 Robocup SPL Team Kouretes

Team Kouretes (www.kouretes.gr) is the RoboCup team of the Technical University of Crete. The team was founded in 2006 and participates in the main RoboCup competition ever since in various leagues (Four-Legged, Standard Platform, MSRS, Webots), as well as in various local RoboCup events (German Open, Mediterranean Open, Iran Open, RC4EW, RomeCup) and RoboCup exhibitions (Athens Digital Week, Micropolis, Schoolfest). In May 2010, the team hosted the 1st official SPL tournament in Greece (with

Figure 2.2: Team Kouretes at RoboCup 2012 in Mexico City

three invited teams) within the Hellenic Conference on Artificial Intelligence (SETN). Distinctions of the team include: **2nd** place in MSRS at RoboCup 2007; **3rd** place in SPL-Nao, **1st** place in SPL-MSRS, among the **top 8** teams in SPL-Webots at RoboCup 2008; **1st** place in RomeCup 2009; **6th** place in SPL-Webots at RoboCup 2009; **2nd** place in SPL at RC4EW 2010; and **2nd** place in SPL Open Challenge Competition at RoboCup 2011 (joint team Noxious-Kouretes).

The team has been developing its own (publicly-available) software for the Nao robots since 2008. The team code repository includes a custom software architecture, a custom communication framework, graphical tools for monitoring and behavior specification, and modules for object recognition, state estimation, localization, obstacle avoidance, behavior execution, team coordination. The members of the team are senior undergraduate ECE students working on their diploma thesis on a RoboCup-related topic; 15 diploma theses have been completed so far. Recently, the team participated in the RoboCup German Open 2012 competition in Magdeburg, in RoboCup Iran Open 2012 in Tehran, and in RoboCup 2012 in Mexico City (Figure 2.2). In the most recent RoboCup 2012

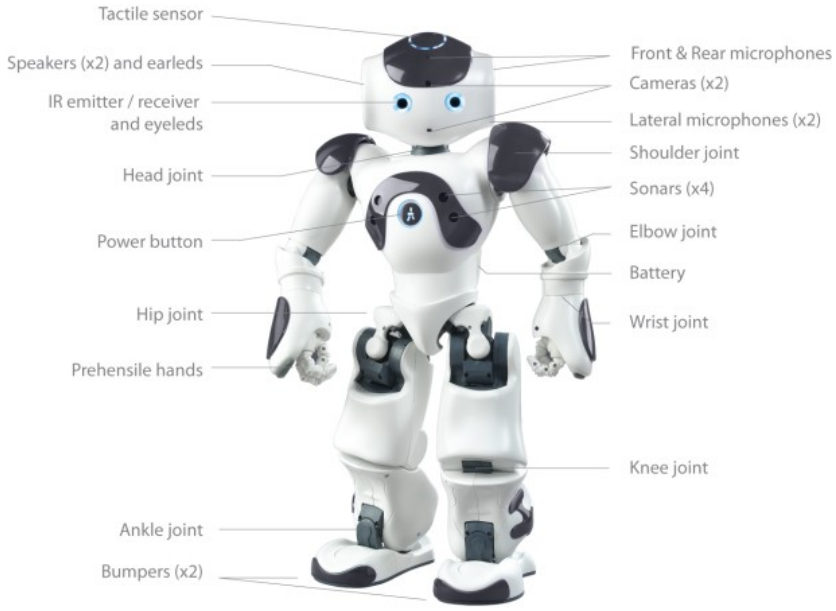



Figure 2.3: Aldebaran NAO v3.3 (Academic edition) components

competition, the team succeeded to proceed to the second round-robin round and rank among the top-16 SPL teams in the world.

## 2.2 Aldebaran NAO Humanoid Robot

NAO is an integrated, programmable, medium-sized humanoid robot developed by Aldebaran Robotics in Paris, France [**?**]. Project NAO started in 2004. In August 2007 NAO officially replaced Sony's AIBO quadruped robot in the RoboCup SPL. In the past few years NAO has evolved over several designs and several versions.

NAO (version V3.3) is a 58cm, 5kg humanoid robot (Figure 2.3). The NAO robot carries a fully capable computer on-board with an x86 AMD Geode processor at 500 MHz, 256 MB SDRAM, and 2 GB flash memory running an Embedded Linux distribution. It is powered by a 6-cell Lithium-Ion battery which provides about 30 minutes of continuous operation and communicates with remote computers via an IEEE 802.11g wireless or a wired ethernet link.

NAO RoboCup edition has 21 degrees of freedom; 2 in the head, 4 in each arm, 5 in each leg and 1 in the pelvis (there are two pelvis joints which are coupled together

on one servo and cannot move independently). NAO, also, features a variety of sensors. Two cameras are mounted on the head in vertical alignment providing non-overlapping views of the lower and distant frontal areas, but only one is active each time and the view can be switched from one to the other almost instantaneously. Each camera is a $640 \times 480$ VGA devise operating at 30fps. Four sonars (two emitters and two receivers) on the chest allow NAO to sense obstacles in front of it. In addition, the NAO has a rich inertial unit, with one 2-axis gyroscope and one 3-axis accelerometer, in the torso that provides real-time information about its instantaneous body movements. Two bumpers located at the tip of each foot are simple ON/OFF switches and can provide information on collisions of the feet with obstacles. Finally, an array of force sensitive resistors on each foot delivers feedback of the forces applied to the feet, while encoders on all servos record the actual values of all joints at each time.

## 2.3 Robot Kinematics

A *robot kinematic chain* is an articulated manipulator that interacts with the environment and is typically described as an assembly of robotic links connected by (rotary) joints. The joints rotate and control the relative angular positioning of the links of the manipulator. Not all combinations of joints' positions in the chain are valid, because some combinations lead to collisions between the links of the chain or with some fixed item of the environment, such as the floor or a wall. All the valid combinations of joint values form the *joint space*. The term *degrees of freedom* (DOF) refers to the number of joints in a kinematic chain; clearly, more DOF imply more flexibility in the motion of the chain.

*Robot kinematics* is the application of geometry to the study of kinematic chains with multiple degrees of freedom. More specifically, robot kinematics provide the transformation from the joint space, where the kinematic chains are defined, to the Cartesian space, where the robot manipulator moves, and vice versa. Robot kinematics are quite useful, because they can be used for planning and executing movements, as well as calculating actuator forces and torques.

### 2.3.1 Forward Kinematics

The joint space reveals very little information about the position and orientation of the end effector of the kinematic chain. The *forward kinematics* define a mapping from the joint space to the three-dimensional Cartesian space. Given a kinematic chain with $m$ joints and a set of joint values $(\theta_1, \theta_2, \ldots, \theta_m)$, the forward kinematics can find the position $(p_x, p_y, p_z)$ and the orientation $(a_x, a_y, a_z)$ of the end effector of the kinematic chain in the three-dimensional $x$-$y$-$z$ space. Forward kinematics is a domain-independent problem and can be solved for any simple or complex kinematic chain yielding a closed-form, analytical solution.

### 2.3.2 Inverse Kinematics

Robot manipulators typically need to reach target points or follow trajectories in the three-dimensional space. To make the end effector reach a point or follow a trajectory, one has to specify appropriate values for the joints of the kinematic chain. The *inverse kinematics* define ways to go from the three-dimensional space to the joint space. In particular, the inverse kinematics define a relation between points in the three-dimensional space (position $(p_x, p_y, p_z)$ and orientation $(a_x, a_y, a_z)$) and joint values/angles $(\theta_1, \theta_2, \ldots, \theta_m)$ in the joint space of a kinematic chain with $m$ joints. The problem of inverse kinematics is domain-dependent and every kinematic chain has a different solution. The solution to the inverse kinematics problem can lead to an analytical, closed-form equation or to a numerical, iterative approximation (e.g. with the Jacobian approximation method). As the number of DOF increases, a point in the three-dimensional space may have more than one matching points in the joint space. This multiplicity of solutions makes the inverse kinematics a relation, not a mapping.

## 2.4 Affine Transformations

An affine transformation is a mapping that transforms points and vectors from one space to another, in a way that preserves the ratios of distances. The source and target spaces can be $n$-dimensional with $n \geq 2$. The following are affine transformations: geometric contraction, expansion, dilation, reflection, rotation, shear, similarity transformations, spiral similarities, and translation. All the possible combinations of the above produce

an affine transformation as well. The flexibility of affine transformations with respect to object representation in different spaces, makes it a very useful tool in computer graphics.

For the purposes of this thesis we only use rotation and translation, so we will focus only on these two types of affine transformation. Additionally, we are working in a three-dimensional Cartesian work space and therefore all the definitions and examples from now on will focus on this space.

### Affine Transformation Matrix

An affine transformation matrix is a $\big((n+1)\times(n+1)\big)$ matrix, where $n$ is the number of dimensions in the space the transformation is defined on. In general, an affine transformation matrix is a block matrix of the form:

$$T = \begin{bmatrix} & X & & Y \\ [0 & \cdots & 0] & 1 \end{bmatrix}$$

where $X$ is a $(n \times n)$ matrix, $Y$ is a $(n \times 1)$ vector and the last line of $T$ contains $n-1$ zeros followed by a 1. If we want to apply the transformation, to a given point $p = (p_1, p_2, \ldots, p_n)$ in the $n$-dimensional space, we simply multiply the affine transformation matrix with the column vector $v = (p_1, p_2, \ldots, p_n, 1)^\top$:

$$v' = \begin{bmatrix} p'_1 \\ \vdots \\ p'_n \\ 1 \end{bmatrix} = Tv = \begin{bmatrix} & X & & Y \\ [0 & \cdots & 0] & 1 \end{bmatrix} \begin{bmatrix} p_1 \\ \vdots \\ p_n \\ 1 \end{bmatrix}$$

For a point $p = (p_x, p_y, p_z)$ in the three-dimensional space, the transformation will be:

$$v' = \begin{bmatrix} p'_x \\ p'_y \\ p'_z \\ 1 \end{bmatrix} = Tv = \begin{bmatrix} X_{xx} & X_{xy} & X_{xz} & Y_x \\ X_{yx} & X_{yy} & X_{yz} & Y_y \\ X_{zx} & X_{zy} & X_{zz} & Y_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$$

The matrix that results from the multiplication of two affine transformation matrices $T_1$ and $T_2$ is still an affine transformation:

$$T = T_1 T_2 = \begin{bmatrix} & X_1 & & Y_1 \\ [0 & \cdots & 0] & 1 \end{bmatrix} \begin{bmatrix} & X_2 & & Y_2 \\ [0 & \cdots & 0] & 1 \end{bmatrix} = \begin{bmatrix} & X_1 X_2 & & X_1 Y_2 + Y_1 \\ [0 & \cdots & 0] & 1 \end{bmatrix}$$

This property generalizes to the product of any number of affine transformation matrices:

$$\widehat{T} = T_1 T_2 T_3 \cdots T_k = \begin{bmatrix} & \widehat{X} & & \widehat{Y} \\ \begin{bmatrix} 0 & \cdots & 0 \end{bmatrix} & & & 1 \end{bmatrix}$$

An affine transformation matrix is invertible, if and only if X is invertible, and takes the form:

$$T^{-1} = \begin{bmatrix} & X^{-1} & & -X^{-1}Y \\ \begin{bmatrix} 0 & \cdots & 0 \end{bmatrix} & & & 1 \end{bmatrix}$$

### Translation Matrix

Translation in a Cartesian space is a function that moves (translates) every point by a fixed distance in a specified direction. We can describe a translation in the three-dimensional space with a $(4 \times 4)$ matrix of the following form:

$$A = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where $d_x$, $d_y$, and $d_z$ define the distance of translation along the $x$, $y$, and $z$ axis respectively. Apparently, the translation matrix is an affine transformation matrix with $X = I$. Therefore, to move a point $p = (p_x, p_y, p_z)$ in the three-dimensional space by distances $(d_x, d_y, d_z)$, we simply apply the transformation:

$$v' = \begin{bmatrix} p'_x \\ p'_y \\ p'_z \\ 1 \end{bmatrix} = Av = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$$

### Rotation Matrix

Rotation in a Cartesian space is a function that rotates vectors by a fixed angle about a specified direction. A rotation in the $n$-dimensional space is described as an $(n \times n)$ orthogonal matrix $R$ with determinant 1:

$$R^\top = R^{-1} \qquad RR^\top = R^\top R = I \qquad \det(R) = 1$$

In the three-dimensional Cartesian space there are three distinct rotation matrices, each one of them performing a rotation of $\theta_x$, $\theta_y$, $\theta_z$ about the $x$, $y$, $z$ axis respectively, assuming a right-handed coordinate system:

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x \\ 0 & \sin\theta_x & \cos\theta_x \end{bmatrix} \quad R_y = \begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y \\ 0 & 1 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y \end{bmatrix} \quad R_z = \begin{bmatrix} \cos\theta_z & -\sin\theta_z & 0 \\ \sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

To rotate a vector defined by the end point $p = (p_x, p_y, p_z)$ about a specific axis, one can simply multiply with the corresponding rotation matrix. To rotate the vector first about the $x$ axis and then about the $y$ axis, one has to multiply with the corresponding rotation matrices in the following order:

$$p' = \begin{bmatrix} p'_x \\ p'_y \\ p'_z \end{bmatrix} = R_y R_x p = \begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y \\ 0 & 1 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x \\ 0 & \sin\theta_x & \cos\theta_x \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$$

Apparently, all three rotation matrices can be combined to form new rotation matrices to perform complex rotations about all dimensions. For example, the rotation matrix that rotates vectors first about the $x$ axis, then about the $y$ axis, and finally about the $z$ axis is the following:

$$R' = R_z R_y R_x$$

The analytical form of the above rotation matrix is the following:

$$R' = \begin{bmatrix} \cos\theta_y \cos\theta_z & -\cos\theta_x \sin\theta_z + \sin\theta_x \sin\theta_y \cos\theta_z & \sin\theta_x \sin\theta_z + \cos\theta_x \sin\theta_y \cos\theta_z \\ \cos\theta_y \sin\theta_z & \cos\theta_x \cos\theta_z + \sin\theta_x \sin\theta_y \sin\theta_z & -\sin\theta_x \cos\theta_z + \cos\theta_x \sin\theta_y \sin\theta_z \\ -\sin\theta_y & \sin\theta_x \cos\theta_y & \cos\theta_x \cos\theta_y \end{bmatrix}$$

We can easily transform any rotation matrix $\widehat{R}$ to an affine transformation matrix $R$ just by padding the last line and the last column with $(0, \ldots, 0, 1)$:

$$R = \begin{bmatrix} & & & \\ & \widehat{R} & & \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \\ & & & \\ \begin{bmatrix} 0 & \cdots & 0 \end{bmatrix} & & & 1 \end{bmatrix}$$

From now on, any rotation matrix will be an affine transformation matrix.

### Affine Transformation Matrices and Kinematics

For the purposes of kinematics, we are using rotation and translation matrices, so that we can transform points in the three-dimensional space. We consider affine transformation matrices that combine rotation and translation; the $X$ block of the matrix defines the rotation, while the $Y$ block of the matrix defines the translation:

$$T = \begin{bmatrix} & \widehat{R} & & \begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix} \\ \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} & & & 1 \end{bmatrix}$$

## 2.5 Denavit-Hartenberg (DH) Parameters

Denavit and Hartenberg [**?**, **?**] have devised a way of creating a transformation matrix that describes points in one end of a joint to a coordinate system that is fixed to the other end of the joint, as a function of the joint state. They concluded that we can fully describe this transformation matrix using only four parameters, known as Denavit-Hartenberg (DH) parameters: $a$, $\alpha$, $d$, and $\theta$. Before we can explain these parameters we must first establish the reference frame of each joint $i$ with respect to the reference frame of its previous joint:

- The $z_i$-axis is set to the direction of the joint axis (the rotation direction).
- The $x_i$-axis is parallel to the common normal between $z_i$ and $z_{i-1}$ (exterior product). The direction of $x_i$ is derived using the right-hand rule from $z_{i-1}$ to $z_i$.
- The $y_i$-axis follows from the $x_i$ and $z_i$ axes to form a right-handed coordinate system.

Now, we can describe the DH parameters [**?**] (cf. Figure 2.4):

- $a$: length of the common normal
- $\alpha$: angle about the common normal, from $z_{i-1}$-axis to $z_i$-axis
- $d$: offset along the $z_{i-1}$-axis to the common normal
- $\theta$: angle about the $z_{i-1}$-axis, from $x_{i-1}$-axis to $x_i$-axis

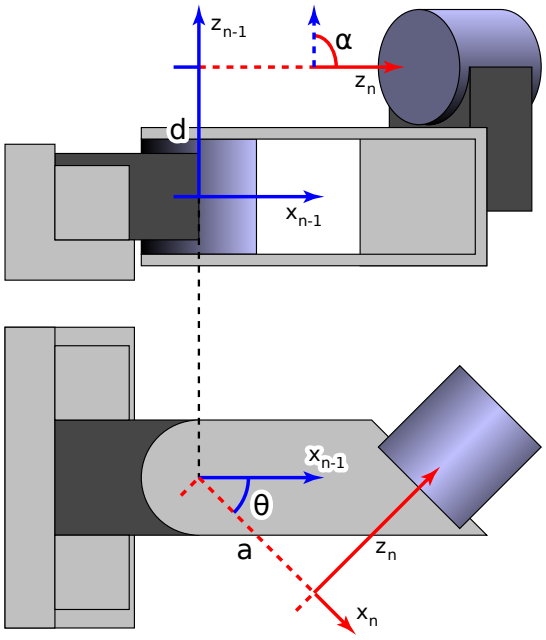



Figure 2.4: Denavit-Hartenberg (DH) parameters: $a$, $\alpha$, $d$, $\theta$ (from www.tekkotsu.org)

The Kinematics section in the documentation of the Tekkotsu framework [**?**] is a great resource with text, figures, and videos for understanding the role of these parameters and how they are found.

Now, we can move from the base reference frame of some joint to the transformed reference frame of this joint using the transformation matrix $T_{DH}$, which consists of two translations and two rotations parametrized by the DH parameters of the joint:

$$T_{DH} = R_x(\alpha)T_x(a)R_z(\theta)T_z(d)$$

The analytical form of the resulting matrix from the above composition is the following:

$$T_{DH} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & a \\ \sin\theta\cos\alpha & \cos\theta\cos\alpha & -\sin\alpha & -d\sin\alpha \\ \sin\theta\cos\alpha & \cos\theta\sin\alpha & \cos\alpha & d\cos\alpha \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

It is easy to see that the matrix above is an affine transformation matrix, because it is the product of affine transformation matrices.

## 2.6 Mathematica

Mathematica© is a software tool for mathematical computations created by the Wolfram company (www.wolfram.com). This tool is widely-used, because, among other things, it can easily find solutions to differential equations and can perform symbolic computations. For the purposes of this thesis, we exploited its capability to perform large-scale symbolic computations with matrices and simplify symbolic expressions in reasonable time.

The following code excerpt is a small example of symbolic computation with Mathematica. We construct two matrices with cosines and sines containing two symbols, `theta1` and `theta2`. Next, we multiply those two matrices symbolically and finally we simplify the result:

```
Matrix1 = {{Cos[theta1], -Sin[theta1]}, {Cos[theta1], -Cos[theta1]}};
Matrix2 = {{Cos[theta2], -Sin[theta2]}, {Cos[theta2], -Cos[theta2]}};
T = Matrix1.Matrix2;
Simplify[T];
MatrixForm[%]
```

The result of these symbolic computations is the following:

$$\text{Matrix1} = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 \\ \cos\theta_1 & -\cos\theta_1 \end{bmatrix}$$

$$\text{Matrix2} = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 \\ \cos\theta_2 & -\cos\theta_2 \end{bmatrix}$$

$$\text{T} = \begin{bmatrix} \cos\theta_1\cos\theta_2 - \cos\theta_2\sin\theta_1 & \cos\theta_2\sin\theta_1 - \cos\theta_1\sin\theta_2 \\ 0 & \cos\theta_1\cos\theta_2 - \cos\theta_1\sin\theta_2 \end{bmatrix}$$

$$\text{T}_{\text{simplified}} = \begin{bmatrix} \cos\theta_2\left(\cos\theta_1 - \sin\theta_1\right) & \sin\left(\theta_1 - \theta_2\right) \\ 0 & \cos\theta_1\left(\cos\theta_2 - \sin\theta_2\right) \end{bmatrix}$$

The example above is quite simple, but fully illustrates the symbolic capabilities of Mathematica and particularly the simplification step, which is very important for our work, given that we have to deal with much larger and more complex matrices.

# Chapter 3

# NAO Kinematics: The Problem

## 3.1 NAO Robot Specifications

Aldebaran NAO is a humanoid robot with five kinematic chains (head, two arms, two legs). It is 58cm tall and it has about 5kg of mass. The version we are working on is the RoboCup edition v3.3 with 21 DOF. NAO has two DOF on the head, four DOF on each arm, five DOF on each leg, and one DOF in the pelvis, which is shared between the two legs. The five kinematic chains and their joints are the following:

**Head:** HeadYaw, HeadPitch

**Left Arm:** LShoulderPitch, LShoulderRoll, LElbowYaw, LElbowRoll

**Right Arm:** RShoulderPitch, RShoulderRoll, RElbowYaw, RElbowRoll

**Left Leg:** LHipYawPitch, LHipRoll, LHipPitch, LKneePitch, LAnklePitch, LAnkleRoll

**Right Leg:**RHipYawPitch, RHipRoll, RHipPitch, RKneePitch, RAnklePitch, RAnkleRoll

The joints LHipYawPitch and RHipYawPitch are just different names for the shared (common) joint (HipYawPitch) between the two legs. Figure 3.1 shows the physical arrangement of the five chains and their joints on the NAO robot (Academic edition). Note that the RoboCup edition of the NAO robot is missing four DOF from the two hands (LWristYaw, LHand, RWristYaw, and RHand).

To fully specify the joints of the NAO robot, we give the length of all the links of the robot (Table 3.2), the operational range in radians and degrees of the head joints

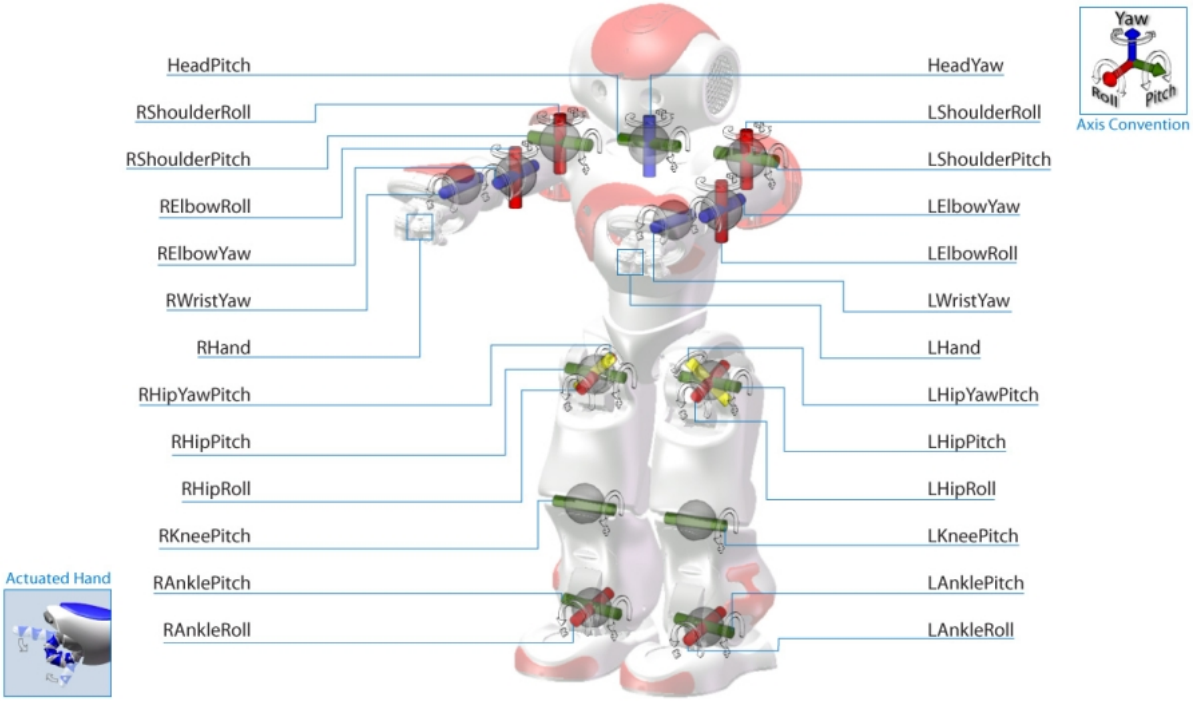
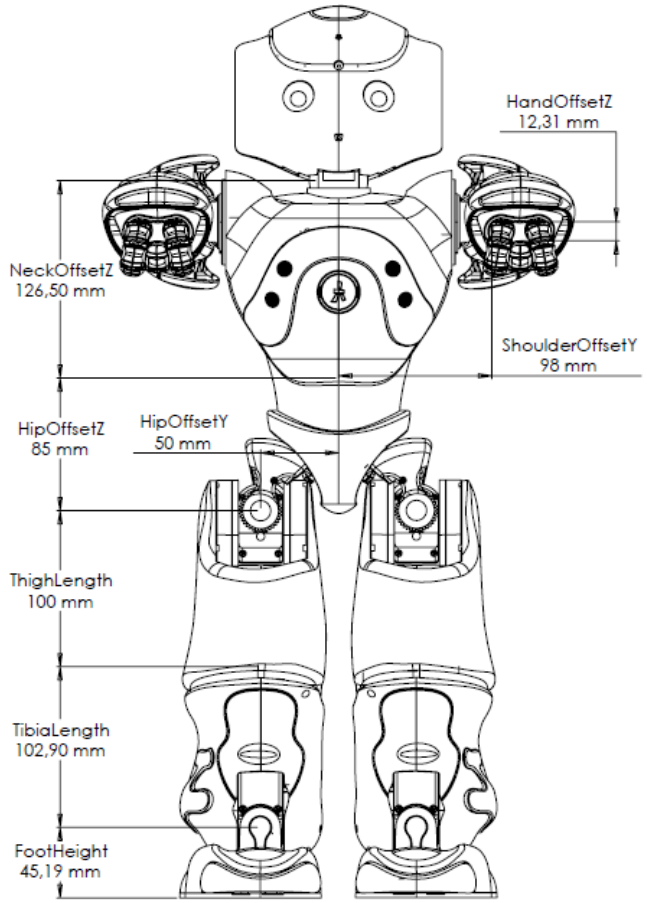



Figure 3.1: Aldebaran NAO v3.3 (Academic edition) kinematic chains and joints

(Figures 3.3), the arm joints (Figure 3.4), and the leg joints (Figure 3.5), as well as the mass of each joint/link (Table 3.1). These values have been extracted from the documentation [**?**] provided by the manufacturer of the robot, Aldebaran Robotics. The center of mass for each link/joint is represented by a point in the three-dimensional space of that joint assuming a zero posture of that joint. The documentation gives mass values only for the right part of the robot; we assume that the robot is fully symmetric with respect to the sagittal plane to obtain the masses for the left part. In general, the robot is supposed to be fully symmetric, but interestingly, according to the manufacturer, some joints on the left side have a different range than the corresponding joints on the right side. Additionally, although some joints appear to be able to move within a large range, the hardware controller of the robot prohibits access to the extremes of these ranges, because of possible collisions with the NAO shell.




| Name | Size (mm) |
|---|---|
| NeckOffsetZ | 126.50 |
| ShoulderOffsetY | 98.00 |
| ElbowOffsetY | 15.00 |
| UpperArmLength | 105.00 |
| LowerArmLength | 55.95 |
| ShoulderOffsetZ | 100.00 |
| HandOffsetX | 57.75 |
| HipOffsetZ | 85.00 |
| HipOffsetY | 50.00 |
| ThighLength | 100.00 |
| TibiaLength | 102.90 |
| FootHeight | 45.19 |
| HandOffsetZ | 12.31 |

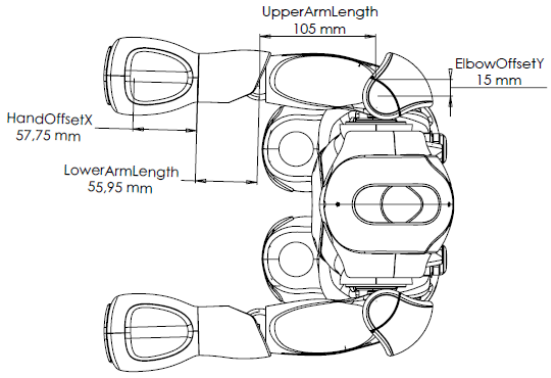
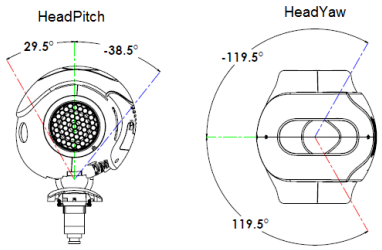
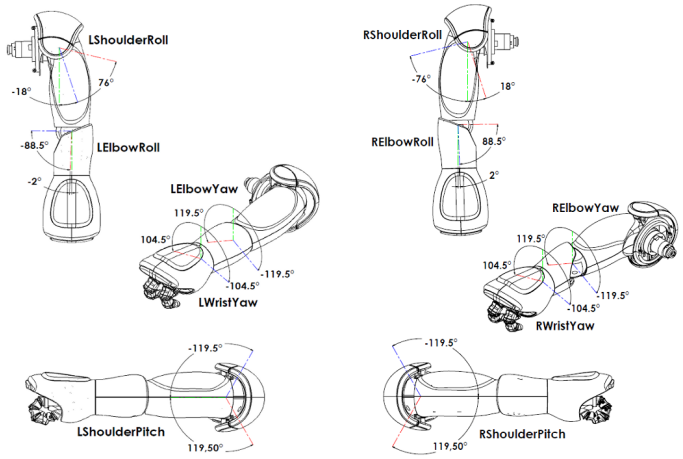



Figure 3.2: NAO links and their sizes



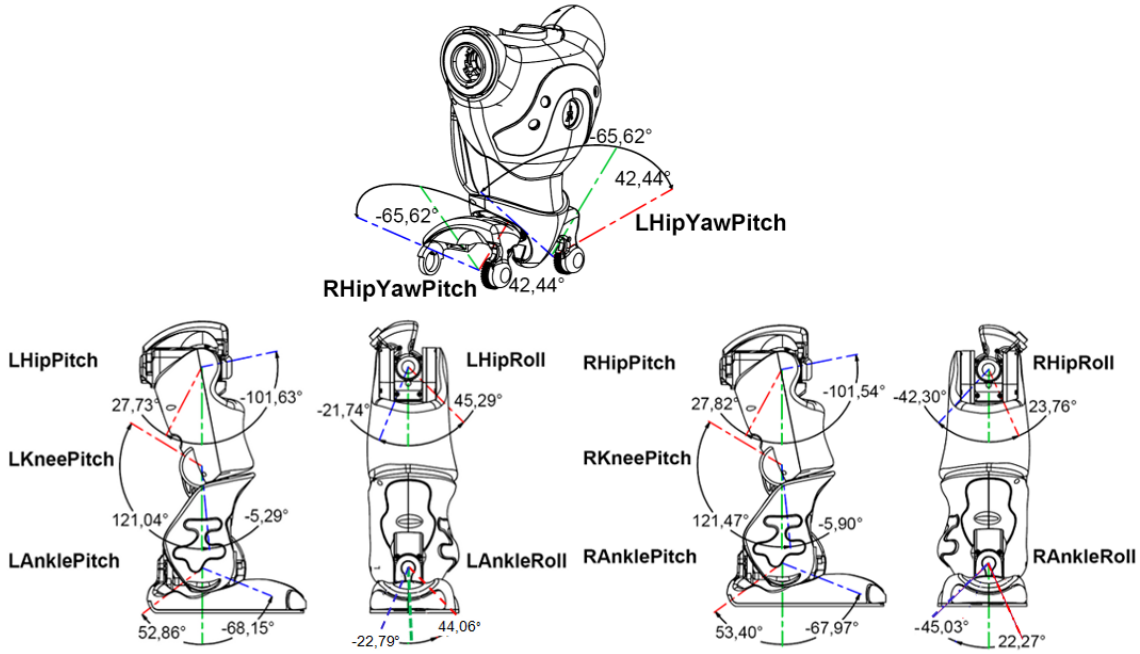

| Joint Name | Range in Degrees° | Range in Radians |
|---|---|---|
| HeadYaw | -119.5° to 119.5° | -2.0857 to 2.0857 |
| HeadPitch | -38.5° to 29.5° | -0.6720 to 0.5149 |

Figure 3.3: NAO head joints and their operational range




| Joint Name | Range in Degrees° | Range in Radians |
|---|---|---|
| LShoulderPitch | -119.5° to 119.5° | -2.0857 to 2.0857 |
| LShoulderRoll | -18° to 76° | -0.3142 to 1.3265 |
| LElbowYaw | -119.5° to 119.5° | 1.5446 to 0.0349 |
| LElbowRoll | -88.5° to -2° | -0.6720 to 0.5149 |
| RShoulderPitch | -119.5° to 119.5° | -2.0857 to 2.0857 |
| RShoulderRoll | -38.5° to 29.5° | -1.3265 to 0.3142 |
| RElbowYaw | -119.5° to 119.5° | -2.0857 to 2.0857 |
| RElbowRoll | -38.5° to 29.5° | 0.0349 to 1.5446 |
| LWristYaw and RWristYaw | disabled | disabled |

Figure 3.4: NAO arms joints and their operational range




| Joint Name | Range in Degrees° | Range in Radians |
|---|---|---|
| LHipYawPitch-RHipYawPitch | -65.62 to 42.44 | -1.145303 to 0.740810 |
| LHipRoll | -21.74° to 45.29° | -0.379472 to 0.790477 |
| LHipPitch | -101.63° to 27.73° | -1.773912 to 0.484090 |
| LKneePitch | -5.29° to 121.04° | -0.092346 to 2.112528 |
| LAnklePitch | -68.15° to 52.86° | -1.189516 to 0.922747 |
| LAnkleRoll | -22.79° to 44.06° | -0.397880 to 0.769001 |
| RHipRoll | -42.30° to 23.76° | -0.738321 to 0.414754 |
| RHipPitch | -101.54° to 27.82° | -1.772308 to 0.485624 |
| RKneePitch | -5.90° to 121.47° | -0.103083 to 2.120198 |
| RAnklePitch | -67.97° to 53.40° | -1.186448 to 0.932056 |
| RAnkleRoll | -45.03° to 22.27° | -0.785875 to 0.388676 |

Figure 3.5: NAO legs joints and their operational range

Table 3.1: Masses of links/joints (frames) of the NAO robot

| **Masses for NAO v3.3 RoboCup edition** | | | | |
|---|---|---|---|---|
| **Frame Name** | **Mass (Kg)** | **CoM$_x$ (mm)** | **CoM$_y$ (mm)** | **CoM$_z$ (mm)** |
| Torso | 1.03948 | -4.15 | 0.07 | 42.58 |
| HeadYaw | 0.05930 | -0.02 | 0.17 | 25.56 |
| HeadPitch | 0.52065 | 1.2 | -0.84 | 53.53 |
| RShoulderPitch | 0.06996 | -1.78 | 24.96 | 0.18 |
| RShoulderRoll | 0.12309 | 18.85 | -5.77 | 0.65 |
| RElbowYaw | 0.05971 | -25.6 | 0.01 | -0.19 |
| RElbowRoll | 0.185 | 65.36 | -0.34 | -0.02 |
| LShoulderPitch | 0.06996 | -1.78 | -24.96 | 0.18 |
| LShoulderRoll | 0.12309 | 18.85 | 5.77 | 0.65 |
| LElbowYaw | 0.05971 | -25.6 | -0.01 | -0.19 |
| LElbowRoll | 0.185 | 65.36 | 0.34 | -0.02 |
| RHipYawPitch | 0.07117 | -7.66 | 12 | 27.17 |
| RHipRoll | 0.1353 | -16.49 | -0.29 | -4.75 |
| RHipPitch | 0.39421 | 1.32 | -2.35 | -53.52 |
| RKneePitch | 0.29159 | 4.22 | -2.52 | -48.68 |
| RAnklePitch | 0.13892 | 1.42 | -0.28 | 6.38 |
| RAnkleRoll | 0.16175 | 25.4 | -3.32 | -32.41 |
| LHipYawPitch | 0.07117 | -7.66 | -12 | 27.17 |
| LHipRoll | 0.1353 | -16.49 | 0.29 | -4.75 |
| LHipPitch | 0.39421 | 1.32 | 2.35 | -53.52 |
| LKneePitch | 0.29159 | 4.22 | 2.52 | -48.68 |
| LAnklePitch | 0.13892 | 1.42 | 0.28 | 6.38 |
| LAnkleRoll | 0.16175 | 25.4 | 3.32 | -32.41 |
| **Total Mass** | **4.88083** | | | |

## 3.2 The Kinematics Problem for NAO

The NAO robot has a large number of DOF, therefore it can perform several complex moves. Some examples of such moves are walking, kicking a ball, standing up, etc. Kinematics are quite useful for NAO software developers, because they can be used for

planning and executing such complex movements. For example, using forward kinematics and the current joint values, one can easily find the exact position and orientation of the camera with respect to the floor the robot is standing on and therefore determine the horizon in the camera view. Likewise, using inverse kinematics, one can easily follow planned trajectories with one foot, while standing on the other, to perform dynamic kick motions.

### 3.2.1 The Forward Kinematics Problem for NAO

The forward kinematics problem is to define a mapping from the joint space of the robot to the three-dimensional space with respect to any base coordinate frame. All joints of the NAO robot are equipped with 12-bit encoders, which are updated at a frequency of 100Hz, and therefore the current joint values are readily available at any time. Despite the 21 DOF, the forward kinematics problem can be easily decomposed because three of the five kinematic chains (the head and the two arms) are completely independent and two of them (the two legs) only have one common joint. Given that in forward kinematics we do not affect the values of the joints, but only read the current state of each joint, we can assume that even the two legs chains are completely independent. Therefore, forward kinematics for NAO can been seen as five independent problems with corresponding solutions, one for each kinematic chain. Each of these solutions provides the exact point (position and orientation) in the three-dimensional space with respect to any base coordinate frame of any end effector along the corresponding kinematic joint. These solutions can be combined to obtain a solution for a bigger kinematic chain formed by any combination of the five independent chains (e.g. the kinematic chain from the right foot to the head or the kinematic chain from the left foot to the right hand).

The importance of solving the forward kinematics problem for NAO is twofold: apart from the ability to locate the exact position and orientation of any end effector of the robot, it provides the means to calculate the center of mass of the robot for the current configuration, which is most-needed for balancing. In addition, as we shall see later, solution to the inverse kinematics problem would be intractable without solving the forward kinematics problem first.

### 3.2.2 The Inverse Kinematics Problem for NAO

The inverse kinematics problem is to define ways to go from the three-dimensional space of the robot to the joint space. In particular, it defines a relation between points in the three-dimensional space (position and orientation) and joint values in the joint space of a kinematic chain. For the reasons stated above, the inverse kinematics problem can be decomposed again into five independent problems. The coupling between the two legs due to the common joint is initially ignored. This is a required assumption to make the problem solvable; the obtained solutions can be combined in different ways to form a unique solution. A solution to each of these independent problems can provide the joint values which place the end effector of the corresponding kinematic chain to a specific point in the three-dimensional space of the robot torso.

Inverse kinematics represents a much more difficult problem compared to forward kinematics for at least two reasons. First, it leads to a system of non-linear equations, which may, or may not, have an analytical solution. Second, as the number of DOF increases and the kinematic chain becomes more flexible, a point in the three-dimensional space may have more than one matching points in the joint space of the chain. This multiplicity of solutions defines a complex relation, but not a mapping, between the two spaces.

The importance of solving the inverse kinematics problem for NAO lies in the ability to follow any (predefined or dynamically-generated) trajectory in the three-dimensional space with any of the five end effectors. Inverse kinematics essentially provide the mechanism to transform such a trajectory into another trajectory in the joint space of the robot.

The inverse kinematics problem can be solved analytically with closed-form equations or numerically with an iterative approximation method [**?**]. The analytical solution is in general faster than the fastest numerical solution and therefore is more appropriate for real-time execution. Numerical solutions are also subject to singularities, which result in a failure to obtain a solution, even if one exists. Additionally, numerical solutions are iterative; for real-time execution the number of iterations is limited and therefore they may fail to converge. For these reasons, we aim to find an analytical solution to the inverse kinematics problem for the NAO robot. It is well-known that inverse kinematics can be obtained analytically, if the chain has five or less DOF. If the chain has six DOF,

an analytical solution can be obtained, only if three consecutive joints have intersecting axes. Three of the kinematics chains of the NAO robot have less than five DOF. The legs have six DOF, however the meet the condition given above because the three hip joints have intersecting axes. Therefore, it is possible to obtain a fully analytical solution for the inverse kinematics of the NAO.

## 3.3 Related Work

The problem of forward and inverse kinematics for the NAO robot is a familiar problem to all the teams participating in RoboCup SPL. The solution to forward kinematics is quite straightforward and most teams have implemented their own code for forward kinematics computations. There are only a few known solutions for the problem of inverse kinematics. We review existing related work in the next few sections.

### 3.3.1 Aldebaran Robotics

Aldebaran Robotics provides a forward kinematics mechanism integrated within the proprietary NaoQi middleware for the NAO robot. However, this mechanism does not accept any input and provides a solution only for the current joint configuration of the robot. As a result, it is impossible to run Aldebaran's forward kinematics for a specific set of joint values, for example the joint values recorded when a specific picture was acquired from the camera. The ability to provide any set of joints is important, not only for finding the position of the camera in the three-dimensional at specific times, but also for verifying candidate solutions returned by inverse kinematics. On the other hand, Aldebaran provides the DH parameters for all the joints of the NAO robot and that was very useful.

Aldebaran Robotics provides an inverse kinematics mechanism integrated within the proprietary NaoQi middleware for the NAO robot. These functions in the API of the robot can move the end effector of a kinematic chain to a given point in the three-dimensional space. The method used to provide the solution is based on the Jacobian iterative approximation method. Furthermore, the omni-directional walk engine provided by Aldebaran Robotics uses this approach to follow planned foot trajectories. Although the resulting solutions are in most cases accurate, the method can easily fall into sin-

gularities; if that happens, the robot gets stuck in a specific configuration. Singularities present a serious problem with possible catastrophic consequences for the robot.

### 3.3.2 B-Human Inverse Kinematics Solution

B-Human is the RoboCup SPL team of the University of Bremen in Germany. Each year they publish a code release, which includes the full code they used in the last RoboCup and a documentation for this code. In their recent code release [**?**] they include an inverse kinematics solution for the legs of NAO, albeit with under certain simplification assumptions and approximations. The solution provided always makes the foot parallel to the plane defined by the $z$-axis and the $x$-axis of the torso. If the target point violates this assumption, the solution will reach the target position, but will ignore the target orientation, and therefore it will only be an approximate solution.

### 3.3.3 QIAU Inverse Kinematics Solution

MRL is the RoboCup SPL team of the Qazvin Islamic Azad University (QIAU) in Tehran, Iran. They have published [**?**] an analytical solution for the problem of inverse kinematics for the legs. We have tried to implement their solution, but unfortunately we were not able to reproduce their results.

# Chapter 4

# NAO Kinematics: The Solution

As mentioned in Chapter **??**, the existing approaches to kinematics for the NAO robot are not completely suitable for our needs. We seek to find a solution to the forward kinematics problem for any set of joint values as input and not only for the current joint values. In addition, we seek to find a real-time analytical solution for the problem of inverse kinematics without any approximations. In the sections below, we describe our solutions to both of these problems.

## 4.1 Forward Kinematics for the NAO Robot

Aldebaran Robotics provides the DH parameters for each kinematic chain of the robot in the documentation [**?**]. However, our experimentation with the provided values revealed that the given parameters for the arm chains are incorrect. Therefore, we found our own parameters for the arms and we used the provided parameters for the legs and the head.

### NAO Zero Position

We must define the base frame of the robot and the zero position of the joints before we proceed. The base frame is taken to be the torso frame; Figure 4.1 shows the axes of this frame. The same figure shows also the zero position of all the joints of the robot, which is the one provided by Aldebaran Robotics. As we can see, in this position the ShoulderRoll joints are not really roll joints, but are yaw joints, so we can understand

that the names of the joints do not necessarily describe the actual movement of the joint in the base frame given the zero position.

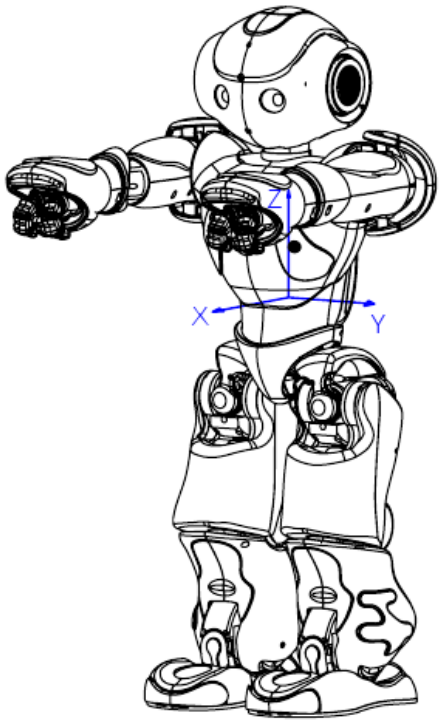



Figure 4.1: Base (torso) frame and zero position of the joints

### Notation

We provide a brief description of the symbols we use in our math calculations. All matrices used are affine transformation matrices of three types: $T$ is a transformation matrix, $R_x, R_y, R_z$ are basic rotations matrices, and $A$ is a translation matrix. The subscript of a symbol refers to the start frame and the superscript refers to the destination frame. The torso is the point where all the kinematic chains begin and is located at the center of the NAO body. "Base" is the start frame of the chain (the torso frame), while "End" is the end effector of the chain. The numbers appearing as subscripts or superscripts refer to the joints in the current kinematic chain, numbered consistently

with the ordering given in the tables of Chapter 3. Also, we denote the initialization of a translation matrix as $A(d_x, d_y, d_z)$ and of rotation matrices as $R_x(\theta_x)$, $R_y(\theta_y)$, or $R_z(\theta_z)$.

We present the DH parameters of each kinematic chain in a separate table and, besides the DH parameters, we provide the translations from the "Base" to the first joint and from the last joint to the "End". Finally, we provide some necessary rotations to adjust the frame of the last joint to the frame of the end effector ("End").

### Forward Kinematics Equations

Forward kinematics for each chain of the NAO robot is a transformation that maps a point from the frame of the last joint to the base frame. In our case, the end effector is the point of interest. Forward kinematics are defined in terms of transformation, rotation, and translation matrices, and the final result is a single transformation matrix that maps points from one frame to another.

### Extracting the Point in the Three-Dimensional Space

The result of forward kinematics is an affine transformation matrix $T$ with the $X$ block being a rotation matrix and the $Y$ block being a translation vector. We need to extract the $(p_x, p_y, p_z)$ position and the $(a_x, a_y, a_z)$ orientations of the final point. The position $(p_x, p_y, p_z)$ can be simply read off the translation part of the transformation matrix:

$$\begin{aligned} p_x &= T_{(1,4)} \\ p_y &= T_{(2,4)} \\ p_z &= T_{(3,4)} \end{aligned}$$

The rotation of the final transformation table is a $R_z R_y R_x$ rotation table, whose analytical form is shown in Section 2.4. Now it's easy to extract the orientation $(a_x, a_y, a_z)$:

$$\begin{aligned} a_x &= \text{arctan2}\left(T_{(3,2)}, T_{(3,3)}\right) \\ a_y &= \text{arctan2}\left(-T_{(3,1)}, \sqrt{{T_{(3,2)}}^2 + {T_{(3,3)}}^2}\right) \\ a_z &= \text{arctan2}\left(T_{(2,1)}, T_{(1,1)}\right) \end{aligned}$$

Table 4.1: DH parameters for the head chain of the NAO robot

| **Frame (Joint)** | **a** | $\boldsymbol{\alpha}$ | **d** | $\boldsymbol{\theta}$ |
|---|---|---|---|---|
| Base | $A(0, 0, \text{NeckOffsetZ})$ | | | |
| HeadYaw | 0 | 0 | 0 | $\theta_1$ |
| HeadPitch | 0 | $-\frac{\pi}{2}$ | 0 | $\theta_2 - \frac{\pi}{2}$ |
| Rotation | $R_x(\frac{\pi}{2})R_y(\frac{\pi}{2})$ | | | |
| Top Camera | $A(\text{topCameraX}, 0, \text{topCameraZ})$ | | | |
| Bottom Camera | $A(\text{bottomCameraX}, 0, \text{bottomCameraZ})$ | | | |
| topCameraX=53.9mm, topCameraZ=67.9mm, bottomCameraX=48.8mm, bottomCameraZ=23.8mm | | | | |

### 4.1.1 Forward Kinematics for the Head

The head is the simplest kinematic chain of the NAO robot, but it has two useful end effectors, namely the top and the bottom cameras. Table 4.1 shows the DH parameters for the head chain. Now, we can combine these matrices to find the point of the end effector in the frame space of the torso:

$$T^{\text{End}}_{\text{Base}} = A^0_{\text{Base}} T^1_0 T^2_1 R_x(\tfrac{\pi}{2}) R_y(\tfrac{\pi}{2}) A^{\text{End}}_2$$

$T^1_0$ and $T^2_1$ are the DH transformation matrices of the corresponding joints (HeadYaw, HeadPitch). $A^{\text{End}}_2$ is one of the two translation matrices given in Table 4.1 for the two end effectors (top and bottom camera). The point of the end effector in the three-dimensional space of the torso can be extracted from $T^{\text{End}}_{\text{Base}}$.

### 4.1.2 Forward Kinematics for the Left Arm

The kinematic chain for the left arm consists of four joints. So, we need to find four sets of DH parameters, one for each joint. First, we must move from the torso to the base of the joint and we can do that with a simple translation along the $y$-axis and the $z$-axis. After that, we must align the coordinate frame with the rotation axis of the first joint (LShoulderPitch). So, we rotate about the $x$-axis of the coordinate frame by $-\frac{\pi}{2}$, thus the $\alpha$ parameter for LShoulderPitch is $-\frac{\pi}{2}$, while $d$, $a$ are 0. Now, we must rotate the coordinate frame again to become aligned with the rotation axis of the second joint (LShoulderRoll). So, we rotate about the $x$-axis by $\frac{\pi}{2}$, thus the $\alpha$ parameter for

Table 4.2: DH parameters for the left arm chain of the NAO robot

| **Frame (Joint)** | **a** | $\boldsymbol{\alpha}$ | **d** | $\boldsymbol{\theta}$ |
|---|---|---|---|---|
| Base | $A(0, \text{ShoulderOffsetY}, \text{ShoulderOffsetZ})$ | | | |
| LShoulderPitch | 0 | $-\frac{\pi}{2}$ | 0 | $\theta_1$ |
| LShoulderRoll | 0 | $\frac{\pi}{2}$ | 0 | $\theta_2 + \frac{\pi}{2}$ |
| LElbowYaw | ElbowOffsetY | $\frac{\pi}{2}$ | UpperArmLength | $\theta_3$ |
| LElbowRoll | 0 | $-\frac{\pi}{2}$ | 0 | $\theta_4$ |
| Rotation | $R_z(-\frac{\pi}{2})$ | | | |
| End effector | $A(\text{HandOffsetX+LowerArmLength}, 0, 0)$ | | | |

LShoulderRoll is $\frac{\pi}{2}$, while $d$, $a$ are 0. Next, we need to align the coordinate frame with the rotation axis of the third joint (LElbowYaw). To do so, we must rotate about the $y$-axis. The DH parameters do not directly encode a rotation about the $y$-axis, so we must first rotate about the $z$-axis and then about the $x$-axis to effectively realize a rotation about the $y$-axis. Thus, we add $\frac{\pi}{2}$ to the angle $\theta_2$ of the previous joint (to rotate about the $z$-axis) and then rotate about the $x$-axis by $\frac{\pi}{2}$ (the $\alpha$ parameter of LElbowYaw). Also, we must move ElbowOffsetY towards the $x_2$ axis so the parameter $a$ is ElbowOffsetY. Then, we move along the $z$-axis to reach the position of the LElbowYaw joint, so its $d$ parameter is set to UpperArmLength. Finally, for the fourth joint (LElbowRoll) we rotate about the $x$-axis by $-\frac{\pi}{2}$, thus the $\alpha$ parameter for LElbowRoll is $-\frac{\pi}{2}$, while $d$, $a$ are 0. At the end, we only need a simple rotation to fix the orientation of our coordinate frame and a simple translation to reach the end effector.

Table 4.2 shows the DH parameters for all the joints of the left arm chain along with the necessary translations and rotations. Now, we can easily calculate the final transformation matrix:

$$T^{\text{End}}_{\text{Base}} = A^0_{\text{Base}} T^1_0 T^2_1 T^3_2 T^4_3 R_z(-\tfrac{\pi}{2}) A^{\text{End}}_4$$

### 4.1.3 Forward Kinematics for the Right Arm

The kinematic chain of the right arm is fully symmetric with the left arm chain relatively to the plane defined by the $x$-axis and the $z$-axis. So, the differences between the two chains are only in the distances along the $y$-axis and in the joints that rotate about

Table 4.3: DH parameters for the right arm chain of the NAO robot

| **Frame (Joint)** | **a** | $\boldsymbol{\alpha}$ | **d** | $\boldsymbol{\theta}$ |
|---|---|---|---|---|
| Base | $A(0, -\text{ShoulderOffsetY}, \text{ShoulderOffsetZ})$ | | | |
| RShoulderPitch | 0 | $-\frac{\pi}{2}$ | 0 | $\theta_1$ |
| RShoulderRoll | 0 | $\frac{\pi}{2}$ | 0 | $\theta_2 + \frac{\pi}{2}$ |
| RElbowYaw | $-\text{ElbowOffsetY}$ | $\frac{\pi}{2}$ | UpperArmLength | $\theta_3$ |
| RElbowRoll | 0 | $-\frac{\pi}{2}$ | 0 | $\theta_4$ |
| Rotation | $R_z(-\frac{\pi}{2})$ | | | |
| End effector | $A(\text{HandOffsetX+ LowerArmLength}, 0, 0)$ | | | |

the $y$-axis. Also, in this chain we must add one extra rotation matrix after the final translation, because the $z$-axis is inverted. All the DH parameters for this chain can be seen in Table 4.3 and the final transformation matrix is:

$$T^{\text{End}}_{\text{Base}} = A^0_{\text{Base}} T^1_0 T^2_1 T^3_2 T^4_3 R_z(-\tfrac{\pi}{2}) A^{\text{End}}_4$$

### 4.1.4 Forward Kinematics for the Left Leg

The kinematic chain for the left leg has six joints and it is the longest chain on the NAO robot. The DH parameters for these joints are interesting, because of the "weird" orientation of the HipYawPitch joint. Table 4.4 shows the DH parameters for the entire kinematic chain of the left leg and the final transformation matrix is:

$$T^{\text{End}}_{\text{Base}} = A^0_{\text{Base}} T^1_0 T^2_1 T^3_2 T^4_3 T^5_4 T^6_5 R_z(\pi) R_y(-\tfrac{\pi}{2}) A^{\text{End}}_6$$

### 4.1.5 Forward Kinematics for the Right Leg

Similarly to the arms, the kinematic chains for the legs are fully symmetric relatively to the plane defined by the $x$-axis and the $z$-axis. So, the differences between the two chains is only in the distances along the $y$-axis and in the joints that rotate about the $y$-axis. Table 4.5 shows all the DH parameters for the right leg chain and the final transformation matrix is:

$$T^{\text{End}}_{\text{Base}} = A^0_{\text{Base}} T^1_0 T^2_1 T^3_2 T^4_3 T^5_4 T^6_5 R_z(\pi) R_y(-\tfrac{\pi}{2}) A^{\text{End}}_6$$

Table 4.4: DH parameters for the left leg chain of the NAO robot

| **Frame (Joint)** | **a** | $\boldsymbol{\alpha}$ | **d** | $\boldsymbol{\theta}$ |
|---|---|---|---|---|
| Base | $A(0, \text{HipOffsetY}, -\text{HipOffsetZ})$ | | | |
| LHipYawPitch | 0 | $-\frac{3\pi}{4}$ | 0 | $\theta_1 - \frac{\pi}{2}$ |
| LHipRoll | 0 | $-\frac{\pi}{2}$ | 0 | $\theta_2 + \frac{\pi}{4}$ |
| LHipPitch | 0 | $\frac{\pi}{2}$ | 0 | $\theta_3$ |
| LKneePitch | $-\text{ThighLength}$ | 0 | 0 | $\theta_4$ |
| LAnklePitch | $-\text{TibiaLength}$ | 0 | 0 | $\theta_5$ |
| LAnkleRoll | 0 | $-\frac{\pi}{2}$ | 0 | $\theta_6$ |
| Rotation | $R_z(\pi)R_y(-\frac{\pi}{2})$ | | | |
| End effector | $A(0, 0, -\text{FootHeight})$ | | | |

Table 4.5: DH parameters for the right leg chain of the NAO robot

| **Frame (Joint)** | **a** | $\boldsymbol{\alpha}$ | **d** | $\boldsymbol{\theta}$ |
|---|---|---|---|---|
| Base | $A(0, -\text{HipOffsetY}, -\text{HipOffsetZ})$ | | | |
| RHipYawPitch | 0 | $-\frac{\pi}{4}$ | 0 | $\theta_1 - \frac{\pi}{2}$ |
| RHipRoll | 0 | $-\frac{\pi}{2}$ | 0 | $\theta_2 - \frac{\pi}{4}$ |
| RHipPitch | 0 | $\frac{\pi}{2}$ | 0 | $\theta_3$ |
| RKneePitch | $-\text{ThighLength}$ | 0 | 0 | $\theta_4$ |
| RAnklePitch | $-\text{TibiaLength}$ | 0 | 0 | $\theta_5$ |
| RAnkleRoll | 0 | $-\frac{\pi}{2}$ | 0 | $\theta_6$ |
| Rotation | $R_z(\pi)R_y(-\frac{\pi}{2})$ | | | |
| End effector | $A(0, 0, -\text{FootHeight})$ | | | |

### 4.1.6 Forward Kinematics for Combined Chains

The forward kinematics transformations presented above assume the torso frame as the base frame. In practice, a NAO user may be interested in finding the point of the torso relatively to one of the feet. Note that this is the forward kinematics problem for the reverse chain. Given that the forward kinematics transformation matrices are affine transformation matrices, we can obtain the solution for this reverse problem by simply inverting the corresponding transformation matrix. For example, inverting the transformation matrix for the left leg chain yields a transformation matrix for the point

of the torso in the frame of the left foot:

$$T_{\text{LFoot}}^{\text{Torso}} = \left(T_{\text{Torso}}^{\text{LFoot}}\right)^{-1}$$

This solution reversion property of the forward kinematics allows the combination of multiple chains to obtain the point of the end effector of one chain in the frame of the end effector of the other chain through their common point (torso).

For example, it is possible to find the point of the head relatively to the left foot. The kinematic chains for the head and for the left leg are relative to the torso frame. So, by inverting the transformation matrix of the left leg chain, we locate the torso relatively to the left foot frame. Then, we just multiply with the transformation matrix of the head chain and obtain a new transformation matrix that describes the point of the head relatively to the left foot frame.

$$T_{\text{LFoot}}^{\text{Head}} = \left(T_{\text{Torso}}^{\text{LFoot}}\right)^{-1} T_{\text{Torso}}^{\text{Head}}$$

This property is extremely useful, because we can describe any end effector relatively to the frame of any other end effector. For example, with this, we can find the exact height of the camera from the ground.

### 4.1.7 Calculation of the Center Of Mass

The Center of Mass (CoM) of a body in the three-dimensional space is a position, which corresponds to the weighted average location of all the mass in the body. From a physics point of view, the body (even if oddly-shaped) could be represented by a point mass located at the CoM. In humanoid robots, knowledge of the CoM is important to maintain balance. It is easy to see that the CoM changes, as the joint values change and the kinematic chains move in the three-dimensional space.

The NAO robot consists of a group of connected parts (joints and the corresponding links). Each part has its own known mass and its own (local) CoM at a known static position. For any given configuration of the robot, forward kinematics can be applied to locate each of these parts in the three-dimensional space of the torso frame and from there calculate the exact position of the robot CoM relatively to the torso frame.

Aldebaran Robotics provides the information needed for CoM calculations, in particular the mass of the whole robot and the masses of all parts of the robot. The mass of

each distinct part is referenced by the corresponding joint of that part and its own local CoM is given relatively to its own joint frame.

The CoM of the entire robot is calculated relatively to the torso frame and the calculation order is simple. We first construct smaller kinematic chains, one for each of the 21 joints. Each of these smaller kinematic chains terminates at the corresponding joint and the position of the local CoM is set to be the end effector. We compute the forward kinematics for each of these 21 chains. Then, we extract the translation block of each transformation matrix and we multiply it with the mass of the corresponding part/joint. In total, we have 21 chains plus the torso chain (a zero-length chain). Finally, we add all the individual weighted translation matrices and the result is divided by the total mass of the robot. The outcome of this calculation is the position of the CoM relatively to the torso frame.

## 4.2 Inverse Kinematics for the NAO Robot

Forward kinematics can find the point of an end effector, relatively to the start frame, given the values of the joints. Now, we turn our attention to the inverse problem: find a set of values for the joints that drive a given end effector to a desired point relatively to the torso frame. The inverse kinematics presented below solve the problem for the five kinematic chains that start at the robot torso.

A point of the end effector in the three-dimensional space consists of a position $(p_x, p_y, p_z)$ and an orientation $(a_x, a_y, a_z)$. As mentioned above, the outcome of forward kinematics is an affine transformation matrix, which includes a rotation block and a translation block. The rotation block $R$ takes the form $R_z R_y R_x$. Thus, we can construct the complete transformation matrix $T$ from the base frame to the point of the end effector mentioned above:

$$T = \begin{bmatrix} \cos a_y \cos a_z & -\cos a_x \sin a_z + \sin a_x \sin a_y \cos a_z & \sin a_x \sin a_z + \cos a_x \sin a_y \cos a_z & p_x \\ \cos a_y \sin a_z & \cos a_x \cos a_z + \sin a_x \sin a_y \sin a_z & -\sin a_x \cos a_z + \cos a_x \sin a_y \sin a_z & p_y \\ -\sin a_y & \sin a_x \cos a_y & \cos a_x \cos a_y & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Given $T$, our goal now is to find a set of joint values that leads to the same transformation through the kinematic chain. As mentioned before, the problem of inverse kinematics

cannot be solved without the solution of forward kinematics. This is true, because the equations we must solve to find the values of the joints are formed by writing down the forward kinematics transformation matrix symbolically with the $\theta_i$'s of the DH parameters appearing as symbols in the matrix. This symbolic matrix is set to be equal to $T$ to yield twelve non-linear equations with the values $\theta_i$ of the $n$ joints of the chain as unknowns. In fact, we will have a total of $2n$ unknowns, because all the $\theta_i$'s appear inside a sine or a cosine, therefore for each joint $i$ there are two dependent unknowns in the system, $\sin\theta_i$ and $\cos\theta_i$.

As we shall see below, we obtain some of the solutions using arccos and arcsin. The problem is that arcsin returns an angle in $\left[-\frac{\pi}{2}, +\frac{\pi}{2}\right]$ and arccos returns an angle in $[0, \pi]$, even though the possible values of a joint can be in the range $[-\pi, +\pi]$. Thus, if $\theta_i^*$ is a value returned by arcsin or arccos as a solution, there is one additional symmetric solution, $\pi - \theta_i^*$ for arcsin and $-\theta_i^*$ for arccos. Due to these symmetries, the equations lead to a small number of distinct candidate solutions, some of which may be infeasible or invalid because of the constrained range of each joint. To determine a valid and correct solution, we simply run each one of these solutions through the forward kinematics to verify that indeed the end effector of the chain reaches the desired position and orientation. Invalid solutions are discarded and only correct ones are kept.

In summary, the solution methodology we followed to solve the inverse kinematics problem includes the following steps:

1. Construct the (numeric) transformation matrix to the target point
2. Construct the (symbolic) transformation matrix through the chain
3. Form a non-linear system by equating the two matrices
4. Manipulate both sides to make the problem easier
5. Find values for some joints through geometry and trigonometry
6. Find values for the remaining joints from the non-linear system
7. Validate all candidate solutions through forward kinematics

This methodology is quite generic and can be used in solving the problem for any kinematic chain of up to six DOF. Some steps of this methodology appear in the literature, however to our knowledge no previous work has employed all these seven steps together.

### 4.2.1 Inverse Kinematics for the Head

The head chain consists of only two joints, therefore we can work either with the position $(p_x, p_y, p_z)$ or with the orientation $(a_x, a_y, a_z)$ of the target point to obtain a solution. In the latter case, we can achieve the desired target orientation simply by setting the HeadYaw and HeadPitch joints to $a_z$ and $a_y$ respectively. In the former case, we first construct the symbolic matrix from forward kinematics along the head chain:

$$T = \begin{bmatrix} -\cos\theta_1 \sin\widehat{\theta}_2 & -\sin\theta_1 & \cos\theta_1 \cos\widehat{\theta}_2 & l_2 \cos\theta_1 \cos\widehat{\theta}_2 - l_1 \cos\theta_1 \sin\widehat{\theta}_2 \\ -\sin\theta_1 \sin\widehat{\theta}_2 & \cos\theta_1 & \cos\widehat{\theta}_2 \sin\theta_1 & l_2 \cos\widehat{\theta}_2 \sin\theta_1 - l_1 \sin\theta_1 \sin\widehat{\theta}_2 \\ -\cos\widehat{\theta}_2 & 0 & -\sin\widehat{\theta}_2 & l_3 - l_2 \sin\widehat{\theta}_2 - l_1 \cos\widehat{\theta}_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where $\widehat{\theta}_2$ is the DH parameter $\theta$ for the second joint, $l_1$ = cameraX, $l_2$ = cameraZ, and $l_3$ = NeckOffsetZ. Since we only know the position $(p_x, p_y, p_z)$, we cannot reconstruct the rotation block of the matrix, so we focus on the translation block. From the symbolic matrix, we have:

$$T_{(3,4)} = l_3 - l_2 \sin\widehat{\theta}_2 - l_1 \cos\widehat{\theta}_2 = p_z$$

We know from trigonometry that:

$$a \sin\theta + b \cos\theta = \sqrt{a^2 + b^2} \sin(\theta + \psi)$$

$$\psi = \arctan\left(\frac{b}{a}\right) + \begin{cases} 0 & \text{if } a \geq 0 \\ \pi & \text{if } a < 0 \end{cases}$$

Given that $l_2 > 0$, we can now calculate $\widehat{\theta}_2$ as follows:

$$\widehat{\theta}_2 = \arcsin\left(\frac{-p_z + l_3}{\sqrt{{l_1}^2 + {l_2}^2}}\right) - \arctan\left(\frac{l_1}{l_2}\right)$$

Now the final joint value $\theta_2$ is $\theta_2 = \widehat{\theta}_2 + \frac{\pi}{2}$. From now on, we substitute $\theta_2 - \frac{\pi}{2}$ in $\widehat{\theta}_2$. Now, we can easily extract $\theta_1$ from $T_{(1,4)}$:

$$\theta_1 = \arccos\left(\frac{p_x}{l_2 \cos\left(\theta_2 - \frac{\pi}{2}\right) - l_1 \sin\left(\theta_2 - \frac{\pi}{2}\right)}\right)$$

So, the final inverse kinematics equations for the head chain are:

$$\theta_2 = \arcsin\left(\frac{-p_z + l_3}{\sqrt{{l_1}^2 + {l_2}^2}}\right) - \arctan\left(\frac{l_1}{l_2}\right) + \frac{\pi}{2}$$

$$\theta_2 = \pi - \arcsin\left(\frac{-p_z + l_3}{\sqrt{{l_1}^2 + {l_2}^2}}\right) - \arctan\left(\frac{l_1}{l_2}\right) + \frac{\pi}{2}$$

$$\theta_1 = \pm \arccos\left(\frac{p_x}{l_2 \cos\left(\theta_2 - \frac{\pi}{2}\right) - l_1 \sin\left(\theta_2 - \frac{\pi}{2}\right)}\right)$$

provided a target position $(p_x, p_y, p_z)$ or

$$\theta_1 = a_z$$

$$\theta_2 = a_y$$

provided a target orientation $(a_x, a_y, a_z)$.

### 4.2.2 Inverse Kinematics for the Left Arm

The left arm chain is far more complicated than the head chain. First, we construct the symbolic transformation matrix for the left arm chain, then we remove from the chain the Base and End transformation together with the End rotation.

$$T = A^0_{\text{Base}} T^1_0 T^2_1 T^3_2 T^4_3 R_z(\tfrac{\pi}{2}) A^{\text{End}}_4$$

$$T' = \left(A^0_{\text{Base}}\right)^{-1} T \left(A^{\text{End}}_4\right)^{-1} \left(R_z(\tfrac{\pi}{2})\right)^{-1}$$

Then we invert $T'$:

$$T'' = (T')^{-1}$$

$$T'' = \begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ r_{21} & r_{22} & r_{23} & r_{24} \\ r_{31} & r_{32} & r_{33} & r_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$r_{11} = \cos\theta_4 \sin\theta_1 \sin\theta_3 + \cos\theta_1(\cos\widehat{\theta}_2 \cos\theta_3 \cos\theta_4 - \sin\widehat{\theta}_2 \sin\theta_4)$$

$$r_{12} = \cos\theta_3 \cos\theta_4 \sin\widehat{\theta}_2 + \cos\widehat{\theta}_2 \sin\theta_4$$

$$r_{13} = -\cos\widehat{\theta}_2 \cos\theta_3 \cos\theta_4 \sin\theta_1 + \cos\theta_1 \cos\theta_4 \sin\theta_3 + \sin\theta_1 \sin\widehat{\theta}_2 \sin\theta_4$$

$$r_{14} = -l_1 \cos\theta_3 \cos\theta_4 + l_2 \sin\theta_4$$

$$r_{21} = -\sin\theta_1 \sin\theta_3 \sin\theta_4 - \cos\theta_1(\cos\theta_4 \sin\widehat{\theta}_2 + \cos\widehat{\theta}_2 \cos\theta_3 \sin\theta_4)$$

$$r_{22} = \cos\widehat{\theta}_2 \cos\theta_4 - \cos\theta_3 \sin\widehat{\theta}_2 \sin\theta_4$$

$$r_{23} = \cos\theta_4 \sin\theta_1 \sin\widehat{\theta}_2 + (\cos\widehat{\theta}_2 \cos\theta_3 \sin\theta_1 - \cos\theta_1 \sin\theta_3) \sin\theta_4$$

$$
\begin{aligned}
r_{24} &= l_2 \cos\theta_4 + l_1 \cos\theta_3 \sin\theta_4 \\
r_{31} &= \cos\theta_3 \sin\theta_1 - \cos\theta_1 \cos\widehat{\theta}_2 \sin\theta_3 \\
r_{32} &= -\sin\widehat{\theta}_2 \sin\theta_3 \\
r_{33} &= \cos\theta_1 \cos\theta_3 + \cos\widehat{\theta}_2 \sin\theta_1 \sin\theta_3 \\
r_{34} &= l_1 \sin\theta_3
\end{aligned}
$$

where $\widehat{\theta}_2$ is the DH parameter $\theta$ for the second (LShoulderRoll) joint, $l_1$ = ElbowOffsetY, $l_2$ = UpperArmLength.

We can extract $theta_3$ from the resulted transformation matrix:

$$
\theta_3 = \begin{cases} arcsin\left(\frac{T''_{(3,4)}}{l_1}\right) \\ \pi - arcsin\left(\frac{T''_{(3,4)}}{l_1}\right) \end{cases}
$$

The next angle we can find is $\theta_4$, so we focus on equations from the symbolic matrix, where we only have $\theta_4$ and/or $\theta_3$ (now $\theta_3$ is a known variable). Using $T''_{(1,4)}$, we have:

$$
\begin{aligned}
T''_{(1,4)} &= -l_1 \cos\theta_3 \cos\theta_4 + l_2 \sin\theta_4 \qquad \Leftrightarrow \\
\sin\theta_4 &= \frac{T''_{(1,4)} + l_2 \cos\theta_3 \cos\theta_4}{l_3}
\end{aligned}
$$

Now, we focus on $T_{24}$:

$$
\begin{aligned}
T''_{24} &= l_2 \cos\theta_4 + l_1 \cos\theta_3 \sin\theta_4 \qquad \Leftrightarrow \\
T''_{24} &= l_2 \cos\theta_4 + l_1 \cos\theta_3 \frac{T''_{(1,4)} + l_2 \cos\theta_3 \cos\theta_4}{l_3} \qquad \Leftrightarrow \\
\cos\theta_4 \left(l_2^2 + l_1^2 \cos^2\theta_3\right) &= l_3 T''_{(2,4)} - l_2 T''_{(1,4)} \cos\theta_3 \qquad \Leftrightarrow \\
\theta_4 &= \pm arccos\left(\frac{l_3 T''_{(2,4)} - l_2 T''_{(1,4)} \cos\theta_3}{l_2^2 + l_1^2 \cos^2\theta_3}\right)
\end{aligned}
$$

Next we manipulate again our chain. Now bot DH-transformations for $\theta_3$ and $\theta_4$ can be removed from the chain so:

$$
T''' = T' \left(T_2^3\right)^{-1} \left(T_3^4\right)^{-1}
$$

Now the final two joint values can be easily extract:

$$
\widehat{\theta}_2 = arctan\left(\frac{T'''_{(1,2)}}{T'''_{(2,2)}}\right)
$$

$$\theta_2 = \widehat{\theta_2} - \frac{\pi}{2}$$

$$\theta_1 = arctan\left(\frac{T'''_{(3,1)}}{T'''_{(3,3)}}\right)$$

At this point, we have multiple solution sets for all the joints of the left arm, but some of them may be invalid. We find the correct set(s) through a forward kinematics validation step. In summary, the final inverse kinematics equations for the left arm chain are:

$$\begin{aligned}
T' &= \left(A^0_{\text{Base}}\right)^{-1} T \left(A^{\text{End}}_4\right)^{-1} \\
T'' &= (T')^{-1} \left(R_z(\tfrac{\pi}{2})\right)^{-1} \\
\theta_3 &= \begin{cases} arcsin\left(\frac{T''_{(3,4)}}{l_1}\right) \\ \pi - arcsin\left(\frac{T''_{(3,4)}}{l_1}\right) \end{cases} \\
\theta_4 &= \pm arccos\left(\frac{l_3 T''_{(2,4)} - l_2 T''_{(1,4)} \cos\theta_3}{l_2^2 + l_1^2 \cos^2\theta_3}\right) \\
T''' &= T' \left(T^3_2\right)^{-1} \left(T^4_3\right)^{-1} \\
\theta_2 &= arctan\left(\frac{T'''_{(2,1)}}{T'''_{(2,2)}}\right) - \frac{\pi}{2} \\
\theta_1 &= arctan\left(\frac{T'''_{(1,3)}}{T'''_{(3,3)}}\right)
\end{aligned}$$

### 4.2.3 Inverse Kinematics for the Right Arm

The right and left arms are symmetric, thus the solution for the right arm differs only in the distances along the $y$-axis and at the "roll" joints (RShoulderRoll, RElbowRoll). The only basic difference with the distances along the $y$-axis are with the ElbowOffsetY that now becomes negative, thus $l_1 = -\text{ElbowOffsetY}$.

It is clear that the chain for the right arm is the same as the chain for the left arm. As we can see, the changes don't have any impact in the solution given for the left arm, that is, no denominator becomes zero after these changes. So, according to the previous section, the final inverse kinematics equations for the right arm chain are:

$$\begin{aligned}
T' &= \left(A^0_{\text{Base}}\right)^{-1} T \left(A^{\text{End}}_4\right)^{-1} \left(R_z(\tfrac{\pi}{2})\right)^{-1} \\
T'' &= (T')^{-1}
\end{aligned}$$

$$\theta_3 = \begin{cases} arcsin\left(\frac{T''_{(3,4)}}{l_1}\right) \\ \pi - arcsin\left(\frac{T''_{(3,4)}}{l_1}\right) \end{cases}$$

$$\theta_4 = \pm arccos\left(\frac{l_3 T''_{(2,4)} - l_2 T''_{(1,4)} \cos\theta_3}{l_2^2 + l_1^2 \cos^2\theta_3}\right)$$

$$T''' = T' \left(T_2^3\right)^{-1} \left(T_3^4\right)^{-1}$$

$$\theta_2 = arctan\left(\frac{T'''_{(2,1)}}{T'''_{(2,2)}}\right) - \frac{\pi}{2}$$

$$\theta_1 = arctan\left(\frac{T'''_{(1,3)}}{T'''_{(3,3)}}\right)$$

### 4.2.4 Inverse Kinematics for the Left Leg

The kinematic chain of the left leg has six joints, so it is much more difficult to find a solution. Since the first three joints of the chain (LHipYawPitch, LHipRoll, LHipPitch) have intersecting axes, the problem is possibly solvable. The symbolic matrix for this chain is too complicated, therefore in order to simplify it, we remove the known translations from the kinematic chain:

$$T = A^0_{\text{Base}} T_0^1 T_1^2 T_2^3 T_3^4 T_4^5 T_5^6 R_z(\pi) R_y(-\tfrac{\pi}{2}) A_6^{\text{End}}$$

$$\widehat{T} = \left(A^0_{\text{Base}}\right)^{-1} T \left(A_6^{\text{End}}\right)^{-1}$$

Now, we have a chain from the base frame of the first joint to the (rotated) frame of the last joint. The first joint, LHipYawPitch, is by construction rotated by $-\frac{3\pi}{4}$ about the $x$-axis with respect to the torso frame. We rotate the origin of the chain by $\frac{\pi}{4}$ about the $x$-axis to make the first joint (LHipYawPitch) a yaw joint (aligned with the $z$-axis):

$$\widetilde{T} = R_x(\tfrac{\pi}{4}) \widehat{T}$$

Now, the end effector is the LAnkleRoll joint and the base is the rotated LHipYawPitch joint. The first four joints (LHipYawPitch, LHipRoll, LHipPitch, LKneePitch) affect the position and orientation of the end effector and the other two joints (LAnklePitch, LAnkleRoll) affect only its orientation. It would be convenient, if only three joints were affecting the position of the end effector, since we operate in the three-dimensional space.

Thus, we invert the transformation matrix to form the reverse chain. Now only the LAnkleRoll, LAnklePitch, and LKneePitch joints affect the position:

$$T' = \left(\widetilde{T}\right)^{-1}$$

The resulting symbolic matrix is still quite complex, but for now we only need the translation block, which is relatively simple:

$$\begin{aligned} r_{14} &= l_2 \sin\theta_5 - l_1 \sin\left(\theta_4 + \theta_5\right) \\ r_{24} &= \left(l_2 \cos\theta_5 + l_1 \cos\left(\theta_4 + \theta_5\right)\right) \sin\theta_6 \\ r_{34} &= \left(l_2 \cos\theta_5 + l_1 \cos\left(\theta_4 + \theta_5\right)\right) \cos\theta_6 \end{aligned}$$

where $l_1$ = ThighLength and $l_2$ = TibiaLength. We can now find $\theta_4$ the same way we found $\theta_4$ for the arms. We focus on the triangle formed by the leg with ThighLength, TibiaLength, and the distance from the base to the end effector in the reverse chain as sides:

$$d = \sqrt{\left(s_x - p'_x\right)^2 + \left(s_y - p'_y\right)^2 + \left(s_z - p'_z\right)^2}$$

where $(s_x, s_y, s_z) = (0, 0, 0)$ is the new origin and $(p'_x, p'_y, p'_z) = (T'_{(1,4)}, T'_{(2,4)}, T'_{(3,4)})$ is the position of the new target point. Now, we can use the law of cosines to find the interior angle $\theta'_4$ between the thigh and tibia sides of the triangle:

$$\theta'_4 = \arccos\left(\frac{{l_1}^2 + {l_2}^2 - d^2}{2 l_1 l_2}\right)$$

Since $\theta'_4$ represents an interior angle, whereas the LKneePitch joint is stretched in the zero position, the resulting angle $\theta''_4$ in this range is computed by:

$$\theta''_4 = \pi - \theta'_4$$

Since the range of the LKneePitch joint includes both positive and negative angles, we finally extract $\theta_4$ as:

$$\theta_4 = \pm\theta''_4$$

Next, we extract the $\theta_6$ angle from the translation block using $r_{24}$ and $r_{34}$:

$$\frac{r_{24}}{r_{34}} = \frac{p'_y}{p'_z} \qquad \Leftrightarrow$$

$$\frac{(l_2 \cos\theta_5 + l_1 \cos(\theta_4 + \theta_5)) \sin\theta_6}{(l_2 \cos\theta_5 + l_1 \cos(\theta_4 + \theta_6)) \cos\theta_6} = \frac{p'_y}{p'_z} \qquad \Leftrightarrow$$

$$\theta_6 = \arctan\left(\frac{p'_y}{p'_z}\right) \quad \text{if } (l_2 \cos\theta_5 + l_1 \cos(\theta_4 + \theta_5)) \neq 0$$

Since we don't know $\theta_5$, we cannot determine in advance when this solution is valid. Figure 4.2 shows the locus of the equation $l_2 \cos\theta_5 + l_1 \cos(\theta_4 + \theta_5) = 0$ in the space of the KneePitch ($\theta_4$) and AnklePitch ($\theta_5$) joints; in these configurations, which represent only a fraction of extreme values in the joint space, no unique solution can be extracted for $\theta_6$. Essentially, in these configurations, the position of the end effector of the reverse chain (the hip joints) falls on the rotation axis of first joint (LAnkleRoll) in the reverse chain and therefore the LAnkleRoll joint ($\theta_6$) has no effect on its position; an infinity of solutions exists, given that for any choice of $\theta_6$, there will be a corresponding set of choices for the hip joints that achieves the target orientation. In practice, even if one of these configurations shows up during the solution, a division by zero never occurs, because the `atan2` function in our implementation simply returns a result of 0. We adopt the solution $\theta_6 = 0$, and we relay the problem of the target orientation to the hip joints; invalid solutions, if any, will be rejected by the validation step.

To move on, we go back to $\widetilde{T}$ and we remove the two rotations at the end of the chain along with the transformation $T_5^6$ which is now known, because $\theta_6$ is known:

$$\widetilde{T}' = \widetilde{T}\left(T_5^6 R_z(\pi)\, R_y(-\tfrac{\pi}{2})\right)^{-1}$$

Like before, we form the reverse chain:

$$T'' = \left(\widetilde{T}'\right)^{-1}$$

and we extract the new target position $(p''_x, p''_y, p''_z) = (T''_{(1,4)}, T''_{(2,4)}, T''_{(3,4)})$. The translation block of the new symbolic transformation matrix is:

$$\begin{aligned} r_{14} &= l_2 \cos\theta_5 + l_1 (\cos\theta_5 \cos\theta_4 - \sin\theta_5 \sin\theta_4) \\ r_{24} &= -l_2 \sin\theta_5 - l_1 (\sin\theta_5 \cos\theta_4 + \cos\theta_5 \sin\theta_4) \\ r_{34} &= 0 \end{aligned}$$

In these equations, $\theta_5$ is the only unknown. From $r_{14}$, we obtain an expression for $\cos\theta_5$:

$$r_{14} = p''_x \qquad \Leftrightarrow$$

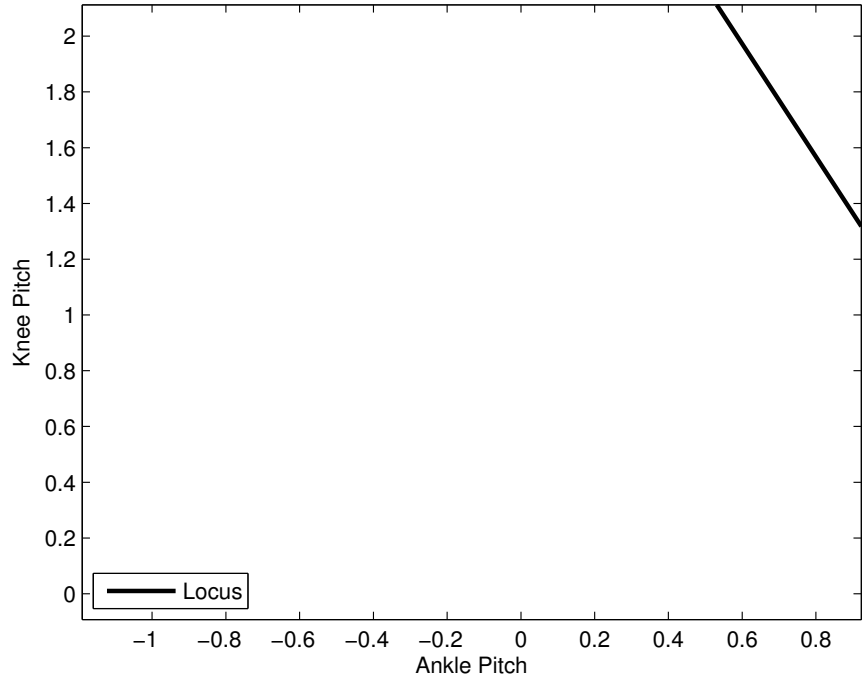



Figure 4.2: Locus of leg configurations corresponding to non-unique solutions to $\theta_6$

$$
\begin{aligned}
(l_2 + l_1 \cos\theta_4)\cos\theta_5 &= p_x'' + l_1 \sin\theta_5 \sin\theta_4 && \Leftrightarrow \\
\cos\theta_5 &= \frac{p_x'' + l_1 \sin\theta_5 \sin\theta_4}{l_2 + l_1 \cos\theta_4} && \text{if } l_2 + l_1 \cos\theta_4 \neq 0
\end{aligned}
$$

Given the lengths of the links of the robot, the denominator $l_2 + l_1 \cos\theta_4$ becomes zero, only if $\cos\theta_4 = -1.029$, which is impossible, since $|\cos\theta_4| \leq 1$. We continue with $r_{24}$:

$$
\begin{aligned}
&\sin\theta_5 (-l_2 - l_1 \cos\theta_4) - l_1 \cos\theta_5 \sin\theta_4 = p_y'' && \Leftrightarrow \\
&\sin\theta_5 (-l_2 - l_1 \cos\theta_4) - l_1 \frac{p_x'' + l_1 \sin\theta_5 \sin\theta_4}{l_2 + l_1 \cos\theta_4} \sin\theta_4 \ = p_y'' && \Leftrightarrow \\
&-\sin\theta_5 (l_2 + l_1 \cos\theta_4) - \frac{l_1 p_x'' \sin\theta_4}{l_2 + l_1 \cos\theta_4} - \frac{{l_1}^2 \sin\theta_5 \sin^2\theta_4}{l_2 + l_1 \cos\theta_4} = p_y'' && \Leftrightarrow \\
&-\sin\theta_5 (l_2 + l_1 \cos\theta_4)^2 - {l_1}^2 \sin\theta_5 \sin^2\theta_4 = p_y'' (l_2 + l_1 \cos\theta_4) + l_1 p_x'' \sin\theta_4 && \Leftrightarrow \\
&\theta_5 = \arcsin\left( -\frac{p_y'' (l_2 + l_1 \cos\theta_4) + l_1 p_x'' \sin\theta_4}{{l_1}^2 \sin^2\theta_4 + (l_2 + l_1 \cos\theta_4)^2} \right)
\end{aligned}
$$

The division is always feasible, because ${l_1}^2 \sin^2\theta_4 + (l_2 + l_1\cos\theta_4)^2$ is obviously greater than zero for any value of $\theta_4$. Now, we can go back to $\widetilde{T}'$ and remove the two transformations $T_3^4$ and $T_4^5$, since $\theta_4$ and $\theta_5$ are known:

$$T''' = \widetilde{T}' \left(T_3^4 T_4^5\right)^{-1}$$

The translation block in the transformation $T'''$ must be zero, because the only joints left are the three hip joints, which only affect the orientation. The rotation block of the transformation is:

$$\begin{aligned}
r_{11} &= \cos\widehat{\theta}_1 \cos\widehat{\theta}_2 \cos\theta_4 - \sin\widehat{\theta}_1 \sin\theta_3 \\
r_{12} &= -\cos\theta_3 \sin\widehat{\theta}_1 - \cos\widehat{\theta}_1 \cos\widehat{\theta}_2 \sin\theta_3 \\
r_{13} &= \cos\widehat{\theta}_1 \sin\widehat{\theta}_2 \\
r_{21} &= -\cos\theta_3 \sin\widehat{\theta}_2 \\
r_{22} &= \sin\widehat{\theta}_2 \sin\theta_3 \\
r_{23} &= \cos\widehat{\theta}_2 \\
r_{31} &= -\cos\widehat{\theta}_2 \cos\theta_3 \sin\widehat{\theta}_1 - \cos\widehat{\theta}_1 \sin\theta_3 \\
r_{32} &= -\cos\widehat{\theta}_1 \cos\theta_3 + \cos\widehat{\theta}_2 \sin\widehat{\theta}_1 \sin\theta_3 \\
r_{33} &= -\sin\widehat{\theta}_1 \sin\widehat{\theta}_2
\end{aligned}$$

where $\widehat{\theta}_1$ is the DH parameter $\theta$ for the first (LHipYawPitch) joint and $\widehat{\theta}_2$ is the DH parameter $\theta$ for the second (LHipRoll) joint. Now, we can extract the remaining three angles as follows:

$$\begin{aligned}
\widehat{\theta}_2 &= \arccos T'''_{(2,3)} \\
\theta_2 &= \widehat{\theta}_2 - \frac{\pi}{4} \\
\theta_3 &= \arcsin\left(\frac{T'''_{(2,2)}}{\sin\left(\theta_2 + \frac{\pi}{4}\right)}\right) \\
\widehat{\theta}_1 &= \arccos\left(\frac{T'''_{(1,3)}}{\sin\left(\theta_2 + \frac{\pi}{4}\right)}\right) \\
\theta_1 &= \widehat{\theta}_1 + \frac{\pi}{2}
\end{aligned}$$

The equations above are valid, because by the robot construction the LHipRoll joint ($\theta_2$) cannot reach $-\frac{\pi}{4}$ or $\frac{3\pi}{4}$ and therefore the denominator $\sin\left(\theta_2+\frac{\pi}{4}\right)$ never becomes zero.

In summary, the equations of inverse kinematics for the left leg are:

$$T' = \left(R_x(\tfrac{\pi}{4})\left(\left(A^0_{\text{Base}}\right)^{-1} T \left(A^{\text{End}}_6\right)^{-1}\right)\right)^{-1}$$

$$\theta_4 = \pm\left(\pi - \arccos\left(\frac{{l_1}^2 + {l_2}^2 - \sqrt{\left(0-T'_{(1,4)}\right)^2+\left(0-T'_{(2,4)}\right)^2+\left(0-T'_{(3,4)}\right)^2}^2}{2l_1l_2}\right)\right)$$

$$\theta_6 = \arctan\left(\frac{T'_{(2,4)}}{T'_{(3,4)}}\right) \qquad \text{if } (l_2\cos\theta_5 + l_1\cos(\theta_4+\theta_5)) \neq 0$$

$$\theta_6 = 0 \qquad \text{if } (l_2\cos\theta_5 + l_1\cos(\theta_4+\theta_5)) = 0$$

$$T'' = \left((T')^{-1}\left(T^6_5 R_z(\pi) R_y(-\tfrac{\pi}{2})\right)^{-1}\right)^{-1}$$

$$\theta_5 = \arcsin\left(-\frac{T''_{(2,4)}(l_2+l_1\cos\theta_4) + l_1 T''_{(1,4)}\sin\theta_4}{{l_1}^2\sin^2\theta_4 + (l_2+l_1\cos\theta_4)}\right)$$

$$\theta_5 = \pi - \arcsin\left(-\frac{T''_{(2,4)}(l_2+l_1\cos\theta_4) + l_1 T''_{(1,4)}\sin\theta_4}{{l_1}^2\sin^2\theta_4 + (l_2+l_1\cos\theta_4)}\right)$$

$$T''' = (T'')^{-1}\left(T^4_3 T^5_4\right)^{-1}$$

$$\theta_2 = \pm\arccos\left(T'''_{(2,3)}\right) - \frac{\pi}{4}$$

$$\theta_3 = \arcsin\left(\frac{T'''_{(2,2)}}{\sin\left(\theta_2+\frac{\pi}{4}\right)}\right)$$

$$\theta_3 = \pi - \arcsin\left(\frac{T'''_{(2,2)}}{\sin\left(\theta_2+\frac{\pi}{4}\right)}\right)$$

$$\theta_1 = \pm\arccos\left(\frac{T'''_{(1,3)}}{\sin\left(\theta_2+\frac{\pi}{4}\right)}\right) + \frac{\pi}{2}$$

### 4.2.5 Inverse Kinematics for the Right Leg

The chains of the two legs are symmetric, so the solution for the right leg will be quite similar to the solution for the left leg. The only difference is in the rotation matrix for

the RHipYawPitch joint, which must be rotated by $-\frac{\pi}{4}$.

$$T = A^0_{\text{Base}} T^1_0 T^2_1 T^3_2 T^4_3 T^5_4 T^6_5 R_z(\pi) R_y(-\tfrac{\pi}{2}) A^{\text{End}}_6$$

$$\widehat{T} = \left(A^0_{\text{Base}}\right)^{-1} T \left(A^{\text{End}}_6\right)^{-1}$$

$$\widetilde{T} = R_x(-\tfrac{\pi}{4})\widehat{T}$$

$$T' = \left(\widetilde{T}\right)^{-1}$$

Besides this change, all the steps followed to reach a solution for the left leg apply without change to the right leg as well.

Therefore, the equations of inverse kinematics for the right leg are:

$$T' = \left(R_x(-\tfrac{\pi}{4}) \left(\left(A^0_{\text{Base}}\right)^{-1} T \left(A^{\text{End}}_6\right)^{-1}\right)\right)^{-1}$$

$$\theta_4 = \pm \left( \pi - \arccos \left( \frac{{l_1}^2 + {l_2}^2 - \sqrt{\left(0 - T'_{(1,4)}\right)^2 + \left(0 - T'_{(2,4)}\right)^2 + \left(0 - T'_{(3,4)}\right)^2}^2}{2 l_1 l_2} \right) \right)$$

$$\theta_6 = \arctan\left(\frac{T'_{(2,4)}}{T'_{(3,4)}}\right) \qquad \text{if } (l_2 \cos\theta_5 + l_1 \cos(\theta_4 + \theta_5)) \neq 0$$

$$\theta_6 = 0 \qquad \text{if } (l_2 \cos\theta_5 + l_1 \cos(\theta_4 + \theta_5)) = 0$$

$$T'' = \left((T')^{-1} \left(T^6_5 R_z(\pi) R_y(-\tfrac{\pi}{2})\right)^{-1}\right)^{-1}$$

$$\theta_5 = \arcsin\left(-\frac{T''_{(2,4)}(l_2 + l_1\cos\theta_4) + l_1 T''_{(1,4)} \sin\theta_4}{{l_1}^2 \sin^2\theta_4 + (l_2 + l_1\cos\theta_4)}\right)$$

$$\theta_5 = \pi - \arcsin\left(-\frac{T''_{(2,4)}(l_2 + l_1\cos\theta_4) + l_1 T''_{(1,4)} \sin\theta_4}{{l_1}^2 \sin^2\theta_4 + (l_2 + l_1\cos\theta_4)}\right)$$

$$T''' = (T'')^{-1} \left(T^4_3 T^5_4\right)^{-1}$$

$$\theta_2 = \pm \arccos\left(T'''_{(2,3)}\right) + \frac{\pi}{4}$$

$$\theta_3 = \arcsin\left(\frac{T'''_{(2,2)}}{\sin\left(\theta_2 - \frac{\pi}{4}\right)}\right)$$

$$\theta_3 = \pi - \arcsin\left(\frac{T'''_{(2,2)}}{\sin\left(\theta_2 - \frac{\pi}{4}\right)}\right)$$

$$\theta_1 = \pm \arccos\left(\frac{T'''_{(1c,3)}}{\sin\left(\theta_2 - \frac{\pi}{4}\right)}\right) + \frac{\pi}{2}$$

## 4.3 Implementation

Having completed all forward and inverse kinematics in analytical form, our next goal is to integrate them in the code of our RoboCup team for real-time, on-board execution on the NAO robot. The software architecture and the entire code of the team is written in `C++`. Given that `C++` offers no library for optimized real-time matrix operations, we relied on KMat, a minimalistic framework for such operations. Using this framework, it was fairly easy to implement our own kinematics library in `C++`, which includes functions that implement all equations of forward and inverse kinematics for the NAO robot.

### 4.3.1 KMat: Kouretes Math Library

KMat is a library developed by Emmanouil Orfanoudakis [**?**] that supports a selected subset of algebraic matrix operations. The focus of the library is mainly on real number operations and the primary goal of KMat is low memory footprint and calculation efficiency. Existing linear algebra libraries typically perform run-time validation for the compatibility of the operands and are optimized for large matrices. On the other hand, KMat is optimized for small matrices (typically $3 \times 3$ or $4 \times 4$) and supports only a selected subset of operations (addition, subtraction, multiplication, scalar addition, scalar multiplication, transposition, inversion). KMat is optimized for and supports two type of matrices: dense matrices and affine transformation matrices.

In our own work, we mainly used the functions offered for affine transformation matrices. In addition, we expanded the library with functionality that is extremely useful in the context of kinematics. These functions include matrix initialization given a set of DH parameters, construction of the transformation matrix for a given target point, etc.

### 4.3.2 Nao Kinematics `C++` Library

We created two libraries, `ForwardKinematics` and `InverseKinematics`, for NAO kinematics in `C++`. `ForwardKinematics` includes functions for calculating the point (position and orientation) of an end effector of a kinematic chain, given the joint values for this

chain. The input to each of these functions is a list of values for the joints in the order they appear in corresponding kinematic chain. The output is a list of six numbers, the three Cartesian coordinates of the position and the three Euler angles for the orientation. The combination of independent chains with a common base frame is also possible, so that one can find, for example, the position and orientation of the top camera relatively to one of the legs. This library also includes a function for calculating the center of mass of the robot given a set of values for all joints.

The `InverseKinematics` library includes five functions, each of which solves the problem for one of the five kinematic chains. All functions take as input the position and the orientation of the desired target point and the output is a list of solutions, where each solution includes values for all the joints of the chain. Although multiple solutions are rare, all solutions found are returned so that the user can decide which one to use.

A practical issue in inverse kinematics may have to be resolved manually, if it comes up. The two legs of the NAO robot have one common joint (HipYawPitch), therefore, if a user gives two target points, one for each leg, inverse kinematics for the left leg may return a different value for this joint than the inverse kinematics for the right leg. The user must decide which value to use to set the HipYawPitch joint. One choice is to give priority to the value returned by the leg that currently acts as support leg (the one on which the robot stands at the moment). Another choice is to set the joint to the mean of the two values. None of these choices is perfect, however it is likely that in a carefully-designed trajectory the resulting values from inverse kinematics for this joint will be close enough and the end result will be close to the desired one, independently of the user's choice.

# Chapter 5

# NAO Kinematics: The Results

There are several ways to validate solutions to forward and inverse kinematics and verify that the method used works properly. The verification of forward kinematics is trivial; we just move the end effector of a kinematic chain to a point and we check if forward kinematics returns the correct position and orientation. On the other hand, verification of inverse kinematics is more difficult, because we must give a valid (accessible) target point as input, otherwise inverse kinematics cannot find a solution.

A practical problem is there is no clear way to know the exact work space for the end effectors of NAO. Therefore, we came up with another way of verifying our solution. We manually move an end effector to a random point, we read off the position and the orientation of this point from the forward kinematics, and then we assign this point as the target point for inverse kinematics. Finally, we check if inverse kinematics finds a solution and we know right away that this solution is correct, because the solution is check through forward kinematics against the given target point before returned.

We created two demonstrations to show the effectiveness and the efficiency of our kinematics in the real world. The first demonstration is a pointing-to-the-ball task, whereby a standing robot tracks a ball in the field and uses forward kinematics of the legs and inverse kinematics of the arms to point to the exact location of the ball with its extended arm(s). The second demonstration is more complicated and consists of a simple balancing method, whereby the robot calculates its current center of mass through forward kinematics and drives one of its legs to the projection of the center of mass on the floor using inverse kinematics to maintain balance.

Table 5.1: On-board execution times of the NAO kinematics library

| Kinematics Function | Time (us) |
|---|---|
| Forward Kinematics for Head | 54.28 |
| Forward Kinematics for Left Arm | 66.72 |
| Forward Kinematics for Right Arm | 69.54 |
| Forward Kinematics for Left Leg | 80.88 |
| Forward Kinematics for Right Leg | 80.78 |
| Inverse Kinematics for Head | 70.79 |
| Inverse Kinematics for Left Arm | 170.55 |
| Inverse Kinematics for Right Arm | 200.00 |
| Inverse Kinematics for Left Leg | 185.29 |
| Inverse Kinematics for Right Leg | 184.85 |
| Calculation of the Center of Mass | 394.55 |

## 5.1 Real-Time Performance

One of the goals of this work was to implement a software library for real-time kinematics computations on the robot. We measured the performance of our implementation for each of the functions we offer. Table 5.1 shows average execution times for each function in microseconds (us). Averages were taken over 6000 measurements of execution time on the robot CPU. Unfortunately, we cannot compare these times with Aldebaran's solution execution time, because we no access inside the proprietary NaoQi middleware.

## 5.2 Locus of Problematic Inverse Kinematics

As mentioned in Section 4.2.4, there are a few target points for the legs which lead to an infinity of solutions for the AnkleRoll joints, while the KneePitch and AnklePitch joints take specific values. Even though this is not a problem, to verify that in practice the robot never reaches any of these configurations, we let the robot perform the entire range of motions available to it during operation in a RoboCup field (walk, kicks, stand-up, etc.). We recorded all configurations encountered in the KneePitch and AnklePitch subspace

of both legs and we plotted these configurations alongside the locus of the problematic configurations. The results are shown in Figure 5.1. It is clear that no motion brought the robot to these configurations. In practice, it is rather unlike that anyone will consistently give target points that drive the joints in that area. This is true, because, when executing a trajectory, inverse kinematics are called on a sequence of target points (at a frequency of 50 to 100 Hz), so even if one of them falls right into the problematic fraction of the three-dimensional space, it is highly unlike that the next one will do the same. At such a high frequency, the problematic solution will go unnoticed and the motion will proceed smoothly.

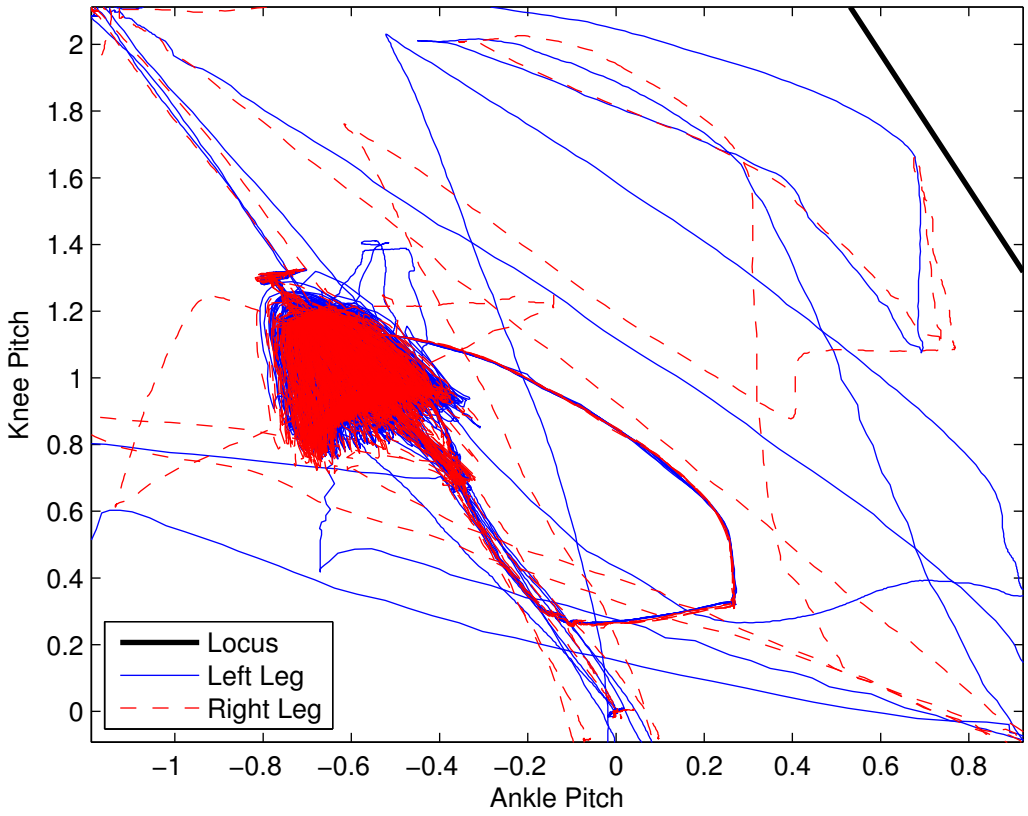



Figure 5.1: Trajectories of motion in a subspace of the leg joints

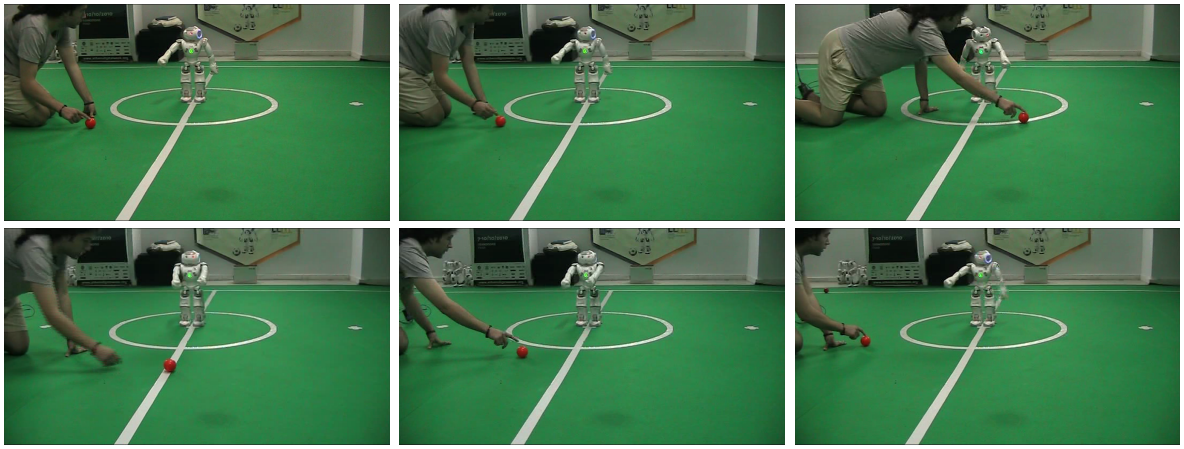


Figure 5.2: Pointing to the ball with the NAO using forward and inverse kinematics

## 5.3 Demonstration I: Pointing to the Ball

In the first demonstration, our goal is to make the NAO point to the ball with its stretched arms. Apart from the kinematics, to realize this task we employed the KVision module [**?**] for ball recognition of the Kouretes team, along with the module that updates the local world state belief of the robot. Initially, NAO searches for the ball by scanning the field with its head. When it finds the ball, it points directly to the ball with the left, the right, or both arms, depending on where the ball is located (left, right, or front). Thanks to the filtering of the world state module, if NAO loses momentarily the ball from its visual field, it can continue pointing to the location indicated by the ball model.

The ball observation can be described as a two-dimensional point without orientation. The location of the ball is given in polar coordinates, that is distance and bearing relatively to the projection of the robot torso on the floor. We must now transform this point from the two dimensions to the three dimensions and give it an orientation. To do so, we transform the polar coordinates to Cartesian coordinates $(p_x, p_y)$ and we add the height of the torso as the third coordinate $p_z$ to form the ball position $(p_x, p_y, p_z)$ in the three-dimensional space. We also know that $a_x$ equals to zero, because we are only rotating about the $y$-axis (up/down) and the $z$-axis (right/left). To find the two other orientations, we focus on the straight line that connects the position of the ball and the position of the ShoulderPitch joint relatively to the torso frame. The orientation $a_y$ is the angle between this line and the $y$-axis and the orientation $a_z$ is the angle between


Figure 5.3: Pointing to the ball at SPL Open Challenge Competition of RoboCup 2012

this line and the $z$-axis. Additionally, the position of the target point lies along this line at a distance $d$ from the ShoulderPitch joint, where $d$ is the total length of the stretched arm. We need to set the target point at that distance to make it a valid target point for inverse kinematics. We run this procedure twice, once for each arm (the difference is the position of the corresponding ShoulderPitch joints), and finally obtain the solution(s) from inverse kinematics for the left and right arms for the corresponding target points. If both solutions are returned, the robot raises both arms pointing to the ball. If only one solution is returned, the robot raises only one arm; the other arm cannot point to the ball, as it is not possible to reach the correct target point due to physical constraints. Figure 5.2 shows snapshots from this demonstration.

This demonstration was presented by team Kouretes at the SPL Open Challenge Competition at RoboCup 2012 in Mexico City, Mexico. Figure 5.3 shows a picture taken during the open challenge presentation.

## 5.4 Demonstration II: Basic CoM Balancing

In the second demonstration, we seek to implement a very basic balancing method. In particular, we want to make NAO move one of its feet to the point of the projection of the CoM on the floor. First, we calculate the position of the CoM using forward kinematics,

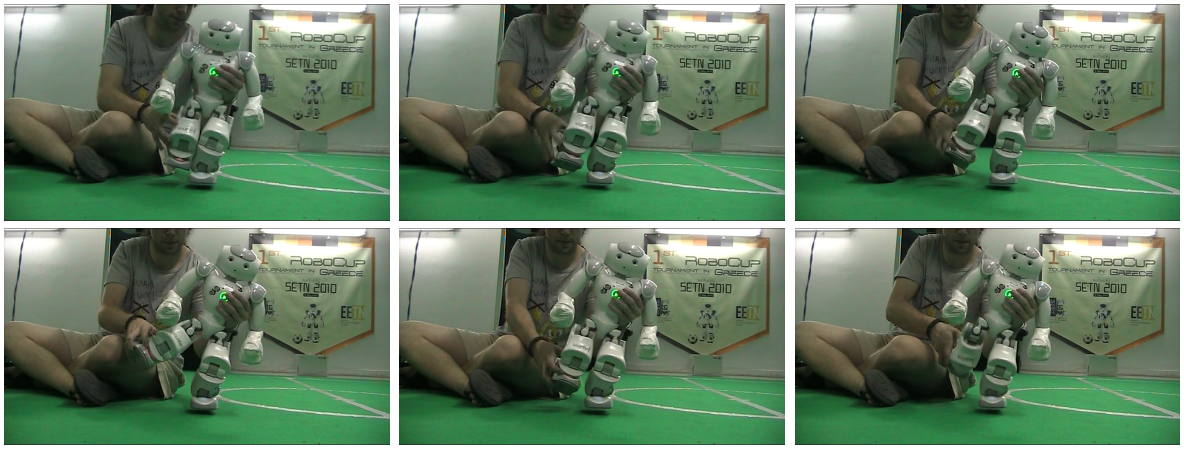


Figure 5.4: Basic balancing for the NAO using the projection of the center of mass

which gives the CoM position relatively to the torso frame. The problem is that the $x$-$y$ plane of the torso frame is rarely parallel to the floor. Thus, we read off the inertial unit of the robot the current rotation (angleX and angleY) of the torso plane. Now, we can calculate the position of the CoM relatively to the rotated torso:

$$T_{\text{rotated}} = R_y(\text{angleY})R_x(\text{angleX})A_{\text{CoM}}$$

Then, we assign a custom value to $p_z$ in $T_{(4,3)}$, which represents the desired torso height from the floor, so, and that yields $T_{\text{rotated}'}$. Now we must rotate back to the torso frame:

$$T_{\text{final}} = \big(R_y(\text{angleY})R_x(\text{angleX})\big)^{-1}T_{\text{rotated}'}$$

Finally, we extract $p_x$, $p_y$, and $p_z$ from $T_{\text{final}}$ and we set $(p_x, p_y, p_z)$ as the target position for inverse kinematics. The target orientation is set to $(a_x, a_y, a_z) = (-\text{angleX}, -\text{angleY}, 0)$, because we do not care about the rotation around the $z$-axis.

Figure 5.4 shows snapshots from this demonstration. Note that the foot is always parallel to the floor. It happens some times, that some robots have displaced inertial unit (accelerometers and gyro-meters). On these robots the foot is not parallel to the floor, but this represents a hardware problem, not a kinematics problem.

# Chapter 6

# Conclusion

Kinematics is the base for several applications related to robot motion. Our approach to NAO kinematics is based on standard principled methods for studying robot kinematic chains. Nevertheless, no complete analytical solution for the NAO robot had been published before. Our work offers such a complete analytical solution, which we expect will be useful not only to RoboCup SPL teams, but also to any NAO software developer.

## 6.1 Future Work

The work in this thesis can be used as the base for a several future research directions, some of which are listed below. It is also a step towards making the software architecture of our team independent from Aldebaran's development framework.

#### Omni-Directional Walk

The ability of a robot to walk towards any desired direction is called omni-directional walk. The feet (and arms) trajectories in omni-directional walk engines are being calculated dynamically and for this reason an inverse kinematics mechanism is more than necessary for the robot to be able to follow them.

#### Dynamic Balancing

As the robot walks, kicks, or performs any kind of motion, it must maintain its balance. Knowledge of the position of the center of mass at all times is more than necessary for

successful balancing.

### Kick Engine

Currently, our team relies only on static predefined kicks designed. The problem with those kicks is that they cannot absorb random disturbances and quite often robots executing them fall down. With inverse kinematics and balancing in place, the team can develop a dynamic kick engine, which takes care of balancing the robot on one leg, while following a dynamic kick trajectory based on the ball's position with the other leg.