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, Jule 2012

oΩ●oΩ●oΩ●oΩ●oΩ●oΩ●oΩ●oΩ●oΩ●oΩ●oΩ●oΩ●oΩ●

oΩ●oΩ●oΩ●oΩ●oΩ●oΩ●oΩ●

oΩ●oΩ●oΩ●oΩ●oΩ●oΩ●

oΩ●oΩ●oΩ●oΩ●oΩ●oΩ●oΩ●oΩ●oΩ●oΩ●oΩ●oΩ●oΩ●oΩ●

oΩ●oΩ●oΩ●oΩ●oΩ●oΩ●oΩ●oΩ●oΩ●oΩ●oΩ●oΩ●oΩ●oΩ●

oΩ●oΩ●oΩ●oΩ●oΩ●oΩ●oΩ●oΩ●oΩ●oΩ●oΩ●oΩ●oΩ●oΩ●

oΩ●oΩ●oΩ●oΩ●oΩ●oΩ●oΩ●oΩ●oΩ●oΩ●oΩ●oΩ●oΩ●oΩ●

o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●

o′Ω●o′Ω●o′Ω● o-

′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●

o-

′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●

o′Ω●o′Ω●o′Ω● o′Ω●o′Ω● o-

′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′

o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω● Nαo

o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●
o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●

o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω● o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●
o′Ω●o′Ω●o′Ω●o′Ω●. o′Ω●o′Ω●o′Ω●. o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω● o′Ω●.
o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω● (o′Ω●o′Ω●o′Ω●o′Ω●)
o′Ω●o′Ω●o′Ω●. o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●
o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω● (o′Ω●o′Ω●o′Ω●o′Ω●)
o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●. o′Ω●o′Ω●o′Ω●. o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●
o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω● (o′Ω●o′Ω●o′Ω●o′Ω●)

o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●, o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω●o′Ω● 2012

First of all, i would like to thank Manolis Orf (a.k.a. “re palikari”)

Abstract

agsdgdsagdsasdagesdag

Περίληψη

δφσαφσαδδσαφσδαφ

Contents

List of Figures

LIST OF FIGURES

Chapter 1

Introduction

NAO is the robot that is used for the soccer games in RoboCup SPL league. In this league, all the teams are using the same robot and the same hardware, too. Thus, the teams need to implement their own omni-directional walk [1] [2], motions and balancing system if they want to be more competitive. The creation of dynamic movements, as the above, are not feasible without an inverse kinematics mechanism. Also, creation of a balancing system is impossible without the calculation of the center of mass using forward kinematics. This mechanism must be fast, because the RoboCup is a real-time environment. This thesis describes a forward kinematics solution for every chain of NAO and an analytical solution for the problem of inverse kinematics without any approximations.

1.1 Thesis Contribution

As it mentioned before, there is a need to know the position of an end effector and the execution of dynamic trajectories. With this thesis we succeeded to implement the mechanism that translates the joints of a chain to a Cartesian position for the end effector. Also, we created a mechanism that translates dynamic trajectories to joint values in real execution time. The contribution of this thesis to our SPL team, Kouretes, is the mechanism of inverse kinematics that makes possible the creation of our own omni-directional walk and kick engine.

1.2 Thesis Outline

Chapter 2 provides a background for the RoboCup competition and the SPL league, as well as a brief description about the Kouretes SPL team. Furthermore, it describes the affine transformation matrices, DH parameters and robot kinematics. In Chapter 3 we provide a complete description of the hardware of NAO as well as the problem of kinematics for NAO. Moreover, in Chapter 4 we will discuss the related work about forward and mainly inverse kinematics. In Chapter 5 we will describe analytically our solutions to the problem of forward and inverse kinematics. Also, we will explain the implementation of kinematics to fit the team code. In Chapter 6 we will show execution times and demos about the implementation of kinematics. Finally, in Chapter 7 we will discuss the results of this thesis and we will compare them with other, similar works as well as pointing out possible future directions.

Chapter 2

Background

2.1 RoboCup

The RoboCup competition was initially inspired by Hiroaki Kitano [3] in 1993 and then his idea led to the foundation 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 that participate in RoboCup have to find solutions to some of the most difficult problems, that the robot community has to solve, in real time (perception, cognition, action, co-ordination). All the competitions in RoboCup (soccer, RoboCup@Home, RoboRescue etc.) are testing the solutions that teams find for the problems above. So far, the researchers that participate in RoboCup have made a lot of progress to solve real-world problems that are presented through the RoboCups competitions.

2.1.1 The Standard Platform League

The Standard Platform League (SPL) is one of the soccer leagues of RoboCup. In this league all the teams use the same robot, Aldebaran-NAO, and they focus only in the software. The teams are prohibited to make any changes to the hardware of the robot, so that everyone use the same platform and compete only in software. The robots are complete autonomous and no human interaction, from the team members, is allowed during the games. The only interaction with

the outer world, that robots have, is the data that Game Controller sends, which declare the state of the game. Current the games are conducted in a field with dimensions $4m \times 8m$ [4]. The field is consists from a green carpet with white lines and 2 yellow goals. The appearance of the field is similar with the real soccer field but it is scaled. The ball is a polished orange street hokey ball. Each team consists of four robots and each robot has a band, the color of the band defines the team (blue or pink). The total game time is twenty minutes and it is broken in two half, its of them has a duration of 10 minutes.

Figure 2.1: RoboCup Standar Platform League

2.2 Kouretes Robocup SPL Team

Kouretes is the first RoboCup team founded in Greece at the Tech-nical University of Crete in February 2006. The first participation of the team was in the Technical Challenges of Robocup 2006 in Bremen, Germany, when still the Four-Legged Sony AIBO robots were used. Next year the team participated also in the Four-Legged league of the RoboCup German Open 2007 competition in

Hannover, Germany and ranked in the 7th/8th place. Months later the team's participation in the MSRS Simulation Challenge at RoboCup 2007 in Atlanta led to the placement of the team at the 2nd place worldwide. In RoboCup 2008 in Suzhou, China, Kouretes team participated and won two trophies: 1st place in the SPL-MSRS league and 3rd place in the SPL-NAO league. In 2009 the team participated both at the RoboCup German Open 2009 competition in Hannover and in RoboCup 2009 in Graz, Austria, in the SPL. In 2010 the team participated in the 1st ever RoboCup Mediterranean Open competition in Rome and in RoboCup 2010 in Singapore. In 2011, the team participated in the RoboCup German Open 2011 competition in Megdeburg, in Iran Open 2011 in Tehran and in RoboCup 2011 in Istanbul. Team Kouretes joined Team Noxious from UK to form a joint team for RoboCup 2011. The joint team won the 2nd place in the SPL Open Challenge competition with 139 points, only 3 points behind the top team. In 2012 the team Kouretes participated in the RoboCup German Open 2012 competion in Magdebug, in Iran Open 2011 in Tehran and in RoboCup 2012 in Mexico. The team, in Mexico, has succeed to qualify to the top 16 teams.

Figure 2.2: Kouretes team in RoboCup 2012 at Mexico City

2.3 Affine Transformations

An affine transformation is a map transferring points and vectors from one space to another, being able to preserve ratios of distances. The space 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. Its flexibility concerning object manipulation, makes it a very useful tool in computer graphics.
For the purpose of this thesis we only used rotation and translation, so we will analyze these two types of affine transformation. Also we are working in a $3-dimensional$ space and all the examples from now on will be in this space.

Affine Transformation Matrix

The affine transformation matrix is a $(n+1 \times n+1)$ matrix where n is the number of dimensions. In the general form the affine transformation matrix consists of 2 parts:

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

X is a $(n \times n)$ matrix and Y is a $(n \times 1)$ vector and the last line has $n-1$ zeros followed by a 1. The matrix is invertible if and only if X is invertible and the representation of the inverse is:

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

Now if we want to apply the transformation, to a given column vector v, we multiply the affine transformation matrix with the given vector:

$$v' = Tv = \begin{bmatrix} & X & & Y \\ \begin{bmatrix}0 & 0 & 0\end{bmatrix} & & & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$$

A matrix that is the result of n multiplications between affine transformation matrices is still an affine transformation. So in the following example, given that

all the matrices in the right part of the equation are affine transformation matrices then the resulting matrix T will be an affine transformation matrix.

$$T = T_1 T_2 T_3 \cdots T_n$$

Translation Matrix

Translation, in the Euclidean space, is a function that moves every point by constant distance in a specified direction. We can describe the translation in the $3 - dimensional$ space with a (4×4) table that has the following form:

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

The d_x, d_y, d_z defines the distance that we will move all of our points through the x, y, z dimension. Obviously, this is an affine transformation matrix with $X = I$. So now if we want to apply the translation on a given column vector v, we apply it in same way as the transformation:

$$v' = Tv = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$$

Rotation Matrix

Rotation matrix is a matrix that is used to perform rotation in Euclidean space. The rotation matrices are always orthogonal matrices with determinant 1.

$$R^T = R^{-1}, det(R) = 1$$

In the $3 - dimensional$ Euclidean space there are three rotation matrices, each one of them makes a rotation over the x or y or z axis. The size of the rotation matrices, for this space, is (3×3) and they have the following form:

$$R_z = \begin{bmatrix} \cos\theta_z & -\sin\theta_z & 0 \\ \sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_y = \begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y \\ 0 & 1 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y \end{bmatrix}$$

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x \\ 0 & \sin\theta_x & \cos\theta_x \end{bmatrix}$$

Now if we want to rotate an object point, defined by the (3×1) column vector v, by the x axis and then by the y axis we will do the following:

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

We can use combinations of those 3 matrices and make other useful rotation matrices. For example we can construct the rotation matrix that rotates all the axes with the following order, first the z axis then the y axis and finally the x axis.

$$R = R_z R_y R_x$$

The analytical form of the above rotation matrix:

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

Finally we can easily transform a rotation matrix to an affine transformation matrix just by padding with zeros and an one in the corner. So from now on whenever we use a rotation matrix, this matrix will have the following form:

$$R' = \begin{bmatrix} & R & & \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} & & & 1 \end{bmatrix}$$

Affine Transformation Matrix in this Thesis

In this thesis we are using the rotation and the translation matrix so we can move points in the $3 - dimensional$ space. We can assume that our affine transformation matrix is composed by a rotation and a translation matrix. The affine

transformation matrix, that we will work with, has $X = R$ and $Y = v$ where v is the vector with the distance points from the translation matrix.

$$T = \begin{bmatrix} & R & & \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \\ \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} & & & 1 \end{bmatrix}$$

Also we can decompose this transformation matrix to a rotation matrix followed by a translation matrix. Given a rotation R and a translation matrix A:

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

$$A = \begin{bmatrix} & I & & \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \\ \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} & & & 1 \end{bmatrix}$$

The result from the multiplication of R with A is:

$$T = RA = \begin{bmatrix} & R & & R \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \\ \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} & & & 1 \end{bmatrix}$$

So we can assume that the translation matrix A' is:

$$A' = \begin{bmatrix} & I & & R \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \\ \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} & & & 1 \end{bmatrix}$$

Now if we multiply A' with the R we will have the same transformation matrix as before:

$$A'R = \begin{bmatrix} & I & & R \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \\ \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} & & & 1 \end{bmatrix} \begin{bmatrix} & R & & \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} & & & 1 \end{bmatrix} = \begin{bmatrix} & R & & R \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \\ \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} & & & 1 \end{bmatrix} = RA$$

2.4 Robot Kinematics

Robot kinematics is the way to apply the geometry to the study of the multi-degree of freedom kinematic chains. Kinematic chain is the assembly of links connected by joints. The degree of freedom (DOF) refers to the number of joints in the kinematic chain. Robot kinematics is the way to go from the *joint* space, where the kinematic chains are defined, to the Euclidean space and vice versa. They are, also, very useful because we can plan and control movement and calculate actuator forces and torques.

Joint Space

Kinematic chain typically is a manipulator that interacts with the environment. The joints are controlling the manipulator. Not all the combinations of all joints positions in the chain are valid, because some combinations lead to collisions between the links of the chain or with some item of the environment. All the valid combinations create the *joint* space.

2.4.1 Forward Kinematics

The *joint* space reveals very little information about the position of the end effector of the kinematic chain. With the forward kinematics we can move from the *joint* space to the $3-dimensional$ space. Given a kinematic chain and all the current joint values, forward kinematics can find the point p_x, p_y, p_z and the orientation a_x, a_y, a_z of the end effector of the kinematic chain. Forward kinematics is not an application specific problem and can be addressed to any kinematic chain.

2.4.2 Inverse Kinematics

Generally it is very easy for humans to give a target point or design a trajectory in $3-dimensional$ space, but if we want the end effector to follow the trajectory, we must assign the right values to the joints of the kinematic chain. So, we need a way to go from $3-dimensional$ space back to *joint* space. The problem of inverse kinematics is application specific and every kinematic chain has a different solution. The solution of the problem can be analytical or it can be approximate

(e.g with Jacobian approximation method). As the DOF increases, each point in the $3-dimensional$ space may have more than one matching point in the *joint* space, so we may have more than one solution in the *joint* space for a given point in $3-dimensional$ space.

2.5 DH (Denavit Hartenberg) Parameters

Denavit and Hartenberg found a way to create a transformation matrix that describes points in one end of the joint to a coordinate system that is fixed to the other end, as a function of the joint state [5] [6]. They found that we can construct this transformation matrix using only 4 parameters, the DH (Denavit Hartenberg) Parameters. These parameters are:

$$a, \alpha, d \text{ and } \theta$$

Before we can explain these parameters we must first establish the reference frame of each joint respectively to the previous joint reference frame.

- The z_n-axis is in the direction of the joint axis.
- The x_n-axis is parallel to the common normal: $x_n = z_n \times z_{n-1}$. The direction of x_n is from z_{n-1} to z_n.
- The y_n-axis follows from the x_n and z_n-axis by choosing it to be a right-handed coordinate system.

Now we can describe the DH parameters as following:

- a: length of the common normal.
- *alpha*: angle about common normal, from z_{n-1}-axis to z_n-axis.
- d: offset along z_{n-1} to the common normal.
- θ: angle about z_{n-1}, from x_{n-1} to x_n.

There is a great video that visualizes how to calculate all the above parameters (http://www.youtube.com/watch?v=rA9tm0gTln8). Now we can go from one reference frame to the other by composing the transformation matrix:

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

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

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

We can notice that the above transformation matrix is an affine transformation matrix because it is the product of the multiplication of affine transformation matrices.

2.6 Mathematica

Mathematica© is a software tool for mathematic computations created by the Wolfram company (http://www.wolfram.com/). It is very useful, because it can find solution to differential equations and can very easily execute symbolic computations. For the purpose of this thesis, we want a software with the capability to perform fast and large symbolic computations with matrices. Also it has the capability to simplify the symbolic results really fast.
The following code is a small example for the symbolic computation. We construct two matrices with cosines and sines and we have two symbols, theta1 and theta2. Next we just do the multiplication between those to matrices and 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 symbolic computations are the following:

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

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

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

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

The example above is very simple, but illustrates the Mathematica simplification step, which is very importand for our work, where he have to deal with much larger and more complex matrices.

Chapter 3

Problem statement

3.1 The NAO Robot

Aldebaran NAO is a humanoid robot. It is 58 cm tall and it has 5kg approximate mass. The version we are working on is the RoboCup version 3.3 with 21 DOF. It has 2 DOF on the head, 4 on each arm and 5 on each leg and 1 DOF that is common between the two legs. NAO has five kinematic chains and they are the following:

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

The common joint is the HipYawPitch joint, so LHipYawPitch and RHipYawPitch are the same joint.
In the figure bellow we are presenting the academic edition of NAO. NAO RoboCup edition misses 4 DOF on the hand (LWristYaw, LHand, RWristYaw and RHand).

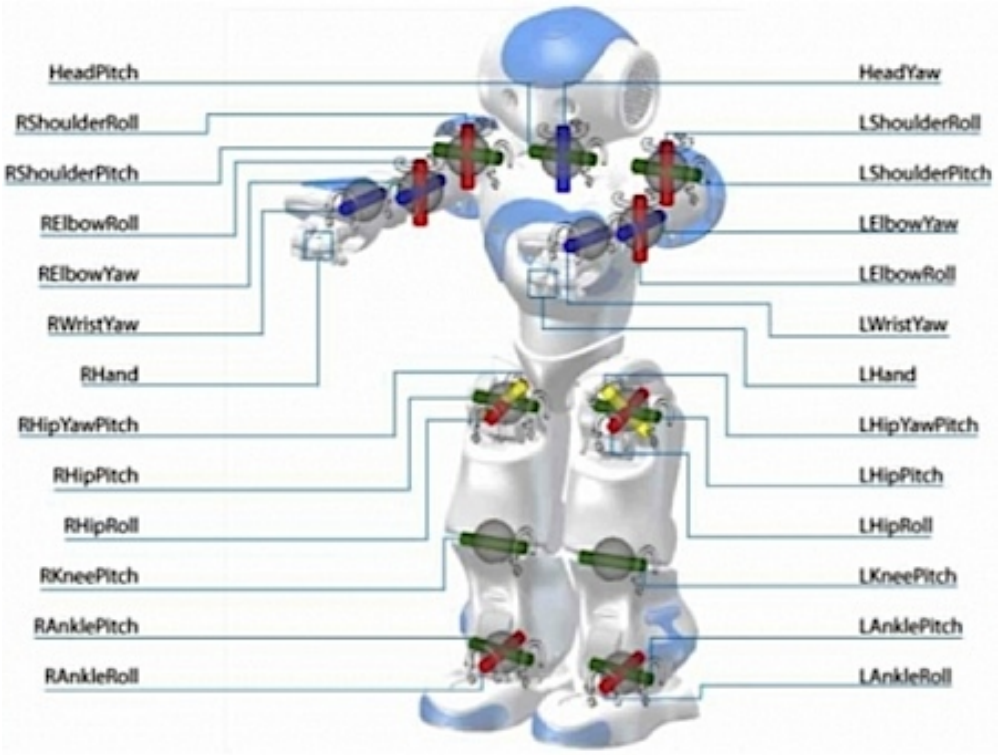
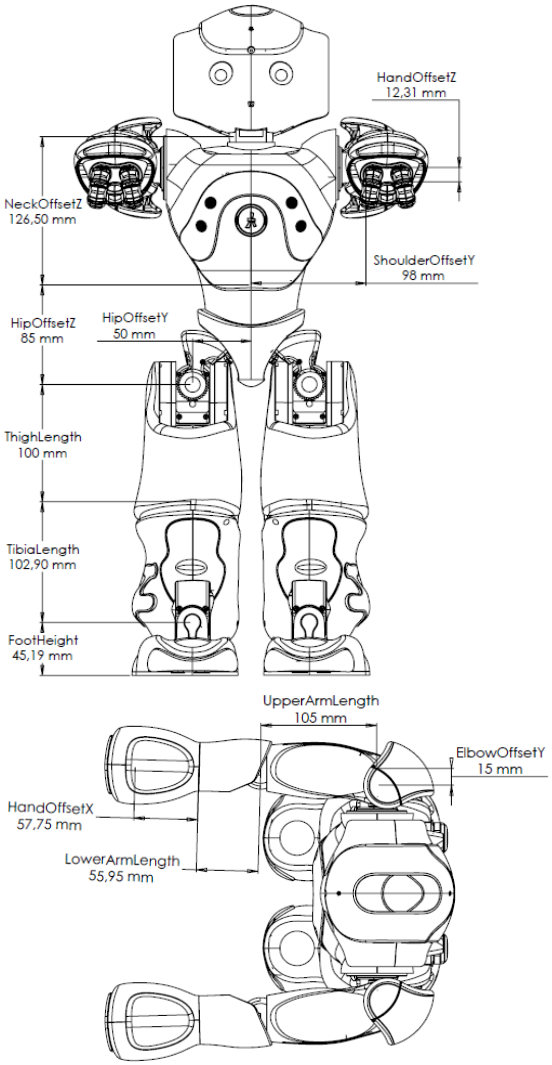


Figure 3.1: Aldebaran NAO V3.3 DOF of Academic Edition

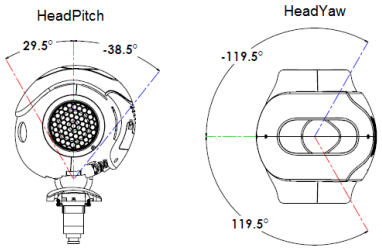
3.1.1 NAO Cartesian Restrictions

In the tables bellow we will present the length of all the links of the robot, the range that each joint is working in radians and in degrees and finally the masses of the NAO. Each joint has its own mass and center of mass. The center of mass for each joint is represented by a point in the $three - dimensional$ space of the joint. The robot theoretically is symmetric but we will see, in the joints ranges, that some left joints have different range than the right ones. The values are extracted from the user manual of the robot that Aldebaran provides.
The user manual has only masses for the right part of the robot but we assume that the robot is fully symmetrical on the masses.
Although some joints appear to be able to go to a position, the hardware controller of the robot will prohibit this movement because of the possible collisions with NAO shell.



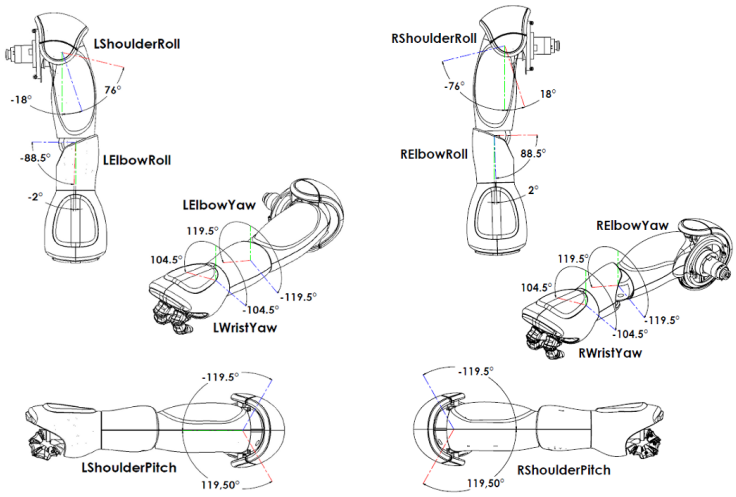
Name	Size (mm)
NeckOffsetZ	126.50
ShoulderOffsetY	98.00
ElbowOffsetY	15.00
UpperArmLength	105.00
LowerArmLength	55.95
ShoulderOffsetZ	100.00
HandOffsetX	57.75
HipOffsetZ	85.00
HipOffsetY	50.00
ThighLength	100.00
TibiaLength	102.90
FootHeight	45.19
HandOffsetZ	12.31

Figure 3.2: NAO links sizes



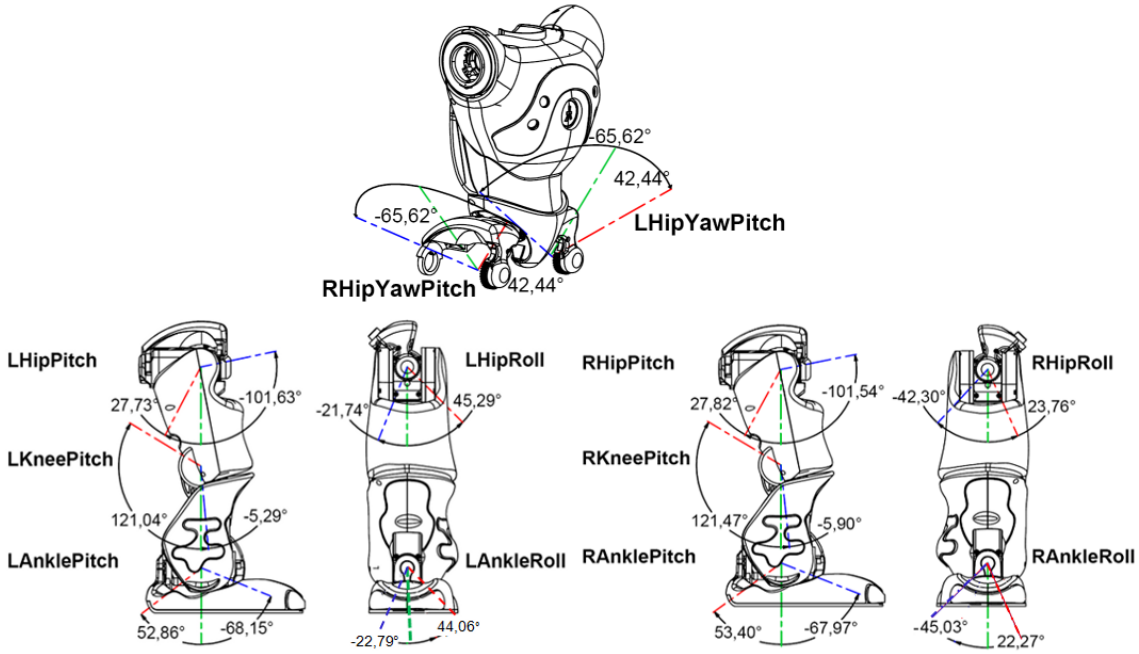
Joint Name	Range in Degrees°	Range in Radians
HeadYaw	-119.5° to 119.5°	-2.0857 to 2.0857
HeadPitch	-38.5° to 29.5°	-0.6720 to 0.5149

Figure 3.3: Head joints 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: Arms joints 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: Legs joints range

Table 3.1: Masses of NAO

Masses for NAO v3.3 robocup edition (H21)				
Total Mass For NAO H21				4.879 kg
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

3.2 Kinematics for NAO

Because NAO has a lot of DOF, it can do several complex moves. Some examples of those complex moves are walking, kicking a ball, standing up etc. NAO has five kinematic chains, three of them completely independent and two of them with one common joint. Kinematics is very useful for the programmers of NAO because they can create dynamic movements with the use of inverse kinematics or they can find the horizon of the camera on the head of the robot using forward kinematics.

3.2.1 The Forward Kinematics problem for NAO

Forward kinematics for NAO can been seen as five individual solutions. NAO has five chains and because we will not manipulate the joints, but we will only take the current state of each joint, we can assume that the five chains are completely independent. Then, we can find five forward kinematics solutions, one for each kinematic chain and we will be able to combine these solutions to find a solution for a bigger kinematic chain (e.g. the kinematic chain from the right foot to the head). The reason for dealing with this problem is twofold:firstly it is of great importance (concerning many applications, such as, finding its horizon, as well as placing it correctly).Besides this, as it will be described below, solution to the inverse kinematics problem would be intractable without solving the forward first.

3.2.2 The Inverse Kinematics problem for NAO

The inverse kinematics problem is a more complex problem and although we have the common joint between two kinematic chains, we are working with the assumption that we have five completely independent kinematic chains, so we will find five solutions for this problem. The reasons that led us to solve it are many. More specifically, we need to create dynamic kicks and an omni-directional walk. Both problems need a mechanism to move trajectories from the $three-dimensional$ space to the $joint$ space and the inverse kinematics can provide this mechanism.

Analytic versus Approximate solutions

Inverse kinematics problem can be solved analytically or approximately. We have chosen to find the analytical solution rather than the approximate solution because the second one has some problems. We need real-time execution to the problem of inverse kinematics. The analytical solution is faster than the fastest approximate solution and for this reason it is a good choice for real-time execution. Second, there are various implementations of approximate solutions for the problem. Some of them are fast but it is very possible to run into a singularity. Other methods have a smaller probability to run into a singularity, but they are slower. On the other hand, the analytical solution doesn't have any singularity. However, inverse kinematics have an analytical solution only if the chain has five or less DOF. If the chain has 6 DOF, it can have analytical solution to the problem of the inverse kinematics only if three sequential joints have intersecting axes.

Chapter 4

Related Work

The problem of forward and inverse kinematics is a familiar problem for all the teams that participate in the RoboCup SPL. The solution to the problem of forward kinematics is very easy and all the teams have implemented their own. There are not too many known solutions for the problem of inverse kinematics for NAO robot. On the other hand, Aldebaran provides an approximate solution for this problem but as we will see below we can't use this solution in our approach. Also, some teams have published their own analytical solutions.

4.1 Aldebaran Forward and Inverse Kinematics Solution

Aldebaran Forward Kinematics Solution

Aldebaran provides a forward kinematics mechanism but the problem is that it provides the solution only for the current state of the robot. So you can't find the position of the camera at the time a specific picture was taken given the joints values. Also, as we said before, we must find the solution for this problem if we want to solve inverse kinematics problem. However, Aldebaran provides us with the DH parameters for all the joints of the robot and that was very useful.

Aldebaran Inverse Kinematics Solution

Aldebaran has implemented in the API of the robot some functions that move an end effector to a given point in the $3-dimensional$ space. This functions are

using the Jacobian approximate method to find the solution to the problem of inverse kinematics. The omni-directional walk that Aldebaran provide us with, uses this solution to execute the trajectories. Although this solution is accurate, it can easily fall into a singularity and if this happens, it will stuck in there. That is a very bad problem because then the whole motion of the robot gets stuck.

4.2 BHuman Inverse Kinematics Solution

B-Human is a RoboCup SPL team from the university of Bremen in Germany. Every year they publish the code release [7], the code that they used in the last RoboCup and a documentation for this code. In the code release they have an inverse kinematics solution for the legs of NAO but with an approximation. The solution provided, always makes the end effector parallel to the plane of the torso, that is defined by the z and x axis. So this approximation supplies us with a solution that moves the end effector to the target point but with different orientation from the target orientation.

4.3 QIAU Inverse Kinematics Solution

MRL SPL team is a team from the university of QIAU in Teheran. They have a paper [8] at `www.academia.edu` in which they present a solution that theoretically solves the problem of inverse kinematics for the legs. We have tried to implement this solution, but the results were not satisfactory.

Chapter 5

Kinematics for NAO: Our approach

As we mentioned before, there are already solutions for both problems but none of them is completely suitable for our needs. We want to be able to find a solution to the forward kinematics problem with any joint values for input and not only with the current values of the joint of NAO. Also, we need a real-time analytical solution for the problem of inverse kinematics without any approximations. Below we will describe our solutions for both of these problems.

5.1 Forward Kinematics for the NAO Robot

Aldebaran provides us with the DH parameters for each kinematic chain of the robot. The problem is that for some chains the given parameters are incorrect. More specifically, the parameters for the NAO arms are incorrect, so we found our own parameters for the arms and we used the parameters that Aldebaran gave us for the legs and the head.

NAO Zero Position

We must define the base frame of the NAO and the zero position before we continue to the solution. First, the base frame is the torso frame and in the figure 5.1 we can see the axis of this frame. In this figure we can also see the

Aldebaran's zero position, that is different than the zero position that we are using. Our zero position is with the arms down and not extended.

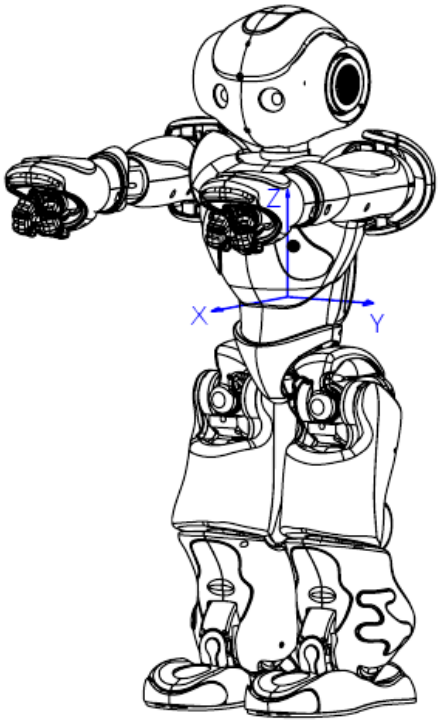


Figure 5.1: Torso Frame

Symbols

We are going to give a simple explanation to the symbols that we are using for our math calculations. First of all, all the matrices that we are using are affine transformation matrices. Second, we have three types of matrices: T, R, A. T is the transformation matrix, R_x, R_y, R_z are the basic rotations matrices and A is the translation matrix. The subscript of the symbol refers to the start frame and the superscript refers to the destination frame. The torso is the point that all the kinematic chains start and it is located on the center of the body of NAO. 'Base' will be the start frame and so will be the torso, and 'End' will be the end effector. The numbers will refer to one of the joints of the kinematic chain, as we see in the tables in Chapter 3. Also, we define the initialization of the translation matrix

as: $A(d_x, d_y, d_z)$ and for the rotation matrices as: $R_x(\theta_x)$ or $R_y(\theta_y)$ or $R_z(\theta_z)$. We will present the DH parameters in tables and except from the DH parameters, in the tables we will have the translations from the base to the first joint and from the last joint to the end effector. Also, we will have some necessary rotations to adjust the frame of the last joint to the frame of the end effector.

Forward Kinematics equations

Forward kinematics for each chain of the NAO robot is an equation that transforms a point from the frame of the last joint to the base frame. In our case we will have an end effector that will be the point of interest. So, we can construct these equations with the use of transformation, rotation and translation matrices.

Extracting the position

As we will see, the result of forward kinematics is an affine transformation matrix with X submatrix to be a rotation matrix and Y a translation vector. We need to extract the p_x, p_y and p_z points and the a_x, a_y and a_z angles of the final position. So, we can extract p_x, p_y and p_z from the translation part of the matrix:

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

Now we must extract the a_x, a_y and a_z from the rotation table. The rotation of the final transformation table is a $R_z R_y R_z$ rotation table. In chapter '2' we present the analytical form of this table. Now it's easy to extract the angles:

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

5.1.1 Forward Kinematics for the Head

The head of NAO is the simplest kinematic chain but, it has two possible end effectors, the top and the bottom camera position on the head. As we will see in our solution, it is easy to change the end effector by just changing the last translation matrix. Below, there is the table with all the DH parameters for this chain and the two possible end effectors.

Frame (Joint)	**a**	$\boldsymbol{\alpha}$	**d**	$\boldsymbol{\theta}$
Base	$A(0, 0, \text{NeckOffsetZ})$			
HeadYaw	0	0	0	θ_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	$A(\text{bottomCameraX}, 0, \text{bottomCameraZ})$			

Table 5.1: DH parameters for Head chain

Now we can combine the tables and find the point of the end effector in the frame space of the torso:

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

The DH transformation matrices are T^1_0 and T^2_1. The translation of the end effector can be replaced by one of the two possible translation matrices. Now we have the position of the end effector in the $three - dimensional$ space of the torso.

5.1.2 Forward Kinematics for the Left Arm

The kinematic chain for the left arm consists of four joints. Below we can see the table with the DH parameters for all joints and with the necessary translations and rotations. Now we can easily calculate the final transformation table:

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

Frame (Joint)	**a**	$\boldsymbol{\alpha}$	**d**	$\boldsymbol{\theta}$
Base	$A(0, \text{ShoulderOffsetY+ElbowOffsetY}, \text{ShoulderOffsetZ})$			
LShoulderPitch	0	$-\frac{\pi}{2}$	0	θ_1
LShoulderRoll	0	$\frac{\pi}{2}$	0	$\theta_2 - \frac{\pi}{2}$
LElbowYaw	0	$-\frac{\pi}{2}$	UpperArmLength	θ_3
LElbowRoll	0	$\frac{\pi}{2}$	0	θ_4
Rotation	$R_z(\frac{\pi}{2})$			
End effector	$A(\text{HandOffsetX+LowerArmLength}, 0, 0)$			

Table 5.2: DH parameters for Left Arm chain

The DH transformation matrices are the T_0^1, T_1^2, T_2^3 and T_3^4. Now we have the transformation table for the position of the end effector of the left arm relatively to the frame of the torso.

5.1.3 Forward Kinematics for the Right Arm

The kinematic chain of the right arm is symmetric with the chain of the left arm relatively to the plain defined by x and z-axis. So, the differences with the left arm will be only in the distances along y-axis and in the joints that move the y-axis (angle roll joints). Also, in this chain we must add one extra rotation matrix after the final translation, because the z-axis is inverted.

Frame (Joint)	**a**	$\boldsymbol{\alpha}$	**d**	$\boldsymbol{\theta}$
Base	$A(0, \text{-ShoulderOffsetY - ElbowOffsetY}, \text{ShoulderOffsetZ})$			
RShoulderPitch	0	$-\frac{\pi}{2}$	0	θ_1
RShoulderRoll	0	$\frac{\pi}{2}$	0	$\theta_2 + \frac{\pi}{2}$
RElbowYaw	0	$-\frac{\pi}{2}$	-UpperArmLength	θ_3
RElbowRoll	0	$\frac{\pi}{2}$	0	θ_4
Rotation	$R_z(\frac{\pi}{2})$			
End effector	$A(\text{-HandOffsetX-LowerArmLength}, 0, 0)$			
Rotation$'$	$R_z(-\pi)$			

Table 5.3: DH parameters for Right Arm chain

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

The DH transformation matrices are the T^1_0, T^2_1, T^3_2 and T^4_3. Now we have the transformation table for the position of the end effector of the right arm relatively to the frame of the torso.

5.1.4 Forward Kinematics for the Left Leg

The kinematic chain for the left leg has six joints and it is the biggest chain on the NAO. The DH parameters of these joints are slightly different because of the YawPitch joint.

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	θ_3
LKneePitch	-ThighLength	0	0	θ_4
LAnklePitch	-TibiaLength	0	0	θ_5
LAnkleRoll	0	$-\frac{\pi}{2}$	0	θ_6
Rotation	$R_z(\pi) R_y(-\frac{\pi}{2})$			
End effector	$A(0, 0, \text{-FootHeight})$			

Table 5.4: DH parameters for Left Leg chain

$$T^{\text{End}}_{\text{Base}} = A^{0}_{\text{Base}} T^{1}_{0} T^{2}_{1} T^{3}_{2} T^{4}_{3} T^{5}_{4} T^{6}_{5} R_z(\pi) R_y(-\tfrac{\pi}{2}) A^{\text{End}}_{6}$$

The DH transformation matrices are the $T^1_0, T^2_1, T^3_2, T^4_3, T^5_4$ and T^6_5. Now we have the transformation table for the position of the end effector of the left leg relatively to the frame of the torso.

5.1.5 Forward Kinematics for the Right Leg

As for the hands, the kinematic chains for the legs are symmetric relatively to the plain defined by x and z-axis. So, the differences of the chain for the right leg

will be only in the distances along y-axis and in the joints that move the y-axis.

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	θ_3
RKneePitch	$-ThighLength$	0	0	θ_4
RAnklePitch	$-TibiaLength$	0	0	θ_5
RAnkleRoll	0	$-\frac{\pi}{2}$	0	θ_6
Rotation	$R_z(\pi)R_y(-\frac{\pi}{2})$			
End effector	$A(0, 0, \text{-FootHeight})$			

Table 5.5: DH parameters for Right Leg chain

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

The DH transformation matrices are the $T^1_0, T^2_1, T^3_2, T^4_3, T^5_4$ and T^6_5. Now we have the transformation table for the position of the end effector of the right leg relatively to the frame of the torso.

5.1.6 Forward Kinematics for merged chains

We have found the solution only for the chains that have as base frame the torso frame. In the real word, someone maybe wants to find the torso position relatively to one of the feet chains. If we get the final transformation table for, e.g., the left leg, we know that it is an affine transformation matrix and because of that, we can invert the table and then we will have the position of the torso relatively to the frame of the left leg.

$$T^{\text{Torso}}_{\text{Lleg}} = \left(T^{\text{Lleg}}_{\text{Torso}}\right)^{-1}$$

Also, it is possible to find the position of the head relatively to the left leg. The kinematic chains for the head and for the left leg are relative to the torso frame.

So, if we invert the chain of the left leg, we will have the torso relatively to the left leg frame. Then we can just multiply the transformation table that has the position of the head relatively to the torso frame and we will have the position of the head relatively to the left leg frame.

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

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

5.1.7 Calculation of Center Of Mass with Forward Kinematics

The calculation of the center of mass (CoM) is very important because NAO consists of a group of moving parts. Every moving part has a mass and with forward kinematics we can calculate the position of the CoM relatively to a given frame. Aldebaran provides us with the whole information that is needed to do the calculation. Aldebaran gives us the mass of the whole robot and the mass for every moving part of NAO. The separate masses are given according to every joint of the robot. Thus, every joint of the robot has a mass and the position of the CoM for this joint relatively to the joint frame.

The CoM is calculated relatively to the torso frame and the calculation order is simple. We construct smaller kinematic chains that stop to an earlier joint, and then we set as the end effector the position of CoM for the frame of the joint where we stop. Then we get the translation part and we multiply it by the mass of the certain part. At the end, we will have 21 chains plus the torso chain. Then we will add all the individual translation matrices that have been weighted by the mass of each joint and the result will be divided by the total mass. The result will be the position of the CoM relatively to the torso frame.

5.2 Inverse Kinematics for NAO

Forward kinematics can find the position of an end effector, relatively to the start frame, given the joints values. Now we have to solve the inverse problem: find the joints values given the desired end effector position relatively to the torso frame. The solutions for the inverse kinematics problem that will be presented below are related to the kinematic chains that start from the torso.
The position of the end effector is the p_x, p_y and p_z points along with the a_x, a_y and a_z angles. As we mentioned before, the product of forward kinematics is an affine transformation matrix and it consists of a rotation and a translation matrix. The rotation R is equal to $R_z R_y R_x$. Thus, we can construct the transformation table:

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

As we mentioned before we can't solve the problem of inverse kinematics without the solution of forward kinematics. That's why the equations that we must solve to find the values of joints are the matrices from the forward kinematics but with the θ from DH parameters as the unknown part. So, we can find the symbolic table that is the product of forward kinematics and equate it with the matrix above. Then, we will have twelve equations (we don't have sixteen because the last line of the matrix is always (0 0 0 1)) and n unknowns, where n is the number of joints in the chain. In fact, we will have $2n$ unknowns, because all the thetas (θ) appear inside sin or cos.
As we will see below, we found some of the desired angles with arccos and arcsin. The problem is that arcsin returns a number in $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ and arccos returns a number in $[0, \pi]$ but the possible range of the joints is in $[-\pi, \pi]$. Thus, we will have two possible solutions for every arcsin and arccos (because of the complimentary angles). Because of that, we must filter our results and keep only the correct angles. We achieve that by doing a forward validation step when we find a complete set of angle values. Then we compare the result of the validation step with the reconstructed table and if they are equal, then we approve the set. Some

complimentary angles will be discarded a long time before the validation step due to the range restriction of each joint.

5.2.1 Inverse Kinematics for the Head

The head chain, as we said before, consists only of two joints. Thus, we can find both desired angles only by the a_z and a_y. On the other hand, someone maybe wants to find the joints for the desired position by only using the target points p_x, p_y and p_z, so we implement another solution that works only with the target points. Below we can see the symbolic result from forward kinematics:

$$T = \begin{bmatrix} -\cos\theta_1 \sin\theta_2 & -\sin\theta_1 & \cos\theta_1 \cos\theta_2 & l_2 \cos\theta_1 \cos\theta_2 - l_1 \cos\theta_1 \sin\theta_2 \\ -\sin\theta_1 \sin\theta_2 & \cos\theta_1 & \cos\theta_2 \sin\theta_1 & l_2 \cos\theta_2 \sin\theta_1 - l_1 \sin\theta_1 \sin\theta_2 \\ -\cos\theta_2 & 0 & -\sin\theta_2 & l_3 - l_2 \sin\theta_2 - l_1 \cos\theta_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where: $l_1 = \text{cameraX}$, $l_2 = \text{cameraZ}$ and $l_3 = \text{NeckOffsetZ}$.
Because we only know the p_x, p_y and p_z and we can't reconstruct the rotation part of the matrix, we will only use the translation part that is reconstructed. Now we can see from the symbolic matrix that $T_{(3,4)} = l_3 - l_2 \sin\theta_2 - l_1 \cos\theta_2 = p_z$ and we know from trigonometry that:

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

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

So, we can now calculate θ_2:

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

Now the final θ_2 is: $\theta_2 = \theta_2 - \frac{\pi}{2}$ because in the DH parameters we had added $\frac{\pi}{2}$ to the θ parameter for the second joint. From now on, we must substract $\frac{\pi}{2}$ from θ_2 if we want to use it to calculate a result from the equations.
Then we can easily extract θ_1 from $T_{(1,4)}$:

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

Finally, we must filter our results because of arccos and arcsin. So, we will do a forward kinematics validation set and we will discard any set of values that doesn't move the end effector to the correct position.

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

$$\theta_2 = \arcsin\left(\frac{-p_z + l_3}{\sqrt{l_1^2 + l_2^2}}\right) + \frac{\pi}{2}$$
$$\theta_1 = \arccos\left(\frac{p_x}{l_2\cos\left(\theta_2 - \frac{\pi}{2}\right) - l_1\sin\left(\theta_2 - \frac{\pi}{2}\right)}\right)$$

or only with angles

$$\theta_1 = a_z$$
$$\theta_2 = a_y$$

5.2.2 Inverse Kinematics for the Left Hand

The left arm chain is far more complicated than the head chain. First, we must construct the symbolic matrix:

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

$r_{11} = \sin\theta_4\left(\sin\theta_1\sin\theta_3 - \cos\theta_1\cos\theta_2\cos\theta_3\right) - \cos\theta_1\cos\theta_4\sin\theta_2$

$r_{12} = \cos\theta_4\left(\sin\theta_1\sin\theta_3 - \cos\theta_1\cos\theta_2\cos\theta_3\right) + \cos\theta_1\sin\theta_2\sin\theta_4$

$r_{13} = \cos\theta_3\sin\theta_1 + \cos\theta_1\cos\theta_2\sin\theta_3$

$r_{14} = l_4\left(\sin\theta_4\left(\sin\theta_1\sin\theta_3 - \cos\theta_1\cos\theta_2\cos\theta_3\right) - \cos\theta_1\cos\theta_4\sin\theta_2\right) - l_3\cos\theta_1\sin\theta_2$

$r_{21} = \cos\theta_2\cos\theta_4 - \cos\theta_3\sin\theta_2\sin\theta_4$

$r_{22} = \cos\theta_2\sin\theta_4 - \cos\theta_3\cos\theta_4\sin\theta_2$

$r_{23} = \sin\theta_2\sin\theta_3$

$r_{24} = l_1 + l_3\cos\theta_2 + l_4\left(\cos\theta_2\cos\theta_4 - \cos\theta_3\sin\theta_2\sin\theta_4\right)$

$$r_{31} = \sin\theta_4\left(\cos\theta_1\sin\theta_3 + \cos\theta_2\cos\theta_3\sin\theta_1\right) + \cos\theta_4\sin\theta_1\sin\theta_2$$
$$r_{32} = \cos\theta_4\left(\cos\theta_1\sin\theta_3 + \cos\theta_2\cos\theta_3\sin\theta_1\right) - \sin\theta_1\sin\theta_2\sin\theta_4$$
$$r_{33} = \cos\theta_1\cos\theta_3 - \cos\theta_2\sin\theta_1\sin\theta_3$$
$$r_{34} = l_2 + l_4\left(\sin\theta_4\left(\cos\theta_1\sin\theta_3 + \cos\theta_2\cos\theta_3\sin\theta_1\right) + \cos\theta_4\sin\theta_1\sin\theta_2\right) + l_3\sin\theta_1\sin\theta_2$$

where: l_1 = ShoulderOffsetY+ElbowOffsetY, l_2 = ShoulderOffsetZ, l_3 = UpperArmLength and l_4 = HandOffsetX+LowerArmLength.
As we can see now, the problem of inverse kinematics for the arms is far more complicated than the problem for the head. It is very difficult to extract a joint value from all these equations, so, we can use trigonometry to find the value of one joint. More specifically this joint is the elbow roll joint. We can observe that the upper arm, the lower arm with the hand offset and the distance from the position of the shoulder pitch joint to the end effector forms a triangle and we know the size of each side. The position of the shoulder pitch joint relative to the torso frame is known, and the position of the end effector is the target position. So, the distance can be calculated:

$$d = \sqrt{\left(s_x - p_x\right)^2 + \left(s_y - p_y\right)^2 + \left(s_z - p_z\right)^2}$$

where: $s_x = 0, s_y = l_1$ and $s_z = l_2$.
Now we can use the cosine law and find θ_4:

$$\theta_4 = \arccos\left(\frac{{l_3}^2 + {l_4}^2 - d^2}{2l_3l_4}\right)$$

Because θ_4 represents an interior angle and the elbow roll joint is being stretched in the zero-position, the resulting angle is computed by:

$$\theta_4 = \pi - \theta_4$$

Because the θ_4 for the left arm take only negative values, we will take the $-\theta_4$ so:

$$\theta_4 = -\theta_4$$

The next angle that we can find is the θ_2, so we can look for equations from the table symbolic matrix, where we only have θ_2 and one more unknown θ. Using r_{22} we have:

$$T_{(2,2)} = -\cos\theta_2 - \cos\theta_3\cos\theta_4\sin\theta_2 \qquad \Leftrightarrow$$

$$\cos\theta_3 \sin\theta_2 = -\frac{\cos\theta_2 \sin\theta_4 + T_{(2,2)}}{\cos\theta_4}$$

We can do the division because θ_4 doesn't reach $\frac{\pi}{2}$ so, the cosine never becomes zero. Now we can go to r_{24}:

$$T_{(2,4)} = p_y = l_1 + l_3\cos\theta_2 + l_4\left(\cos\theta_2\cos\theta_4 - \cos\theta_3\sin\theta_2\sin\theta_4\right) \quad \Leftrightarrow$$

$$p_y - l_1 = l_3\cos\theta_2 + l_4\cos\theta_2\cos\theta_4 - l_4\left(-\frac{\cos\theta_2\sin\theta_4 + T_{(2,2)}}{\cos\theta_4}\right)\sin\theta_4 \quad \Leftrightarrow$$

$$p_y - l_1 - \frac{l_4\sin\theta_4 T_{(2,2)}}{\cos\theta_4} = \cos\theta_2\left(l_3 + l_4\cos\theta_4 + l_4\frac{\sin\theta_4{}^2}{\cos\theta_4}\right) \quad \Leftrightarrow$$

$$\theta_2 = \arccos\left(\frac{p_y - l_1 - \frac{l_4\sin\theta_4 T_{(2,2)}}{\cos\theta_4}}{l_3 + l_4\cos\theta_4 + l_4\frac{\sin^2\theta_4}{\cos\theta_4}}\right) \qquad \text{because } l_3 + l_4\cos\theta_4 + l_4\frac{\sin^2\theta_4}{\cos\theta_4} > 0$$

Now the final θ_2 is: $\theta_2 = \theta_2 + \frac{\pi}{2}$ because in the DH parameters we had added $\frac{\pi}{2}$ to the θ parameter for the second joint of the arm. From now on, we must substract $\frac{\pi}{2}$ from θ_2 if we want to use it to calculate a result from the equations. Next we will calculate the θ_3 angle:

$$T_{(2,3)} = \sin\left(\theta_2 - \frac{\pi}{2}\right)\sin\theta_3 \quad \Leftrightarrow$$

$$\theta_3 = \arcsin\left(\frac{T_{(2,3)}}{\sin\left(\theta_2 - \frac{\pi}{2}\right)}\right) \qquad \text{because } \theta_2 \neq \left|\frac{\pi}{2}\right|$$

Finally, we must extract the value for θ_1:

$$T_{(1,3)} = \cos\theta_3\sin\theta_1 + \cos\theta_1\cos\left(\theta_2 - \frac{\pi}{2}\right)\sin\theta_3 \quad \Leftrightarrow$$

$$\sin\theta_1 = \frac{T_{(1,3)} - \cos\theta_1\cos\left(\theta_2 - \frac{\pi}{2}\right)\sin\theta_3}{\cos\theta_3} \qquad \text{when } \theta_3 \neq \left|\frac{\pi}{2}\right|$$

$$\cos\theta_1 = \frac{T_{(1,3)}}{\cos\left(\theta_2 - \frac{\pi}{2}\right)\sin\theta_3} \qquad \text{when} \quad \theta_3 = \left|\frac{\pi}{2}\right|$$

So, now we have two possible solutions for θ_1 and we choose between them depending on the value of θ_3. So, when $\theta_3 = \left|\frac{\pi}{2}\right|$:

$$\theta_1 = \arccos\left(\frac{T_{(1,3)}}{\cos\left(\theta_2 - \frac{\pi}{2}\right)\sin\theta_3}\right)$$

Else if $\theta_2 \neq \left|\frac{\pi}{2}\right|$, we will find the θ_1 from r_{33} and we will replace $\sin\theta_1$ with the result above. So:

$$T_{(3,3)} = \cos\theta_1\cos\theta_3 - \cos\left(\theta_2 - \frac{\pi}{2}\right)\sin\theta_1\sin\theta_3 \quad \Leftrightarrow$$

$$T_{(3,3)} = \cos\theta_1\cos\theta_3 - \left(\frac{T_{(1,3)} - \cos\theta_1\cos\left(\theta_2 - \frac{\pi}{2}\right)\sin\theta_3}{\cos\theta_3}\right)\sin\theta_3\cos\left(\theta_2 - \frac{\pi}{2}\right) \quad \Leftrightarrow$$

$$\theta_1 = \arccos\left(\frac{T_{(3,3)} - \frac{T_{(1,3)}\sin\theta_3\cos\left(\theta_2 - \frac{\pi}{2}\right)}{\cos\theta_3}}{\cos\theta_3 + \frac{\cos^2\left(\theta_2 - \frac{\pi}{2}\right)\sin^2\theta_3}{\cos\theta_3}}\right)$$

Now we have calculated all the joints for the left arm, but we will have a lot of invalid angle sets. We will find the correct set after the forward kinematics validation step. Below there are all the inverse kinematics joint's equations for the left arm:

$$\theta_4 = -\left(\pi - \arccos\left(\frac{{l_3}^2 + {l_4}^2 - \sqrt{(s_x - p_x)^2 + (s_y - p_y)^2 + (s_z - p_z)^2}^2}{2l_3l_4}\right)\right)$$

$$\theta_2 = \pm\arccos\left(\frac{p_y - l_1 - \left(\frac{l_4\sin\theta_4 T_{(2,2)}}{\cos\theta_4}\right)}{l_3 + l_4\cos\theta_4 + l_4\frac{\sin^2\theta_4}{\cos\theta_4}}\right) + \frac{\pi}{2}$$

$$\theta_3 = \arcsin\left(\frac{T_{(2,3)}}{\sin\left(\theta_2 - \frac{\pi}{2}\right)}\right)$$

$$\theta_3 = \pi - \arcsin\left(\frac{T_{(2,3)}}{\sin\left(\theta_2 - \frac{\pi}{2}\right)}\right)$$

$$\theta_1 = \pm\arccos\left(\frac{T_{(3,3)} - \frac{T_{(1,3)}\sin\theta_3\cos\left(\theta_2 - \frac{\pi}{2}\right)}{\cos\theta_3}}{\cos\theta_3 + \frac{\cos^2\left(\theta_2 - \frac{\pi}{2}\right)\sin^2\theta_3}{\cos\theta_3}}\right) \quad \text{if } \theta_3 \neq \frac{\pi}{2}$$

$$\theta_1 = \pm\arccos\left(\frac{T_{(1,3)}}{\cos\left(\theta_2 - \frac{\pi}{2}\right)\sin\theta_3}\right) \quad \text{if } \theta_3 = \frac{\pi}{2}$$

5.2.3 Inverse Kinematics for the Right Hand

As we said before, the right and left arm are symmetric, thus, the solution is almost the same. This differs from above only at the distances over the y-axis and for the right hand the θ parameter does not subtract $\frac{\pi}{2}$ but it adds $\frac{\pi}{2}$. The kinematic chain for the right arm has an extra rotation matrix after the last translation:

$$T_{\text{Base}}^{\text{End}} = A_{\text{Base}}^{0} T_0^1 T_1^2 T_2^3 T_3^4 R_z(\tfrac{\pi}{2}) A_4^{\text{End}} R_z(-\pi)$$

We can remove this rotation from the reconstructed matrix , and then the chain will be the same as the chain for the left arm. So, first of all we reconstruct the final transformation matrix from the target point parameters and then we remove the rotation:

$$T_{\text{rhandUnrotated}} = T_{\text{rhand}}(R_z(-\pi))^{-1}$$

The only changes on the symbolic table are in the translation part:

$$r_{14} = l_4\left(\sin\theta_4\left(\sin\theta_1\sin\theta_3 - \cos\theta_1\cos\theta_2\cos\theta_3\right) - \cos\theta_1\cos\theta_4\sin\theta_2\right) + l_3\cos\theta_1\sin\theta_2$$
$$r_{24} = -l_1 - l_3\cos\theta_2 - l_4\left(\cos\theta_2\cos\theta_4 - \cos\theta_3\sin\theta_2\sin\theta_4\right)$$
$$r_{34} = l_2 - l_4\left(\sin\theta_4\left(\cos\theta_1\sin\theta_3 + \cos\theta_2\cos\theta_3\sin\theta_1\right) + \cos\theta_4\sin\theta_1\sin\theta_2\right) - l_3\sin\theta_1\sin\theta_2$$

As we can see, the changes don't have any impact in our solution for the left arm, e.g. no one denominator becomes zero with these changes. So, as in the previous section, we will get the equations below:

$$T_{\text{rhandUnrotated}} = T_{\text{rhand}}(R_z(-\pi))^{-1}$$
$$\theta_4 = \pi - \arccos\left(\frac{{l_3}^2 + {l_4}^2 - \sqrt{(s_x - p_x)^2 + (s_y - p_y)^2 + (s_z - p_z)^2}^2}{2 l_3 l_4}\right)$$
$$\theta_2 = \pm\arccos\left(\frac{-p_y - l_1 - \left(\frac{l_4\sin\theta_4 T_{(2,2)}}{\cos\theta_4}\right)}{l_3 + l_4\cos\theta_4 + l_4\frac{\sin^2\theta_4}{\cos\theta_4}}\right) - \frac{\pi}{2}$$
$$\theta_3 = \arcsin\left(\frac{T_{(2,3)}}{\sin\left(\theta_2 + \frac{\pi}{2}\right)}\right)$$

$$\theta_3 = \pi - \arcsin\left(\frac{T_{(2,3)}}{\sin\left(\theta_2 + \frac{\pi}{2}\right)}\right)$$

$$\theta_1 = \pm\arccos\left(\frac{T_{(3,3)} - \frac{T_{(1,3)}\sin\theta_3\cos\left(\theta_2+\frac{\pi}{2}\right)}{\cos\theta_3}}{\cos\theta_3 + \frac{\cos^2\left(\theta_2+\frac{\pi}{2}\right)\sin^2\theta_3}{\cos\theta_3}}\right) \qquad \text{if } \theta_3 \neq \frac{\pi}{2}$$

$$\theta_1 = \pm\arccos\left(\frac{T_{(1,3)}}{\cos\left(\theta_2+\frac{\pi}{2}\right)\sin\theta_3}\right) \qquad \text{if } \theta_3 = \frac{\pi}{2}$$

5.2.4 Inverse Kinematics for the Left Leg

The kinematic chain for the legs has six joints, so, it will be much more difficult to find a solution. The symbolic matrix for this chain was too huge, therefore we want to make it smaller. First, to make the matrix smaller, we will remove the known rotations and translations from the kinematic chain:

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

$$T_{\text{lleg}'} = T_{\text{lleg}} \left(A^{\text{End}}_6\right)^{-1}$$

$$T_{\text{lleg}''} = \left(A^0_{\text{Base}}\right)^{-1} T_{\text{lleg}'}$$

Now we have the chain from the base frame of the first joint to the frame of the last joint. The first joint, HipYawPitch, is rotated by $\frac{-3\pi}{4}$ in the x-axis. We will add a rotation a the start of the chain to rotate the x-axis by $\frac{\pi}{4}$. Then the first joint will be a yaw joint and then we will have three intersected joints and the problem will becomes solvable:

$$T_{\text{llegRotated}} = R_x(\tfrac{\pi}{4}) T_{\text{lleg}''}$$

Now the end effector is the last joint (ankle roll) and the base is the first joint (rotated HipYawPitch). The first four joints are responsible for the position and orientation of the end effector and the other two joints are only responsible for the orientation. It would be easier if we have only three joints responsible for the

position of the end effector, thus, we will invert the transformation matrix. Now only the ankle roll, ankle pitch and knee pitch are responsible for the position:

$$T_{\text{llegInv}} = (T_{\text{llegRotated}})^{-1}$$

The resulted symbolic matrix is pretty large and we will use only the translation part, so:

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

where l_1 = ThightLength and l_2 = TibiaLength.
We can now find θ_4 with the same way that we find θ_4 for the arms. So, we have a triangle with the following sides: ThighLength, TibiaLength and the distance from the base to the end effector:

$$d = \sqrt{(s_x - p_x)^2 + (s_y - p_y)^2 + (s_z - p_z)^2}$$

where $s_x = 0$, s_y =HipOffsetY, s_z =HipOffsetZ, $p_x = T_{\text{llegInv(1,4)}}$, $p_y = T_{\text{llegInv(2,4)}}$ and $p_z = T_{\text{llegInv(3,4)}}$. Now from the law of cosines we have:

$$\theta_4 = \arccos\left(\frac{{l_1}^2 + {l_2}^2 - d^2}{2 l_1 l_2}\right)$$

Because θ_4 represents an interior angle and the knee roll joint is being stretched in the zero-position, the resulting angle is computed by:

$$\theta_4 = \pi - \theta_4$$

Next, we will extract the θ_6 angle. We can extract this angle from the translation matrix using r_{24} and r_{34}:

$$\begin{aligned} \frac{r_{24}}{r_{34}} &= \frac{p_y}{p_z} && \Leftrightarrow \\ \frac{(l_2 \cos\theta_5 + l_1 \cos(\theta_4 + \theta_5)) \sin\theta_6}{(l_2 \cos\theta_5 + l_1 \cos(\theta_4 + \theta_6)) \cos\theta_5} &= \frac{p_y}{p_z} && \Leftrightarrow \\ \theta_6 &= \arctan\left(\frac{p_y}{p_z}\right) \quad \text{if } (l_2 \cos\theta_5 + l_1 \cos(\theta_4 + \theta_5)) \neq 0 \end{aligned}$$

Because we don't know θ_5, we can't find when this equation is zero. So we will have division by zero, but in reality we will not have this problem, because we are using the atan2 function in our program and the result will be just undefined. So, the program will continue to run but the validation step will reject all the solutions. In figure 5.2 we can see the locus of the equation, which gives the undefined points. In Subsection 5.2.6 we present the locus along with the values of the ankle pitch (θ_5) and knee pitch (θ_4) for a series of moves of the robot and we discuss what we do when we are in this locus.

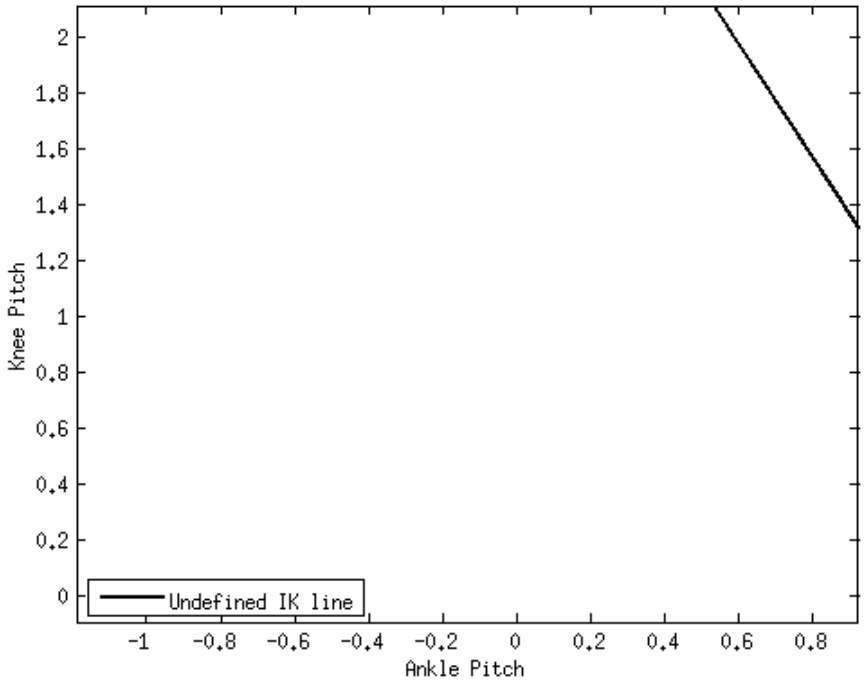


Figure 5.2: Undefined Locus For Legs

Now we can reconstruct and remove the rotation at the end and the transformation from ankle pitch to ankle roll from the chain to make it simpler:

$$T_{\text{llegRotated}'} = T_{\text{llegRotated}} \left(T_5^6 R_z\left(\pi\right) R_y(-\tfrac{\pi}{2}) \right)^{-1}$$

$$T_{\text{llegInv}'} = \left(T_{\text{llegRotated}'}\right)^{-1}$$

Now we have $p_x = T_{\text{llegInv}'(1,4)}$, $p_y = T_{\text{llegInv}'(2,4)}$ and $p_z = T_{\text{llegInv}'(3,4)}$. From the new symbolic transformation matrix we have:

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

We only need the translation part to extract θ_5, so:

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

The $l_2 + l_1 \cos\theta_4$ is zero if and only if $\cos\theta_4 = 1.029$. So it is always greater than zero, because $-1 \leq \cos \leq 1$:

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

We can do the division because ${l_1}^2 \sin^2\theta_4 + \left(l_2 + l_1 \cos\theta_4\right)^2$ is obviously greater than zero. Now we can remove the two transformations of θ_4 and θ_5 from the symbolic matrix and then we will have a transformation matrix that is a rotation matrix:

$$T_{\text{llegRotated}''} = T_{\text{llegRotated}'} \left(T_3^4 T_4^5\right)^{-1}$$

$$T_{\text{llegInv}''} = \left(T_{\text{llegRotated}''}\right)^{-1}$$

And the rotation part is:

$$
\begin{aligned}
r_{11} &= \cos\theta_1 \cos\theta_2 \cos\theta_4 - \sin\theta_1 \sin\theta_3 \\
r_{12} &= -\cos\theta_3 \sin\theta_1 - \cos\theta_1 \cos\theta_2 \sin\theta_3 \\
r_{13} &= \cos\theta_1 \sin\theta_2 \\
r_{21} &= -\cos\theta_3 \sin\theta_2 \\
r_{22} &= \sin\theta_2 \sin\theta_3 \\
r_{23} &= \cos\theta_2 \\
r_{31} &= -\cos\theta_2 \cos\theta_3 \sin\theta_1 - \cos\theta_1 \sin\theta_3 \\
r_{32} &= -\cos\theta_1 \cos\theta_3 + \cos\theta_2 \sin\theta_1 \sin\theta_3 \\
r_{33} &= -\sin\theta_1 \sin\theta_2
\end{aligned}
$$

Finally we can extract, from the transformation matrix, the remaining three angles:

$$
\begin{aligned}
\theta_2 &= \arccos T_{\text{llegInv}''(2,3)} \\
\theta_2 &= \theta_2 - \frac{\pi}{4} \\
\theta_3 &= \arcsin\left(\frac{T_{\text{llegInv}''(2,2)}}{\sin\left(\theta_2 + \frac{\pi}{4}\right)}\right) \\
\theta_1 &= \arccos\left(\frac{T_{\text{llegInv}''(3,3)}}{\sin\left(\theta_2 + \frac{\pi}{4}\right)}\right) \\
\theta_1 &= \theta_1 + \frac{\pi}{2}
\end{aligned}
$$

The equations above don't have any problem with division by zero because of the restriction of NAO. The HipRoll joint (θ_2) doesn't reach $-\frac{\pi}{4}$ or $\frac{3\pi}{4}$ so the denominator never becomes zero. Finally, we will do a forward validation step for all possible set of angles.
Below there are all the equations of inverse kinematics for the left leg:

$$
\begin{aligned}
T_{\text{llegInv}} &= \left(R_x(\tfrac{\pi}{4})\left(\left(A^0_{\text{Base}}\right)^{-1} T_{\text{lleg}}\left(A^{\text{End}}_6\right)^{-1}\right)\right)^{-1} \\
\theta_4 &= \pi - \arccos\left(\frac{{l_1}^2 + {l_2}^2 - \sqrt{(s_x - p_x)^2 + (s_y - p_y)^2 + (s_z - p_z)^2}^2}{2 l_1 l_2}\right)
\end{aligned}
$$

$$\theta_6 = \arctan\left(\frac{p_y}{p_z}\right) \qquad \text{if } (l_2\cos\theta_5 + l_1\cos(\theta_4+\theta_5)) \neq 0$$

$$T_{\text{llegInv}'} = \left((T_{\text{llegInv}})^{-1}\left(T_5^6 R_z(\pi)\, R_y(-\tfrac{\pi}{2})\right)^{-1}\right)^{-1}$$

$$\theta_5 = \arcsin\left(-\frac{p_y(l_2 + l_1\cos\theta_4) + l_1 p_x \sin\theta_4}{{l_1}^2\sin^2\theta_4 + (l_2 + l_1\cos\theta_4)}\right)$$

$$\theta_5 = \pi - \arcsin\left(-\frac{p_y(l_2 + l_1\cos\theta_4) + l_1 p_x \sin\theta_4}{{l_1}^2\sin^2\theta_4 + (l_2 + l_1\cos\theta_4)}\right)$$

$$T_{\text{llegInv}''} = \left((T_{\text{llegInv}'})^{-1}\left(T_3^4 T_4^5\right)^{-1}\right)^{-1}$$

$$\theta_2 = \pm\arccos T_{\text{llegInv}''(2,3)} - \frac{\pi}{4}$$

$$\theta_3 = \arcsin\left(\frac{T_{\text{llegInv}''(2,2)}}{\sin\left(\theta_2 + \frac{\pi}{4}\right)}\right)$$

$$\theta_3 = \pi - \arcsin\left(\frac{T_{\text{llegInv}''(2,2)}}{\sin\left(\theta_2 + \frac{\pi}{4}\right)}\right)$$

$$\theta_1 = \pm\arccos\left(\frac{T_{\text{llegInv}''(3,3)}}{\sin\left(\theta_2 + \frac{\pi}{4}\right)}\right) + \frac{\pi}{2}$$

5.2.5 Inverse Kinematics for the Right Leg

As we mentioned before, the chains of the legs are symmetric, so we will have a similar solution for the problem as we had with the arms. The only changes are in the rotation matrix that we use to rotate the HipYawPitch joint. Now we must rotate by $-\frac{\pi}{4}$.

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

$$T_{\text{rleg}'} = T_{\text{rleg}}\left(A^{\text{End}}_6\right)^{-1}$$

$$T_{\text{rleg}''} = \left(A^0_{\text{Base}}\right)^{-1} T_{\text{rleg}'}$$

$$T_{\text{rlegRotated}} = R_x(-\tfrac{\pi}{4}) T_{\text{rleg}''}$$

$$T_{\text{rlegInv}} = (T_{\text{rlegRotated}})^{-1}$$

After that, the symbolic matrix for the right leg is exactly the same as the symbolic matrix for the left leg. From this point of view, if we follow all the steps

as we did with the left leg, we will conclude to the same equations. So, the final equations are:

$$T_{\text{rlegInv}} = \left(R_x(\tfrac{\pi}{4}) \left(\left(A^0_{\text{Base}}\right)^{-1} T_{\text{rleg}} \left(A^{\text{End}}_6\right)^{-1} \right) \right)^{-1}$$

$$\theta_4 = \pi - \arccos\left(\frac{{l_1}^2 + {l_2}^2 - \sqrt{(s_x - p_x)^2 + (s_y - p_y)^2 + (s_z - p_z)^2}^2}{2 l_1 l_2} \right)$$

$$\theta_6 = \arctan\left(\frac{p_y}{p_z}\right) \qquad \text{if } (l_2 \cos\theta_5 + l_1 \cos(\theta_4 + \theta_5)) \neq 0$$

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

$$\theta_5 = \arcsin\left(-\frac{p_y (l_2 + l_1 \cos\theta_4) + l_1 p_x \sin\theta_4}{{l_1}^2 \sin^2\theta_4 + (l_2 + l_1 \cos\theta_4)} \right)$$

$$\theta_5 = \pi - \arcsin\left(-\frac{p_y (l_2 + l_1 \cos\theta_4) + l_1 p_x \sin\theta_4}{{l_1}^2 \sin^2\theta_4 + (l_2 + l_1 \cos\theta_4)} \right)$$

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

$$\theta_2 = \pm \arccos T_{\text{rlegInv}''(2,3)} + \frac{\pi}{4}$$

$$\theta_3 = \arcsin\left(\frac{T_{\text{rlegInv}''(2,2)}}{\sin\left(\theta_2 - \frac{\pi}{4}\right)} \right)$$

$$\theta_3 = \pi - \arcsin\left(\frac{T_{\text{rlegInv}''(2,2)}}{\sin\left(\theta_2 - \frac{\pi}{4}\right)} \right)$$

$$\theta_1 = \pm \arccos\left(\frac{T_{\text{rlegInv}''(3,3)}}{\sin\left(\theta_2 - \frac{\pi}{4}\right)} \right) + \frac{\pi}{2}$$

5.2.6 Undefined Target Points For Legs

As we mentioned before, we have some target points for which we can't find an inverse kinematics solution. These target points are presented in the figure 5.3, alongside with all the values of the angles that are responsible for this singularity. As we can see, we let the robot to do a lot of movements in the field, and none of those movements was close to the points of the singularity. In real world, it is very rare for someone to give target points in that area. Also, when someone is executing a trajectory, they "feed" inverse kinematics with a lot of target

points per second (usually they provide inverse kinematics with target points with frequency 50 to 100 Hz), so if one of those target points is in the area of singularity, we will find a solution for the next target point. Because we are working in high frequency rate, the singularity will be unnoticed and the movement will be continued normally.

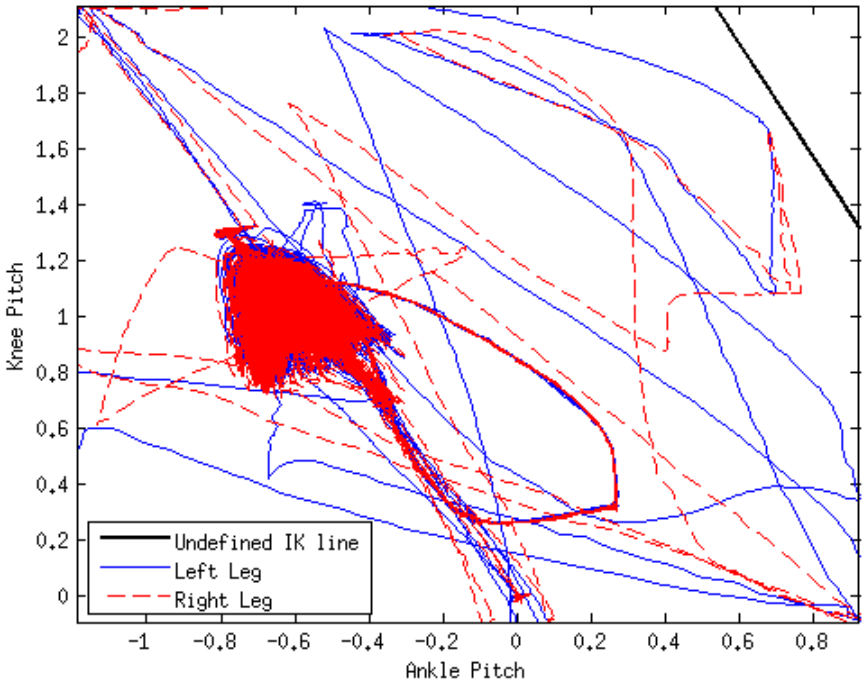


Figure 5.3: Undefined Target Points For Legs

5.3 Implementation

Now that we have all the equations for both problems, we must integrate them in the code of the team. The code of the team is written in `C++` so because `C++` doesn't have any library for fast linear algebra operations, we must develop a minimalistic matrix framework. Then, using this framework, we must write some

functions that will implement the equations of forward and inverse kinematics in C++.

5.3.1 KMat: Kouretes Math Library

KMat is a library developed by Emmanouil Orfanoudakis [9] that supports a strict subset of algebraic operations. The focus of the library is mainly in real numbers operations and the primary goal of KMat is low memory footprint and calculation efficiency. The typical linear algebra libraries do 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 (3×3) or (4×4)) and only a subset of operations (addition, subtraction, multiplication, scalar addition, scalar multiplication, transposition, inversion) have been implemented. KMat supports two type of matrices: Dense matrices and Affine transformation matrices.
For our work we used mainly the affine type of matrices and we expanded the functions of the library with some functions that are extremely useful for the kinematics computations. These functions are some initialization functions that initialize a matrix with the given DH parameters or reconstruct of the matrix given the target points etc.

5.3.2 Nao Kinematics in C++

Along with KMat we created two more libraries ForwardKinematics and InverseKinematics. ForwardKinematics has the functions to calculate the location of an end effector of a kinematic chain, given the joint values for this chain. The union of independent chains with common base frame is possible, so someone can take the position of the top camera relatively to one of the legs. Finally, we have a function to calculate the center of mass of the robot. The input of each function is the values of the joints with the order that they appear in the kinematic chains. The inverse kinematics library has five functions, each of which solves the problem for a kinematic chain. All functions take, as input, the position and the orientation of the target point and the output is the values of the joints, for all the possible solutions for this point. It is possible to have a solution with two set of values and then the user can decide what output will be used.

As we mentioned before, the legs have one common joint (HipYawPitch) but if we give two target points, one for each leg, inverse kinematics for the left leg will most likely return a different value for this joint than the inverse kinematics for the right leg. We must decide what value will be set to the HipYawPitch joint and one solution to this problem is to make the support leg (the leg that keeps the robot in balance) the master of this joint or another solution is to get the mean of these two values. Both solutions are not perfect but with a large probability, if we design trajectories carefully, the resulted values from inverse kinematics for this joint will be close enough. Thus, we can use any of these two solutions and the result will be similar to the result if the joint was independent.

Chapter 6

Results

We have a lot of ways to check if solutions for both problems are working correctly. The testing for forward kinematics is trivial; we just move the end effector to a position and we just check if forward kinematics returned the correct position and orientation. On the other hand, the testing of inverse kinematics was more difficult because we must give a valid target point if we want inverse kinematics to return a solution.
The problem is that we don't know the work space for the end effectors of NAO. So, we came up with another way to validate our solution. We moved one end effector to a random position, then we took the position and the orientation of this point from forward kinematics and then we assigned this point as the target point for inverse kinematics. Then we checked if inverse kinematics found a solution and we know that the solution inverse kinematics returned is correct, because there is a step of forward kinematics inside inverse kinematics that validates the output.
In table 6.1 we can see the execution times for each function (of inverse kinematics) in microseconds (us). Unfortunately, we can't compare these times with the Aldebaran's solution execution time, because it is not independent and along with the solution for inverse kinematics, it does the movement of the joints of the chain. So, we can't extract "clean" times from the execution.
We made two demos to present kinematics in real world. The first demo tests inverse kinematics of the arm along with forward kinematics of the legs. The second demo is more complicated and checks inverse kinematics of the legs and

the calculation of the center of mass.

Table 6.1: Execution Times

Function	**Time (us)**
Forward For Head	59.43
Forward For Left Arm	75.94
Forward For Right Arm	79.72
Forward For Left Leg	94.58
Forward For Right Leg	95.16
Inverse For Head	82.04
Inverse For Left Arm	205.91
Inverse For Right Arm	214.56
Inverse For Left Leg	203.32
Inverse For Right Leg	152.62
Calculation of CoM	410.30

6.1 Demo Program

Point To The Ball

In the first demo we want to make NAO point to the ball with the hands. We used, along with the kinematics, the vision [9] and the world state model of the robot. First, NAO searches for the ball and when it finds it, NAO points to the ball with one or two hands. With the help of the world state model, if NAO loses the ball, it can continue to point to the area that the ball must be according to the last vision observations.

The ball observation can be described as a *two*-dimensional point without orientation. The coordination of the ball is described as polar coordinates, so we have the distance and the angle of the observation relatively to the projection of the torso on the floor. We must now move this point from *two*-dimensions to *three*-dimensions and give it an orientation. First, we transform the polar coordinates to Cartesian coordinates (p_x, p_y) and we add the height of the torso as the

p_z. Now we know the position in the *three*-dimensional space and we also know that a_x equals to zero because we are only rotating y-axis (up/down) and z-axis (right/left). To find the two other angles, we get the straight line that connects the point of the ball and the position of the shoulder pitch joint relatively to the torso frame. Then we have that a_y equals to the angle between this line and the y-axis and similarly we find the a_z. Now we must calculate the position of the target point, which is in the line and from a distance d from the joint, where d is the size of the stretched arm (this is the maximum size of the arm). We need to bring the target point to that distance, because only then it will be a valid target point for inverse kinematics. We do this operation two times, one for each arm (different position of the shoulder joint) and then we give to inverse kinematics for left and right arm the resulted target points. Below there are images from the demo:

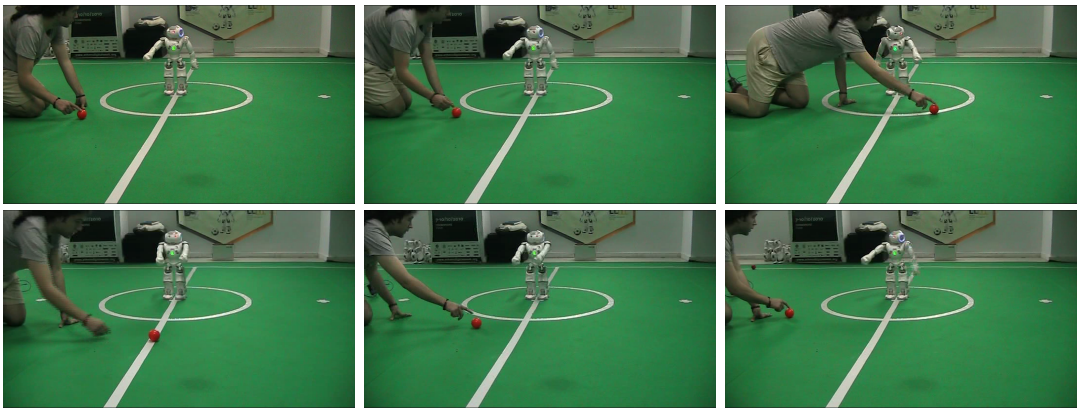


Figure 6.1: Point the ball using forward and inverse kinematics

Balancing Using The Center of Mass

In the second demo we want to make NAO to move one of its legs to the point of the projection of the CoM on the floor, so, we will have a very basic balancing system. First of all, we must calculate the position of the CoM using forward kinematics. Now we have the CoM position relatively to the torso frame. The problem is that the torso frame is never parallel to the floor. Thus, we get from the gyro-meters the angleX and angleY of the torso plain. Now we can take 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 put a custom value to p_z which represents the desired torso height from the floor, so, now we have $T_{\text{rotated}'}$. Now we must rotate back to the torso frame, so:

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

Finally, we set p_x, p_y and p_z from T_{final} as the target point for inverse kinematics and we set $a_x = -\text{angleX}$, $a_y = -\text{angleY}$ and $a_z = 0$ as the target rotation, because we don't care about rotation around z-axis. Now the target point is the projection of the CoM to the floor and the rotation of the end effector is parallel to the frame of the floor.

In the figures below we can see 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 we can see that the foot is not parallel to the floor but this is a hardware problem and not a kinematics problem.

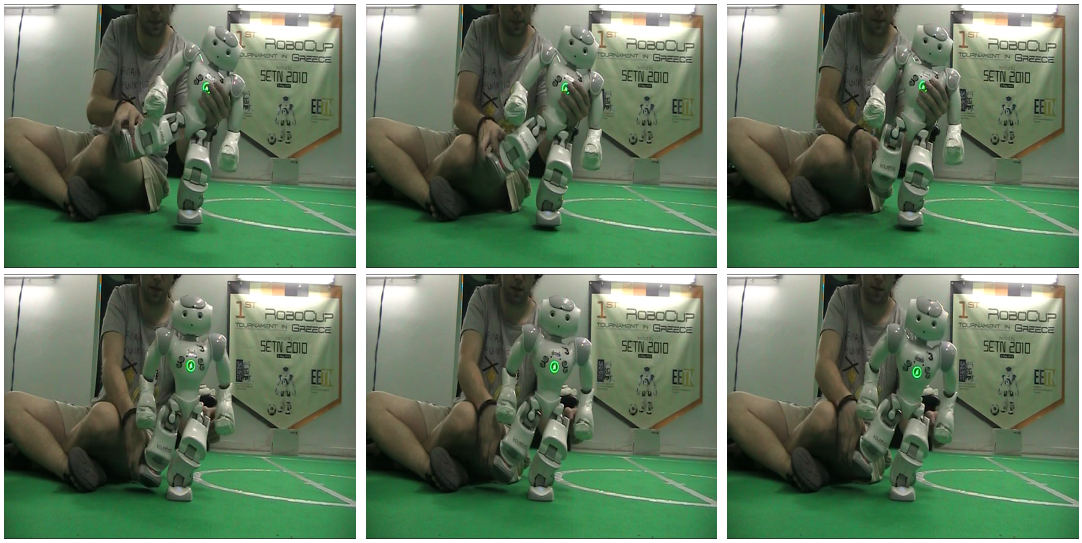


Figure 6.2: Basic balance system using center of mass

Chapter 7

Conclusion

Our approach is not a breakthrough because this is the way to solve kinematics for any chain. Despite this, Kouretes are immediately the only team in RoboCup SPL League that have an analytical solution without approximations for every kinematic chain of the NAO robot. Kinematics is the bases for a lot of applications, as it is described below, and having that implemented, we can expand our code and become more competitive against other teams, because our code lucks a good motion management. Furthermore, it is possible now to extract the accurate position of cammera's horizon being used by the vision module.

7.1 Future Work

Having described the inverse kinematics mechanism, it is now possible for Kouretes team to make their own independent movements (totally separated from Aldebaran's framework). Below, examples for future work follow:

Omni-Directional Walk

We want to make NAO walk to any direction, this is called omni-directional walk. The trajectories of walk are being calculated dynamically and for this reason, we need an inverse kinematics mechanism. Also, while the robot is walking, we must keep it in balance, so we need the position of the center of mass for the balancing system.

Kick Engine

It is crucial to fix a mechanism for dynamic kick executions. Now, our team, has only static kicks, designed before the game but the problem with those kicks is that there are not designed to be balanced and often some robots fall down when they execute these kicks. With inverse kinematics and balancing, we can make a kick engine, that takes care for the balance of the robot and execute a kick trajectory that is designed in run time.

References

[1] J. Strom, G.S., Chown, E.: Omnidirectional walking using zmp and preview control for the nao humanoid robot. Robocup Symposium (2009) 1

[2] Graf, C., Röfer, T.: A closed-loop 3d-lipm gait for the robocup standard platform league humanoid. In Pagello, E., Zhou, C., Behnke, S., Menegatti, E., Röfer, T., Stone, P., eds.: Proceedings of the Fifth Workshop on Humanoid Soccer Robots in conjunction with the 2010 IEEE-RAS International Conference on Humanoid Robots, Nashville, TN, USA (2010) http://www.informatik.uni-bremen.de/kogrob/papers/Humanoids-Graf-Roefer-10.pdf. 1

[3] Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., Osawa, E., Matsubara, H.: Robocup: A challenge problem for AI. AI Magazine **18**(1) (1997) 73–85 3

[4] RoboCup Technical Committee: Standard Platform League rule book (2012) 4

[5] Denavit, J., Hartenberg, R.: A kinematic notation for lower-pair mechanisms based on matrices. Trans ASME J. Appl. Mech (1955) 11

[6] Craig, J.J.: Introduction to Robotics: Mechanics and Control. Third edn. Pearson Hall (2004) 11

[7] Röfer, T., Laue, T., Müller, J., Fabisch, A., Feldpausch, F., Gillmann, K., Graf, C., de Haas, T.J., Härtl, A., Humann, A., Honsel, D., Kastner, P., Kastner, T., Könemann, C., Markowsky, B., Riemann, O.J.L., Wenk, F.: B-human team report and code release 2011 (2011) Only available online: `http://www.tzi.de/spl/pub/Website/Downloads/Rules2012.pdf`. 24

[8] Maani Ghaffari Jadidi, Ehsan Hashemi, M.A.Z.H.H.S.: Kinematic modeling improvement and trajectory planning of the nao biped robot. Published in htt[://www.academia.edu 24

[9] Orfanoudakis, E.: Reliable object recognition for the robocup domain. Diploma thesis, Technical University of Crete, Greece (2011) 48, 52