ENPM 662 – Introduction to Robot Modeling
Final Project

Forward & Inverse Kinematics of the Aldebaran NAO Robot- Computation, Validation and Simulation

---

Dipam Patel | 115809833
Instructor: Professor Chad Kassens

UNIVERSITY OF MARYLAND

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. The robot must keep its balance, self-collisions and collisions with obstacles in the environment must be avoided and, if applicable, the trajectory of the end-effector must follow the constrained motion of a manipulated object in Cartesian space. Humanoid service robots performing complex object manipulation tasks need to plan whole-body motions that satisfy a variety of constraints: These constraints and the high number of degrees of freedom make the whole-body motion planning for humanoids a challenging problem. The design of complex dynamic motions is achievable only using 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. 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.

Keywords- Aldebaran NAO, mobile robot, humanoid robot, forward kinematics, inverse kinematics, robosoccer

CONTENTS

1. INTRODUCTION

NAO is an autonomous, 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. The NAO Academics Edition was developed for universities and laboratories for research and education purposes. It was released to institutions in 2008 and was made publicly available by 2011. NAO robots have been used for research and education purposes in numerous academic institutions worldwide.

It distinguishes itself from its existing Japanese, American, and other counterparts thanks to its pelvis kinematics design, its proprietary actuation system based on brush DC motors, its electronic, computer and distributed software architectures. This robot has been designed to be affordable without sacrificing quality and performance. It is an open and easy-to-handle platform where the user can change all the embedded system software or just add some applications to make the robot adopt specific behaviours. The robot's head and forearms are modular and can be changed to promote further evolution.

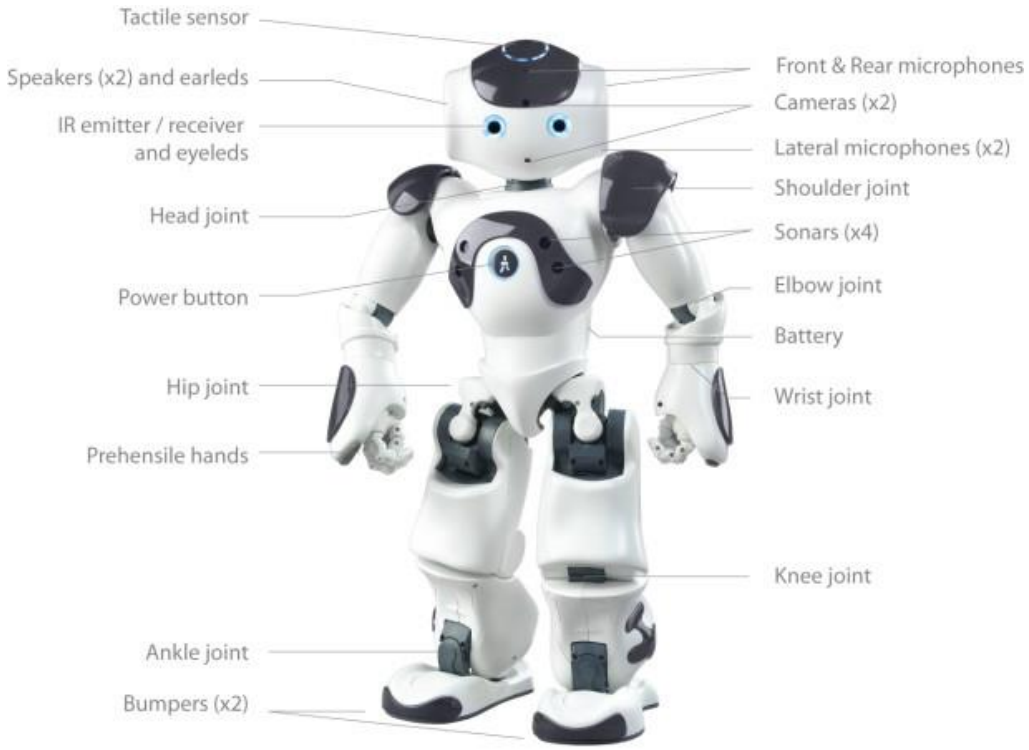


Figure-1: Aldebaran NAO v3.3 (Academic edition) components

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.

Research into the components used in this project was done to better understand how the robot works and the capabilities of the components that would be used. The NAO robot has many different sensors, motors, and joints. There are 14 touch sensors, located on the head, chest, hands, and feet. Some of these are capacitive while others are simple on/off switches. These sensors respond when being pushed, and, except for the chest button, are not utilized unless specified for an application. The chest button is used to turn the robot on and off, as well as make the robot say its IP address, store a position, etc. depending on what state the robot is in. The reactions to any of these sensors being pushed are specified within applications and can range from a vocal reaction to one involving movement from the robot.

These articulated robots with multiple degrees of freedom 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 using the robot kinematics, which is an application of geometry to the study of arbitrary robotic chains. This project using [1] as the main guideline, studies the problems of forward and inverse kinematics for the Aldebaran NAO humanoid robot and presents a complete analytical solution to both problems with no approximations.

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.

1.1 Motivation

It is widely known that the design of complex dynamic motions is achievable only using 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.

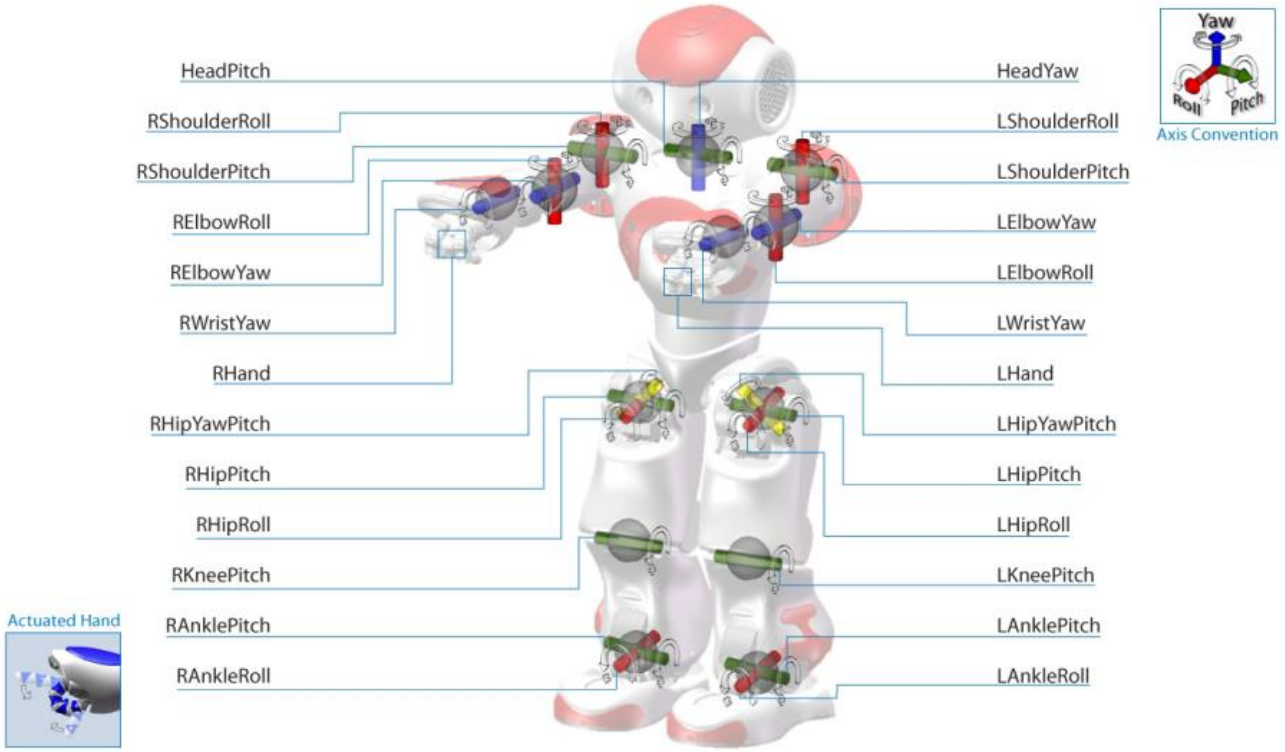


Figure-2: Aldebaran NAO v3.3 (Academic edition) kinematic chains and joints

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.

This project intends 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 to that, finding a real-time analytical solution for the problem for the forward and inverse kinematics without any approximations. This project is targeted to solve 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, 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.

2. BACKGROUND OF NAO

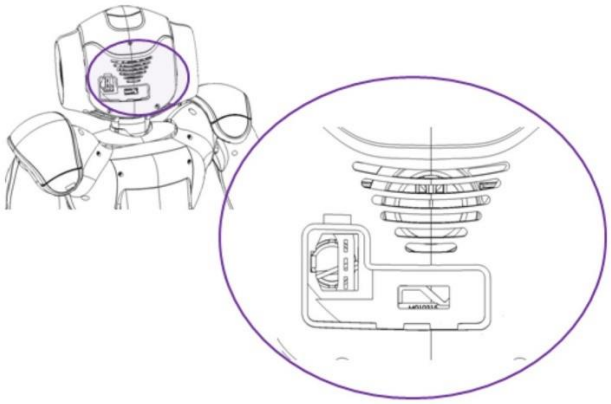
2.1 Aldebaran NAO Humanoid Robot

2.1.1 Types of Humanoid Robots

Many upgrades of NAO body type exist. They can be differentiated with the head's back design.

NAO Versions

NAO V4



NAO V3.3

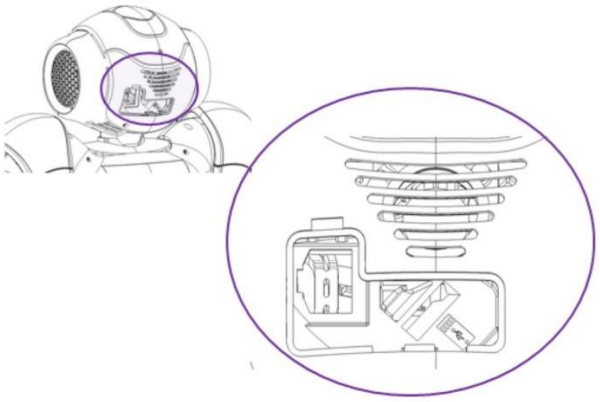


Figure-3: NAO Versions-1

NAO V3+, V3.2

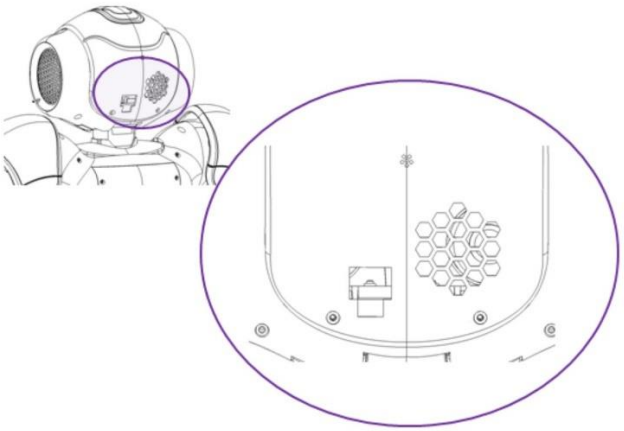
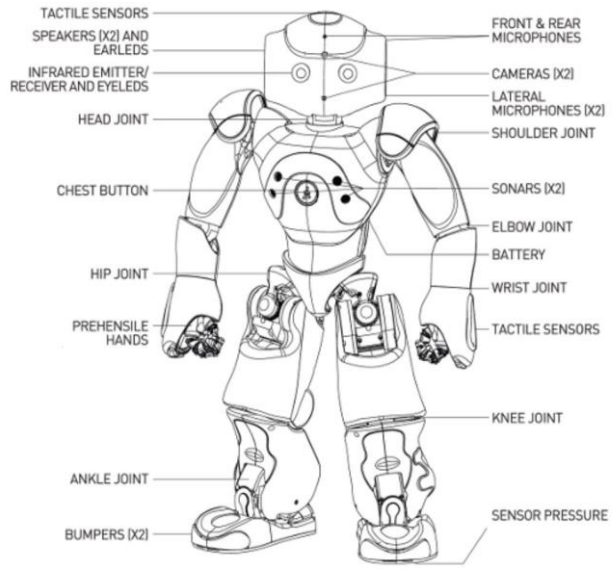


Figure-4: NAO Versions-2

NAO Body Types

NAO H25



NAO H21

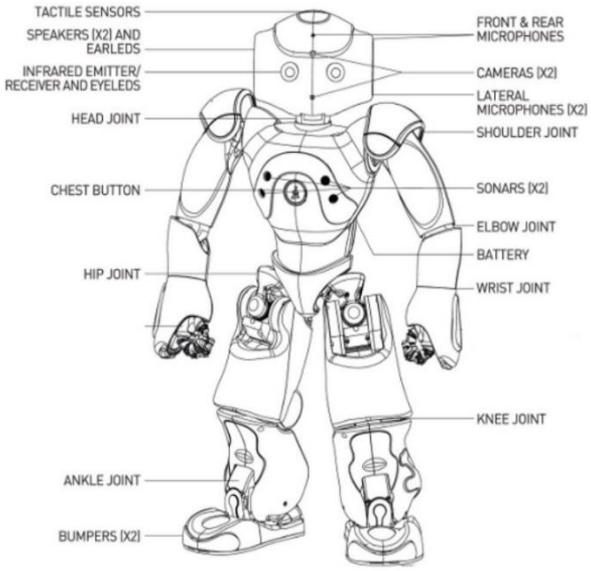
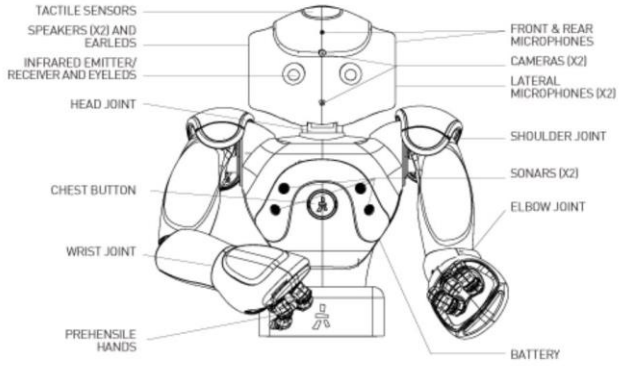


Figure-5: NAO Body Types-1

NAO T14



NAO T2

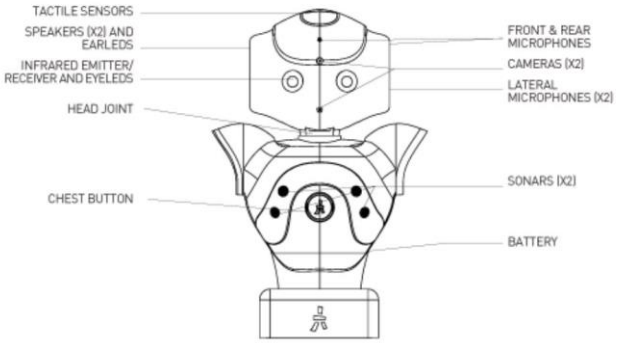


Figure-6: NAO Body Types-2

2.2 Getting Started with NAO

In this report, the discussion and calculations will be based on NAO H25 and V4.0, since the calculations and simulations are performed with respect to that robot type and version.

2.2.1 Software

Nao robots offer a vast array of functions and capabilities that can be utilized as per user's will. In order to organize the sensors and hardware, the NAO comes with embedded software and desktop software. Embedded software is the program running on the motherboard of the NAOs

and allows autonomous behaviors. Desktop software is the one written by the user, allows new behavior to run on robot remotely. In order to run autonomous behavior on NAO, Aldebaran has developed custom operating system for the robot known as OpenNAO. It is a linux based operating system and as such retains linux platform for navigation and running code. Running on top of OpenNAO there is NAOqi, the main development software for the robots. All the modules and behavior that come with NAO is built upon using tools provided by the NAOqi8.

Aside from autonomous behavior available natively through a new NAO, there are ways to program them through different platforms on computers. NAO have vast array of sensors and capabilities which can be utilized by a programmer. In order to make use of these functionalities, Aldebaran offers a few robust platforms. Some of the platforms utilized in this project are:

i. Choregraphe: platform specialized for NAO.
ii. Labview: Visual programming platform.
iii. C++ Software Development Kit (SDK): Written programming platform.

These platforms provide an interface with NAO, making it simpler to control the robots. However, they each have their specialty and limitation and as a result are suited for different tasks and users9. This section will cover what each of the above platforms brings to the table for programming NAOs.

Embedded Software

1. OpenNAO is the Operating Software of the Robot. It is the embedded GNU/Linux distribution based on Gentoo, specifically developed to fit the robot needs.
2. NAOqi is the main software that runs on the robot. Creating behaviors for the robot means calling modules and methods advertised by NAOqi.

Desktop Software

Choregraphe is the visual programming language. It allows you to create animations and behaviors, test them on a simulated robot before trying them with your real robot, and also monitor and control NAO.

2.2.2 Hardware

In total, there are 25 motors controlling the movement of the NAO. All movement is controlled by these motors. Three different types of motors are utilized in the NAO robots. Type 1 is used in the legs, type 2 in the hands, and the third type is used in the arms and head3. The table below

shows the specifications of each motor type, from the data sheet provided by Aldebaran Robotics:

	Motor Type 1	Motor Type 2	Motor Type 3
Model	22NT82213P	17N88208E	16GT83210E
No Load Speed	8300 +- 10%	8400 rpm +- 12%	10700 +- 10%
Stall Torque	68mNm +- 8%	94mNm+-8%	143mNm+-8%
Nominal Torque	16.1mNm	4.9mNm	6.2mNm

Table-1: Motor Types

These motors control each of the joints on the robots. There are two types of speed reduction ratios for each motor type, shown in the table provided by the Aldebaran Robotics data sheet:

	Motor Type 1	Motor Type 2	Motor Type 3
Type A	201.3	50.61	150.27
Type B	130.85	36.24	173.22

Table-2: Motor Types for Joints

The joints are shown in the picture below. The motors can all be controlled using the motion widget in Choregraphe, and the values of each motor will be shown. The picture on the right shows the motion widget, with the boxes used to edit the position of each motor on the right hand.

3. APPROACH

Initially, it was planned to do the simulation in Gazebo or Rviz. The requirements to run that setup requires the following-

1. Ubuntu 16.04
2. ROS Kinetic 14.04
3. Gazebo
4. Rviz
5. NAOqi C++ SDK

The nao_extras, nao_ packages were cloned from GitHub. However, those ROS packages were not as user friendly as expected which made it more difficult to simulate them as well read the angular orientation

I then resorted to Choregraphe which is conventionally used for controlling virtual as well as real NAO robots.

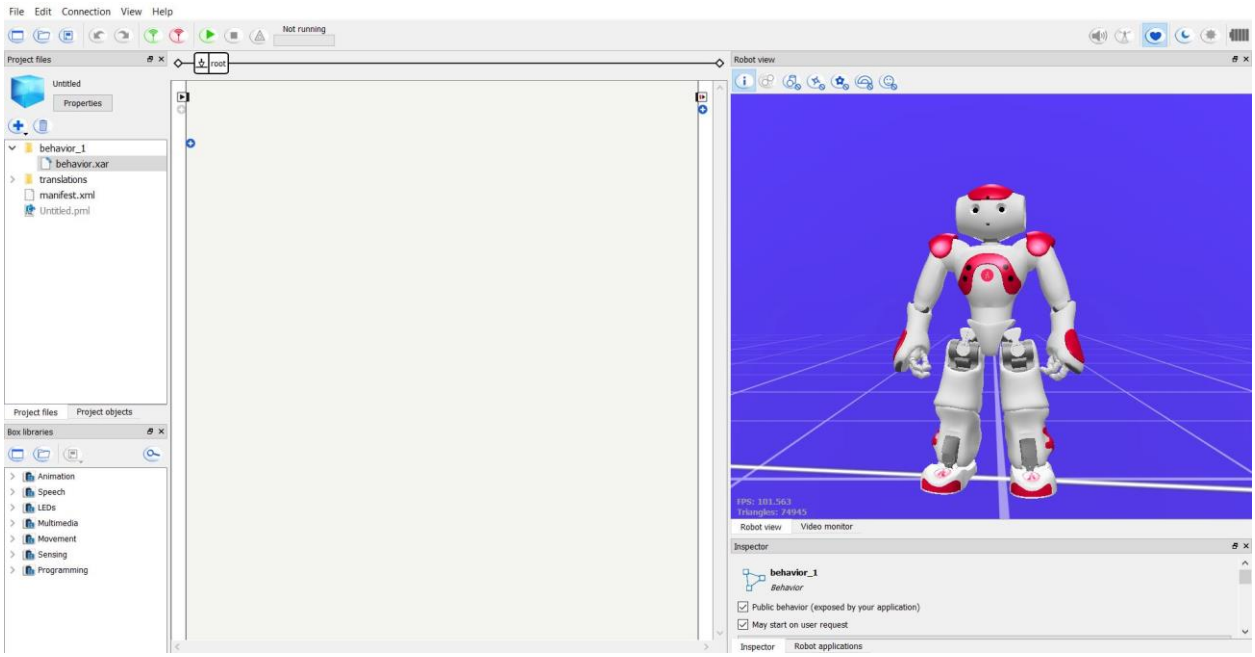


Figure-7: GUI of Choregraphe

And to facilitate my validation more, I showed the simulation in real-time in Webots where the NAO robot can kick the football.

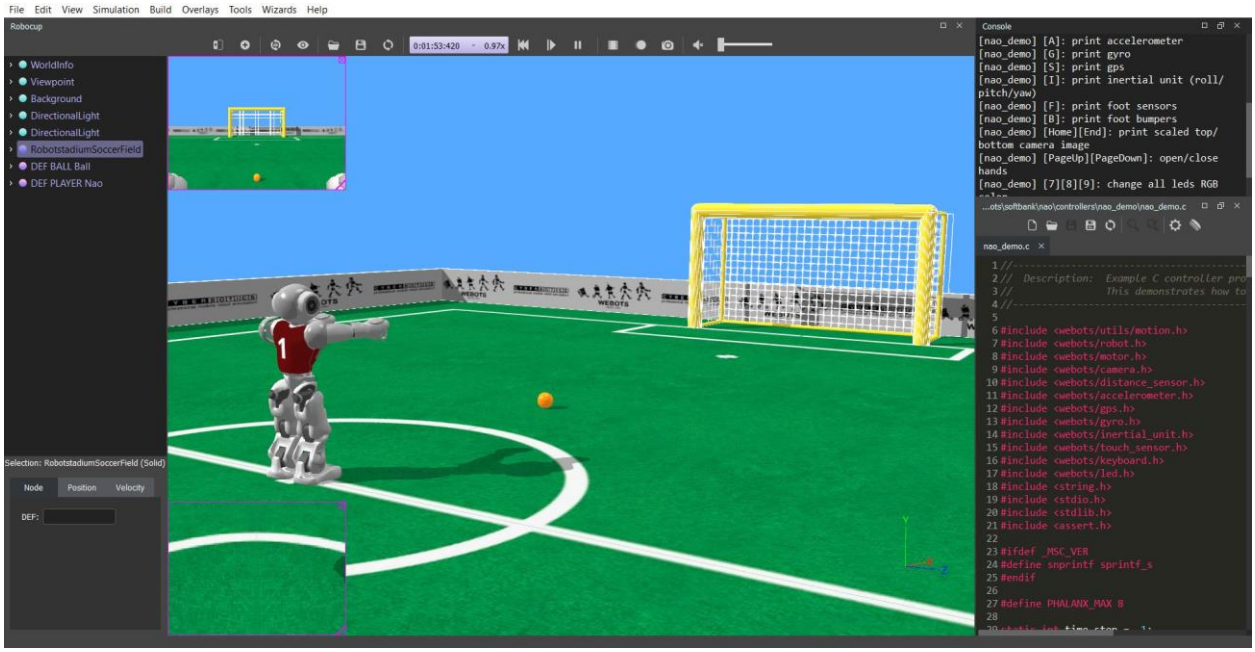


Figure-8: GUI of Webots

4. 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.

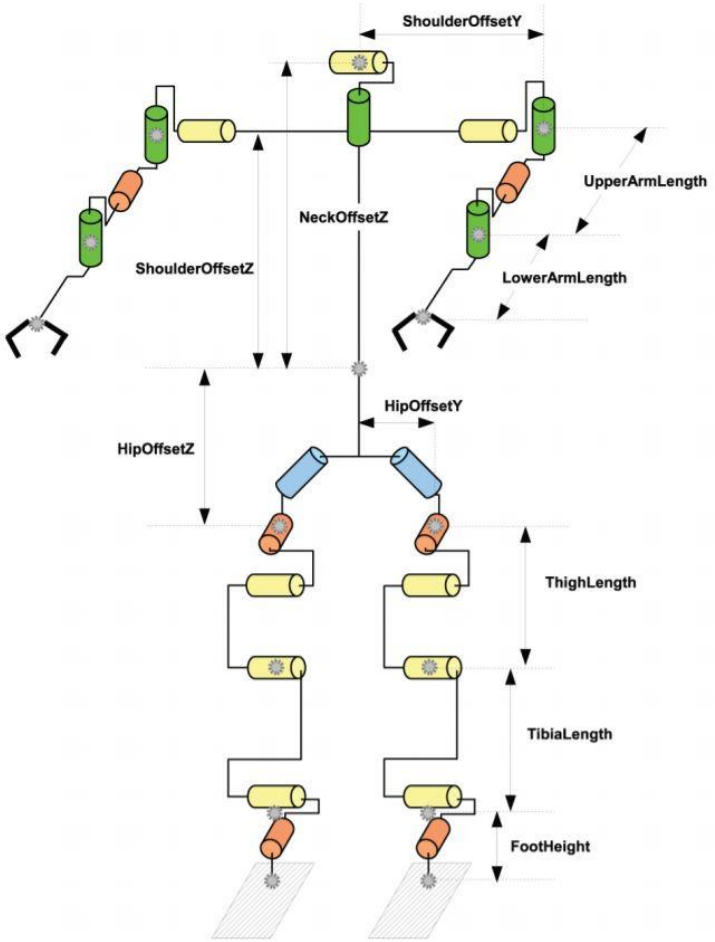


Figure- 9: Detailed Kinematics of NAO

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.

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.

4.1 AFFINE TRANSFORMATION

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, only the rotation and translation is being used, so the focus would only be on these two types of affine transformation. Additionally, the work is based on a three-dimensional Cartesian work space and therefore all the definitions and examples from now on will focus on this space.

An affine transformation matrix is a ((n + 1) x (n +1)) 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

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

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

$$v' = \begin{bmatrix} p'_1 \\ p'_2 \\ \vdots \\ p'_n \\ 1 \end{bmatrix} = Tv = \begin{bmatrix} X & Y \\ [0 \dots 0] & 1 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \\ 1 \end{bmatrix}$$

Translation Matrix

The translation in a cartesian space is a function that moves (translates) every point by a fixed distance in a specified direction. The translation can be described in the three-dimensional space with a (4 x 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). 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 x n) orthogonal matrix R with determinant 1:

$$R^T = R^{-1}, \qquad RR^T = R^TR = I, \qquad det(R) = 1$$

In the three-dimensional Cartesian space there are three distinct rotation matrices, each one of them performing a rotation of θ_x, θ_y, θ_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 must multiply with the corresponding rotation matrices in the following order:

$$p' = R_y R_x p$$

Also, 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:

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

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, α, d, and θ. Before explaining these parameters, first, the reference frame of each joint i with respect to the reference frame of its previous joint has to be established:

- 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, the DH parameters can be described as

- a: length of the common normal

- α: 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
- θ: angle about the z_{i-1} axis, from x_{i-1} axis to x_i axis

It is possible to move from the base reference frame of some joint to the transformed reference frame of this joint using the transformation matrix TDH, 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:

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

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

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 this project is based 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, the length of all the links of the robot is provided, the operational range in radians and degrees of the head joints, the arm joints, and the leg joints, as well as the mass of each joint/link. 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; it is assumed 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.

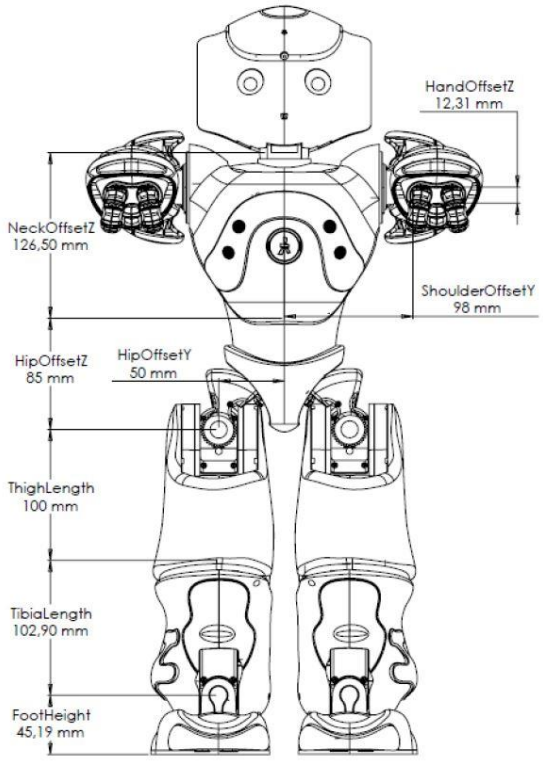


Figure-10: NAO Links

Name	Size(mm)
NeckOffsetZ	126.50
ShoulderOffsetY	98.00
ElbowOffsetY	15.00
UpperArmLength	105.00
LowerArmLength	55.95
ShoulderOffsetZ	100
HandOffsetY	57.75
HipOffsetZ	85.00
HipOffsetY	50.00
ThighLength	100.00
TibiaLength	102.90
FootHeight	45.19
HandOffsetZ	12.31

Table-3: Joint Sizes

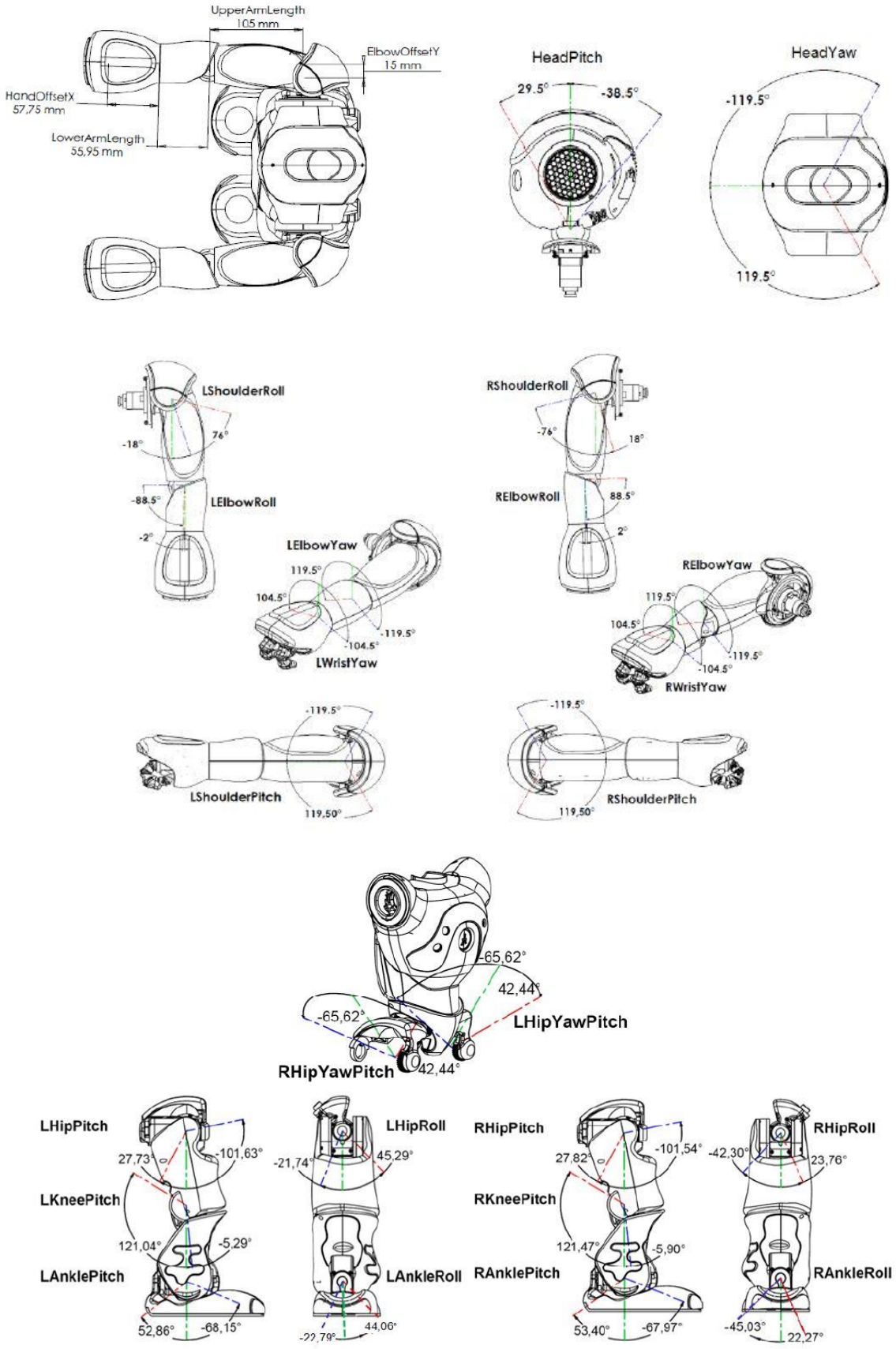


Figure- 11: Joints and Operational Range

Angles of Joints		
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
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

Angles of Joints		
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

Table- 4: Angles of Joints

Masses and CoM				
Frame Name	**Mass (kg)**	**CoM_x (mm)**	**CoM_y (mm)**	**CoM_z (mm)**
Torso	1.03947	-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**			

Table-5: Masses and CoM

4.2 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 (θ_1, θ_2, …. θ_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.

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. 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 it will be seen later, solution to the inverse kinematics problem would be intractable without solving the forward kinematics problem first.

NAO Zero Position

The base frame of the robot and the zero position of the joints must be defined before proceeding further. 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 it can be seen, in this position the Shoulder Roll joints are not really roll joints, but are yaw joints, so it can be understood that the names of the joints do not necessarily describe the actual movement of the joint in the base frame given the zero position.

Notation

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. Also, the notations are 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 $R_z(\theta_z)$.

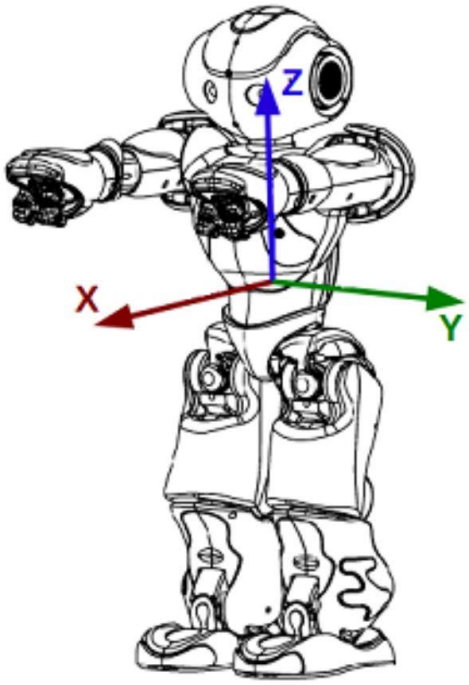


Figure-12: Orientation of the Robot w.r.t. Cartesian Coordinate Frame

The DH parameters of each kinematic chain is presented in a separate table and along with the translations from the "Base" to the first joint and from the last joint to the "End". Finally, some necessary rotations are provided 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 this case, the end effector is the point of interest. Forward kinematics are defined in terms of transformation, rotation, and translation matrices, and the 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. The (p_x; p_y; p_z) position and the (a_x; a_y; a_z) orientations of the final point are to be extracted. The position (p_x; p_y; p_z) can be simply read off the translation part of the transformation matrix:

$$p_x = T_{(1,4)}$$
$$p_y = T_{(2,4)}$$
$$p_z = T_{(3,4)}$$

The rotation of the final transformation table is a $R_zR_yR_x$ rotation table. Now it's easy to extract the orientation (ax; ay; az):

$$a_x = \arctan2(T_{(3.2)}, T_{(3,3)})$$
$$a_y = \arctan2(-T_{(3,1)}, \sqrt{T_{(3,2)}^2 + T_{(3,3)}^2})$$
$$a_z = \arctan2(T_{(2,1)}, T_{(1,1)})$$

Frame (Joint)	a	α	d	θ
Base	A(0,0,NeckOffsetZ)			
HeadYaw	0	0	0	θ_1
HeadPitch	0	$-\pi/2$	0	$\theta_2 - \pi/2$
Rotation	$R_x(\pi/2)R_y(\pi/2)$			
TopCamera	A(topCameraX, 0, topCameraZ)			
BottomCamera	A(bottomCameraX, 0, bottomCameraZ)			

Table-6: DH Parameters for Head

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, these matrices can be combined to find the point of the end effector in the frame space of the torso:

$$T_{Base}^{End} = A_{Base}^{0} T_0^1 T_1^2 R_x(\pi/2) R_y(\pi/2)\, A_2^{End}$$

T_0^1 *and* T_1^2 are the DH transformation matrices of the corresponding joints (HeadYaw, HeadPitch). A_2^{End} 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_{Base}^{End}.

Forward Kinematics for the Left Arm

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

Frame (Joint)	a	α	d	θ
Base	A(0, ShoulderOffsetY, ShoulderOffsetZ)			
LShoulderPitch	0	0	0	θ_1
LShoulderRoll	0	$-\pi/2$	0	$\theta_2 - \pi/2$
LElbowYaw	ElbowOffsetY	$\pi/2$	UpperArmLength	θ_3
LElbowRoll	0	$-\pi/2$	0	θ_4
Rotation	$R_z(-\pi/2)$			
EndEffector	A(HandOffsetX + LowerArmLength, 0, 0)			

Table-7: DH Parameters for Left Arm

Next, the coordinate frame with the rotation axis of the third joint (LElbowYaw) need to be aligned. To do so, the rotation about the y-axis should be performed. The DH parameters do not directly encode a rotation about the y-axis, so first rotation should be about the z-axis and then about the x-axis to effectively realize a rotation about the y-axis.

Thus, $-\frac{\pi}{2}$ should be added to the angle θ_2 of the previous joint (to rotate about the z-axis) and then rotate about the x-axis by $-\frac{\pi}{2}$ (the α parameter of LElbowYaw). Also, the ElbowOffsetY must be moved towards the x_2 axis so the parameter a is ElbowOffsetY. Then, the movement is to be done 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) the rotation about the x-axis

by $-\frac{\pi}{2}$ is performed, thus the α parameter for LElbowRoll is $-\frac{\pi}{2}$, while d, a are 0. At the end, only a simple rotation to fix the orientation of the coordinate frame and a simple translation to reach the end effector is needed. The table shows the DH parameters for all the joints of the left arm chain along with the necessary translations and rotations. Now, the final transformation matrix can be calculated easily:

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

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

Frame (Joint)	a	α	d	θ
Base	A(0, -ShoulderOffsetY, ShoulderOffsetZ)			
LShoulderPitch	0	0	0	θ_1
LShoulderRoll	0	$-\pi/2$	0	$\theta_2 - \pi/2$
LElbowYaw	ElbowOffsetY	$\pi/2$	UpperArmLength	θ_3
LElbowRoll	0	$-\pi/2$	0	θ_4
Rotation	R_z (-π/2)			
EndEffector	A(HandOffsetX + LowerArmLength, 0, 0)			

Table-8: DH Parameters for Right Arm

the y-axis. Also, in this chain one extra rotation matrix after the final translation should be added, 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:

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

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:

Frame (Joint)	a	α	d	θ
Base	A(0, HipOffsetY, -HipOffsetZ)			
LHipYawPitch	0	$-3\pi/4$	0	$\theta_1 - \pi/2$
LHipRoll	0	$-\pi/2$	0	$\theta_2 + \pi/4$
LHipPitch	0	$\pi/2$	0	θ_3
LKneePitch	-ThighLength	0	0	θ_4
LAnklePitch	-TibiaLength	0	0	θ_5
LAnkleRoll	0	$-\pi/2$	0	θ_6
Rotation	$R_z(\pi)R_y(-\pi/2)$			
EndEffector	A(0, 0, -FootHeight)			

Table-9: DH Parameters for Left Leg

$$T_{Base}^{End} = A_{Base}^{0} T_0^1 T_1^2 T_2^3 T_3^4 T_4^5 T_5^6 \; R_z(\pi) R_y(-\pi/2) A_6^{End}$$

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:

Frame (Joint)	a	α	d	θ
Base	A(0, HipOffsetY, -HipOffsetZ)			
RHipYawPitch	0	$-\pi/4$	0	$\theta_1 - \pi/2$
RHipRoll	0	$-\pi/2$	0	$\theta_2 - \pi/4$
RHipPitch	0	$\pi/2$	0	θ_3
RKneePitch	-ThighLength	0	0	θ_4
RAnklePitch	-TibiaLength	0	0	θ_5
RAnkleRoll	0	$-\pi/2$	0	θ_6
Rotation	$R_z(\pi)R_y(-\pi/2)$			
EndEffector	A(0, 0, -FootHeight)			

Table-10: DH Parameters for Right Leg

$$T_{Base}^{End} = A_{Base}^{0} T_0^1 T_1^2 T_2^3 T_3^4 T_4^5 T_5^6 \; R_z(\pi) R_y(-\pi/2) A_6^{End}$$

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, the solution for this reverse problem can be obtained 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

$$T_{Base}^{End} = (T_{Base}^{End})^{-1}$$

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, the torso relative to the left foot frame is located. Then, it is just multiplied 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.

$$T_{LFoot}^{Head} = (T_{Torso}^{LFoot})^{-1} T_{Torso}^{Head}$$

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

4.3 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 (θ_1, θ_2, …. θ_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.

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, it is aimed 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.

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 _ 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 exibility 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 only rotation and translation is used.

Forward kinematics can find the point of an end effector, relatively to the start frame, given the values of the joints. Now, the attention can be turned 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.

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 $R_zR_yR_x$. Thus, the complete transformation matrix T can be constructed from the base frame to the point of the end effector mentioned above:

$$R' = \begin{bmatrix} cosa_y cosa_z & -cosa_x sina_z + sina_x sina_y cosa_z & sina_x sina_z + cosa_x sina_y cosa_z & p_x \\ cosa_y sina_z & cosa_x cosa_z + sina_x sina_y sina_z & -sina_x cosa_z + cosa_x sina_y sina_z & p_y \\ -sina_y & sina_x cosa_y & cosa_x cosa_y & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Given T, the 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 that must be solved to find the values of the joints are formed by writing down the forward kinematics transformation matrix symbolically with the _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 θ_i of the n joints of the chain as unknowns. In fact, a total of 2n unknowns will be present, because all the θ_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 $\cos\theta_i$.

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

1. Transforming matrix to the target point
2. Transforming matrix through the chain
3. Forming a non-linear system by equating the two matrices
4. Manipulating both sides to make the problem easier
5. Finding values for some joints through geometry and trigonometry
6. Finding values for the remaining joints from the non-linear system
7. Validating all candidate solutions through forward kinematics

Inverse Kinematics for the Head

The head chain consists of only two joints, therefore the work should be done 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, the desired target orientation can be achieved simply by setting the HeadYaw and HeadPitch joints to a_z and a_y respectively. In the former case, the symbolic matrix from forward kinematics is constructed 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 θ for the second joint, l_1 = cameraX, l_2 = cameraZ, and l_3 = NeckOffsetZ. Since only the position (p_x, p_y, p_z) is known, the rotation block of the matrix cannot be reconstrute, so the focus is on the translation block. From the symbolic matrix-

$$\boldsymbol{T_{(3,4)} = l_3 - l_2 sin\widehat{\theta_2} - l_1 cos\widehat{\theta_2} = p_z}$$

It is known from trigonometry that-

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

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

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 θ_2 is $\theta_2 = \widehat{\theta_2} + \frac{\pi}{2}$. Substituting $\theta_2 - \frac{\pi}{2}$ in $\widehat{\theta_2}$. Now, $\theta_1$can be easily extracted from $T_{(1,4)}$

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

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

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

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

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

Provided a target position (p_x, p_y, p_z) or

$$\theta_1 = a_x$$
$$\theta_1 = a_x$$

Provided a target orientation (a_x, a_y, a_z)

Inverse Kinematics for the Left Arm

First, the symbolic transformation matrix for the left arm chain, then from the chain, the Base and End transformation are removed together with the End rotation.

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

$$\boldsymbol{T' = (A_{Base}^0)^{-1} T (A_4^{End})^{-1} (R_z(\tfrac{\pi}{2}))^{-1}}$$

Then, T' is inverted

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

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

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

$$r_{12} = cos\theta_3 cos\theta_4 sin\widehat{\theta_2} + cos\widehat{\theta_2} sin\theta_4$$

$$r_{13} = -cos\widehat{\theta_2} cos\theta_4 sin\theta_1 cos\theta_3 + cos\theta_1 cos\theta_4 sin\theta_3 + sin\theta_1 sin\widehat{\theta_2} sin\theta_4)$$

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

$$r_{21} = -sin\theta_1 sin\theta_3 sin\theta_4 - cos\theta_1(sin\widehat{\theta_2} cos\theta_4 + cos\widehat{\theta_2} cos\theta_3 sin\theta_4)$$

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

$$r_{23} = cos\theta_4 sin\theta_1 sin\widehat{\theta_2} + (cos\widehat{\theta_2} cos\theta_3 sin\theta_1 - cos\theta_1 sin\theta_3) sin\theta_4$$

$$r_{24} = l_2 cos\theta_4 + l_1 cos\theta_3 sin\theta_4$$

$$r_{31} = cos\theta_3 sin\theta_1 - cos\theta_1 cos\widehat{\theta_2} sin\theta_3$$

$$r_{32} = -sin\widehat{\theta_2} sin\theta_3$$

$$r_{33} = cos\theta_{41} cos\theta_3 + cos\widehat{\theta_2} sin\theta_1 sin\theta_3$$

$$r_{34} = l_1 sin\theta_3$$

where, $\widehat{\theta_2}$ is the DH Parameter θ for the second (LShoulderRoll) joint, l_1 = ElbowOffsetY, l_2 = UpperArmLength.

The θ_3 from the resulted transformation matrix is extracted:

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

The next angle that can be found is θ_4, so the focus is on equations from the symbolic matrix, where only θ_4 and/or θ_4 (now θ_4 is a known variable) are present. Using $T''_{(1,4)}$

$$T''_{(1,4)} = -l_1 cos\theta_3 cos\theta_4 + l_2 cos\theta_4$$

$$sin\theta_4 = \frac{T''_{(1,4)} + l_2 cos\theta_3 cos\theta_4}{l_3}$$

Now, the focus is on T_{24}

$$T''_{24} = l_2 cos\theta_4 + l_1 cos\theta_3 sin\theta_4$$

$$T''_{24} = l_2 cos\theta_4 + l_1 cos\theta_3 \frac{T''_{(1,4)} + l_2 cos\theta_3 cos\theta_4}{l_3}$$

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

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

Next, the chain is manipulated again. Now, bot DH Transformations for θ_3 *and* θ_4 can be removed from the chain so

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

Now, the final two joint values can be easily extracted

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

$$\theta_4 = \widehat{\theta_2} - \frac{\pi}{2}$$

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

At this point, there can be multiple solution sets for all the joints of the left arm, but some of them may be invalid. The correct set(s) through a forward kinematics validation step can be found.

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

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

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 = −ElbowOffsetY.

It is clear that, the chain for the right arm is the same as the chain for the left arm. As it can be seen, 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:

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

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

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

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

$$T''' = T'(T^3_2)^{-1}(T^4_3)^{-1}$$

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

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

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 to simplify it, the known translations from the kinematic chain can be removed-

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

Now, there exists 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. The origin of the chain is rotated by $\frac{\pi}{4}$ about the x-axis to make the first joint (LHipYawPitch) a yaw joint (aligned with the z-axis):

$$\tilde{T} = R_x(\frac{\pi}{4})\, \hat{T}$$

Now, the end effector is the LAnkleRoll joint and the base is the rotated LHipYawPitch joint. The first four joints (LHipYawPitch, LHipRoll, LHipPitch, LKneePitch) a_ect 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 it is operated in the three-dimensional space. Thus, the transformation matrix is inverted to form the reverse chain. Now only the LAnkleRoll, LAnklePitch, and LKneePitch joints affect the position:

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

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

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

where, l_1 = ThighLength and l_2= TibiaLength. The θ_4 can now be found the same way θ_4 was found for the arms. The focus is now 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-

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

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, the law of cosines can be used to find the interior angle θ'_4 between the thigh and tibia sides of the triangle.

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

So, θ'_4 represents an interior angle, whereas the LKneePitch joint is stretched in the zero position, the resulting angle θ''_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, the θ_4 angle can be finally extracted as

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

Next, the θ_6 angle can be extracted from the translation block using r_{24} and r_{34}

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

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

$$\theta_4 = \arctan\left(\frac{p'_y}{p'_z}\right)$$

Since θ_5 is not known, it cannot be determined 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 (θ_4) and AnklePitch (θ_5) joints; in these con_gurations, which represent only a fraction of extreme values in the joint space, no unique solution can be extracted for θ_6. Essentially, in these con_gurations, 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 (θ_6) has no effect on its position; an infinity of solutions exists, given that for any choice of θ_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 this implementation simply returns a result of 0. The solution θ_6 = 0 can be adopted, and the problem of the target orientation to the hip joints is relayed; invalid solutions, if any, will be rejected by the validation step.

To move on, go back to $\tilde{T}$ and the two rotations at the end of the chain along with the transformation T_5^6 which is now known is to be removed, because θ_6 is known

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

Like before, the reverse chain is formed as:

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

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

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

In these equations, θ_5 is the only unknown. From r_{14}, an expression for $\cos\theta_5$ can be obtained

$$r_{q4} = p''_x$$

$$(l_2 + l_1 cos\theta_4)\cos\theta_5 = p''_x + l_1 sin\theta_5 \sin\theta_4$$
$$\cos\theta_5 = \frac{p''_x + l_1 sin\theta_5 \sin\theta_4}{l_2 + l_1 cos\theta_4}$$

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

$$sin\theta_4(-l_2 - l_1 cos\theta_4) - l_1 cos\theta_5 cos\theta_4 = p''_y$$

$$sin\theta_5(-l_2 - l_1 cos\theta_4) - l_1 \frac{p''_x + l_1 sin\theta_5 sin\theta_4}{l_2 + l_1 cos\theta_4} sin\theta_4 = p''_y$$

$$-sin\theta_5(l_2 + l_1cos\theta_4) - \frac{l_1 p''_x sin\theta_4}{l_2 + l_1cos\theta_4} - \frac{l^2{}_1 sin\theta_5 sin^2\theta_4}{l_2 + l_1cos\theta_4} = p''_y$$

$$-sin\theta_5(l_2 + l_1cos\theta_4)^2 - l^2{}_1 sin\theta_5 sin^2\theta_4 = p''_y\,(l_2 + l_1cos\theta_4) + l_1 p''_x sin\theta_4$$

$$\theta_5 = \arcsin\left(\frac{p''_y\,(l_2 + l_1cos\theta_4) + l_1 p''_x sin\theta_4}{l^2{}_1 sin^2\theta_4 + (l_2 + l_1cos\theta_4)^2}\right)$$

The division is always feasible, because $l^2{}_1 sin^2\theta_4 + (l_2 + l_1cos\theta_4)^2$ is obviously greater than zero for any value of θ_4. Now, going back to T' and removing the two transformations T_3^4 and T_4^5 , since θ_4 and θ_4 are known:

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

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-

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

where $\widehat{\theta_1}$ is the DH Parameter θ for the first (LHipYawPitch) joint and $\widehat{\theta_2}$ is the DH parameter θ for the second (LHipRoll) joint. Now, the remaining three angle can be extracted as follows-

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

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

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

$$\theta_1 = \widehat{\theta_1} + \frac{\pi}{2}$$

The equations above are valid because by the robot construction the LHipRoll joint θ_1 cannot reach $-\frac{\pi}{4}$ or $\frac{3\pi}{4}$ and therefore, the denominator $\sin(\theta_2 + \frac{\pi}{4})$ never becomes zero.

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

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

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

$$\boldsymbol{\theta_6 = \arctan\left(\frac{T'_{(2,4)}}{T'_{(3,4)}}\right)} \quad \text{if } (l_2cos\theta_5 + l_1cos(\theta_4 + \theta_5)) \neq 0$$

$$\boldsymbol{\theta_6 = 0} \quad \text{if } (l_2cos\theta_5 + l_1cos(\theta_4 + \theta_5)) = 0$$

$$\boldsymbol{T'' = ((T')^{-1}\left(T^6_5R_z(\pi)R_y\left(-\frac{\pi}{2}\right)\right)^{-1})^{-1}}$$

$$\boldsymbol{\theta_5 = \arcsin\left(-\frac{T''_{(2,4)}(l_2 + l_1cos\theta_4) + l_1T''_{(1,4)}sin\theta_4}{l^2{}_1sin^2\theta_4 + (l_2 + l_1cos\theta_4)}\right)}$$

$$\boldsymbol{\theta_5 = \pi - \arcsin\left(-\frac{T''_{(2,4)}(l_2 + l_1cos\theta_4) + l_1T''_{(1,4)}sin\theta_4}{l^2{}_1sin^2\theta_4 + (l_2 + l_1cos\theta_4)}\right)}$$

$$\boldsymbol{T''' = (T'')^{-1}(T^4_3T^5_4)^{-1}}$$

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

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

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

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

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 $-\frac{\pi}{4}$.

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

$$\hat{T} = (A^0_{Base})^{-1} T (A^{End}_6)^{-1}$$

$$\tilde{T} = R_x(-\frac{\pi}{4})\, \hat{T}$$

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

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

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

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

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

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

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

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

$$\theta_5 = \arcsin\left(-\frac{T''_{(2,4)}\,(l_2 + l_1 cos\theta_4) + l_1 T''_{(1,4)} sin\theta_4}{l^2{}_1 sin^2\theta_4 + (l_2 + l_1 cos\theta_4)}\right)$$

$$\theta_5 = \pi - \arcsin\left(-\frac{T''_{(2,4)}\,(l_2 + l_1 cos\theta_4) + l_1 T''_{(1,4)} sin\theta_4}{l^2{}_1 sin^2\theta_4 + (l_2 + l_1 cos\theta_4)}\right)$$

$$T''' = (T'')^{-1}(T_3^4 T_4^5)^{-1}$$

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

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

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

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

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 now). Another choice is to set the joint to the mean of the two values. None of these choices are 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.

5. SIMULATION OF NAO

I performed the simulation of NAO in Choregraphe and Webots. The first one gave me an insight about the joint angles, while the second one made it clear visually as well as MATLAB code point of view to witness the simulation of NAO.

5.1 SIMULATION IN CHOREGRAPHE

Using the Aldebaran NAO supported software, the simulation was done in Choregraphe. This robot gives a facility to control the joint angles and move the robot as desired.

As shown in the below figure, the angles of the Right Leg denote individual joint angles that the user can change just to check the connectivity and basic functionality of the robot before going for an actual performance.

All the simulations that can be done here are merely controlled by changing the angles manually or by using pre-defined functions to command the virtual robot to perform some pre-planned actions.

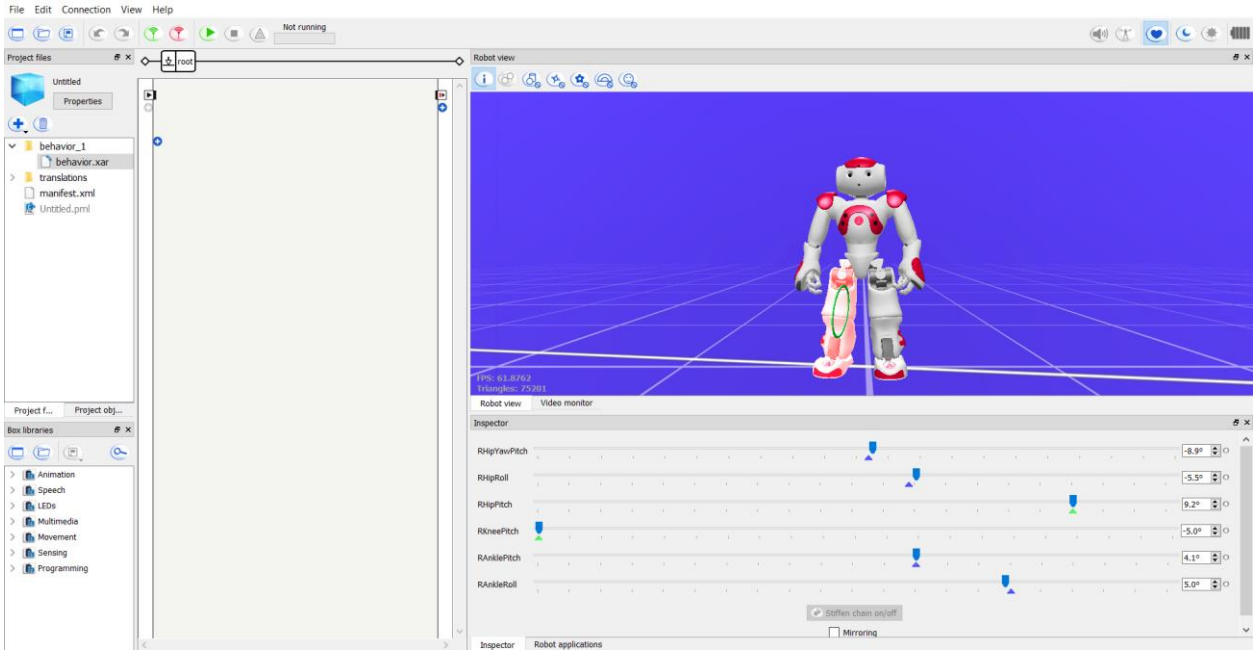


Figure-13: GUI of Choregraphe

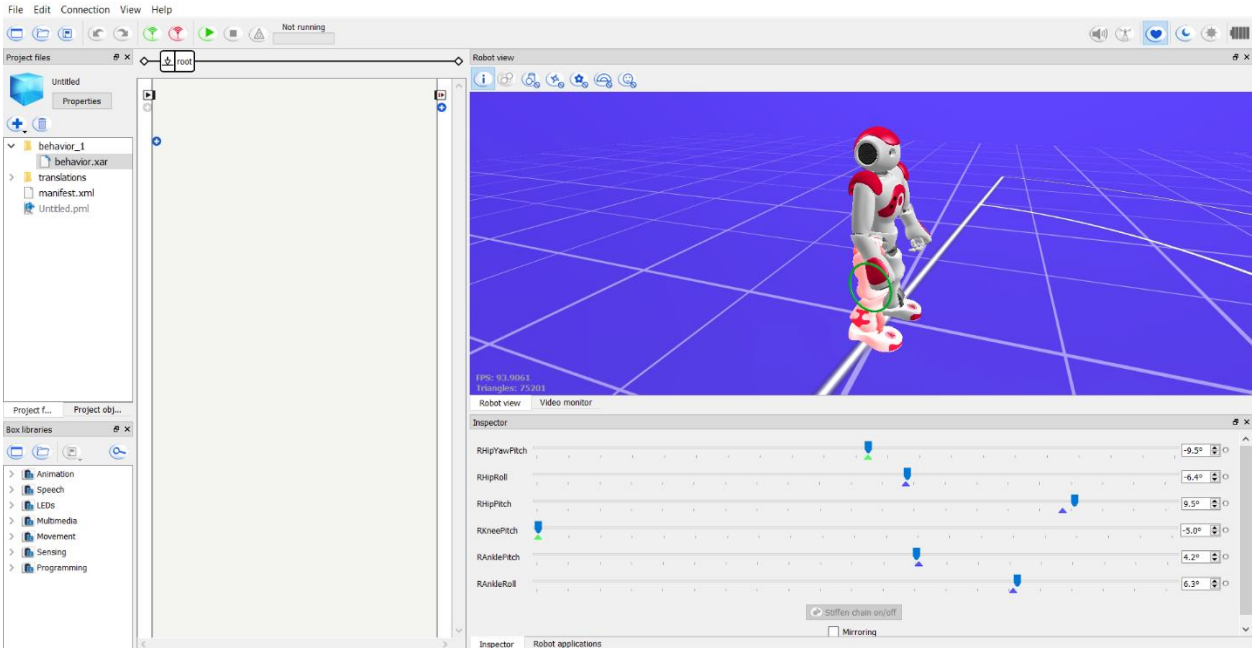


Figure-14: Joint Angles of Right Leg

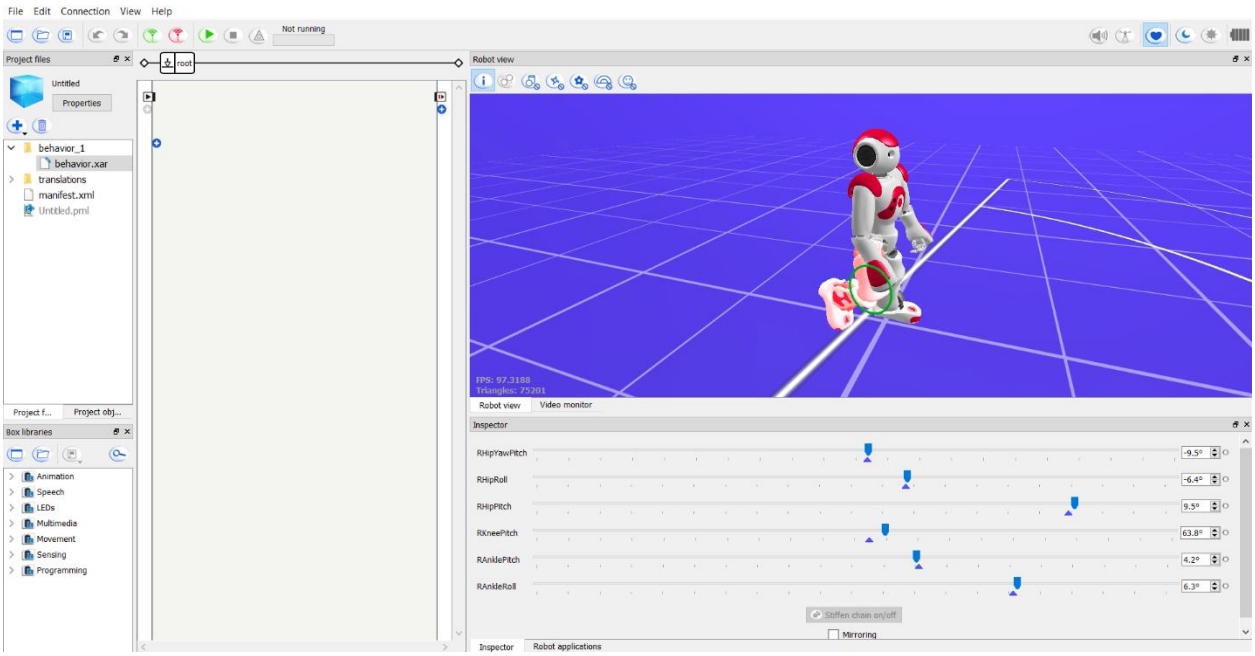


Figure-14: Right Leg Ready to Kick

As it is visible in the above image, upon changing the RKneePitch, the Right Leg was in the ready position to kick the ball.

5.2 SIMULATION IN WEBOTS

Using the MATLAB code, the kinematics of the Robot can be controlled that can be executed either by pressing individual keys or just by executing the code. Following are the screenshots of the simulation.

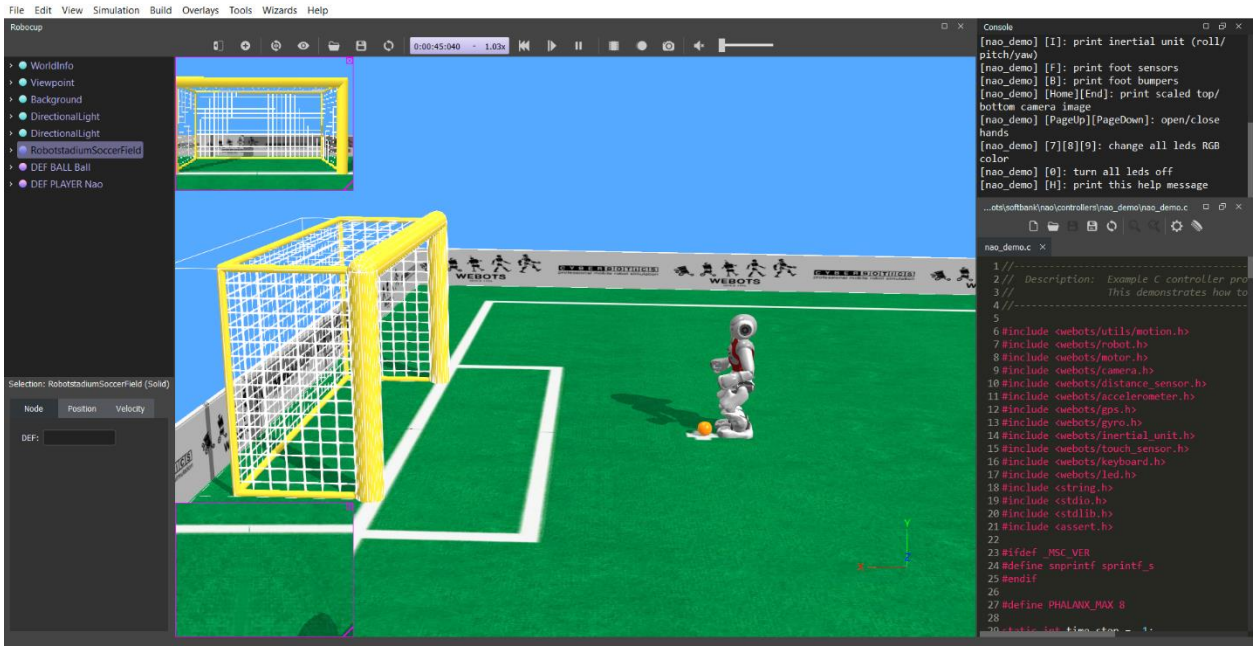


Figure-15: GUI of Webots

Figure-16: Robot ready to Kick

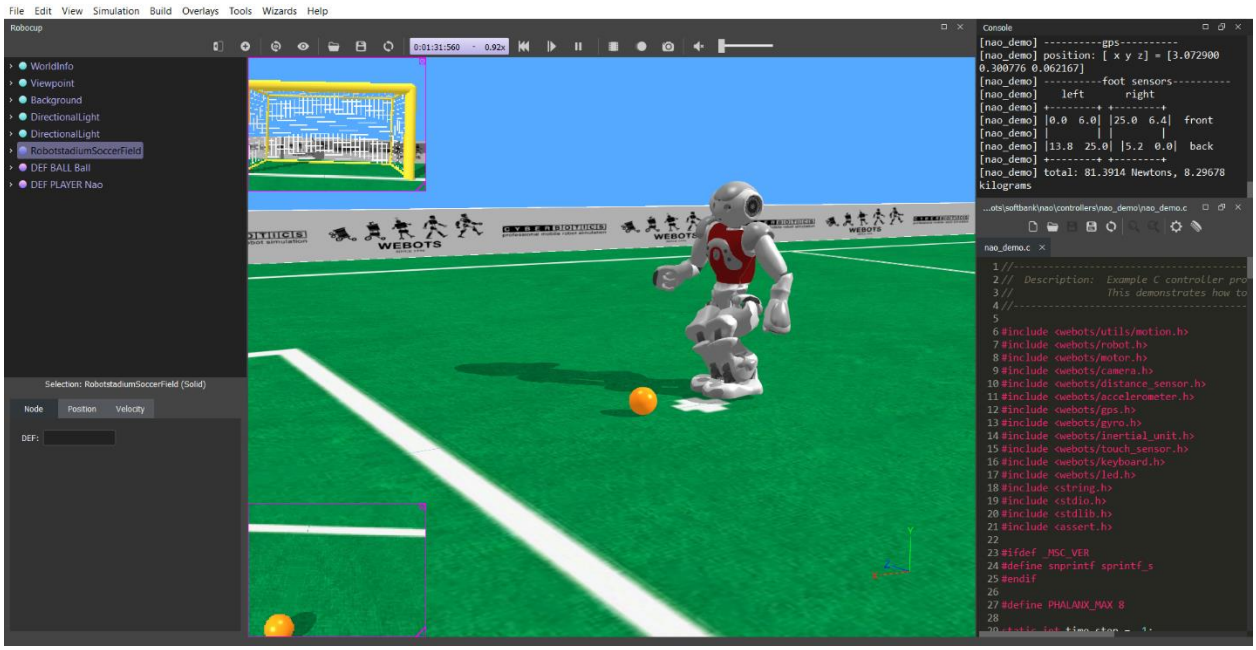


Figure-17: Robot after Kicking the Ball

The entire simulation of the video has been attached in the .zip file with submission

6. VALIDATION OF MODEL

Both for the forward and inverse kinematics validation, it was difficult to validate because of the complexity of the equation and the challenges in manually integrating the desired parameter values in to virtual robot.

An attempt was made to use the values of the ball and robot to calculate the inverse kinematics, but it wasn't possible to implement it.

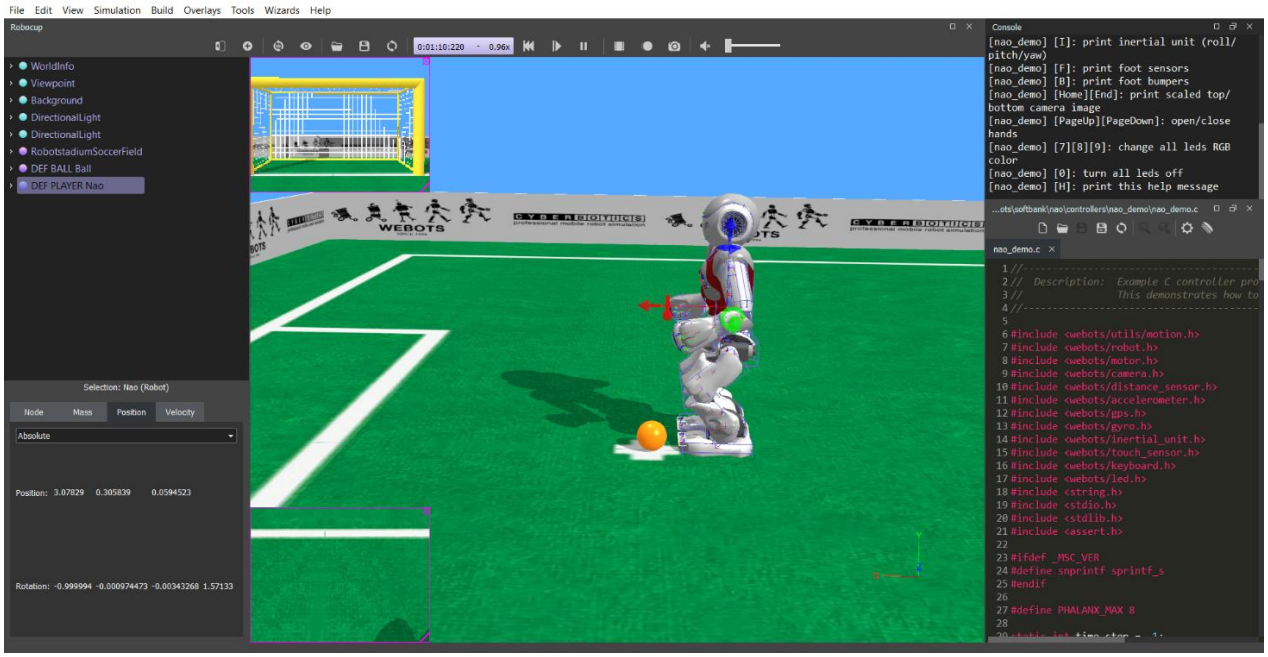


Figure-15: Joint Angles of Right Leg

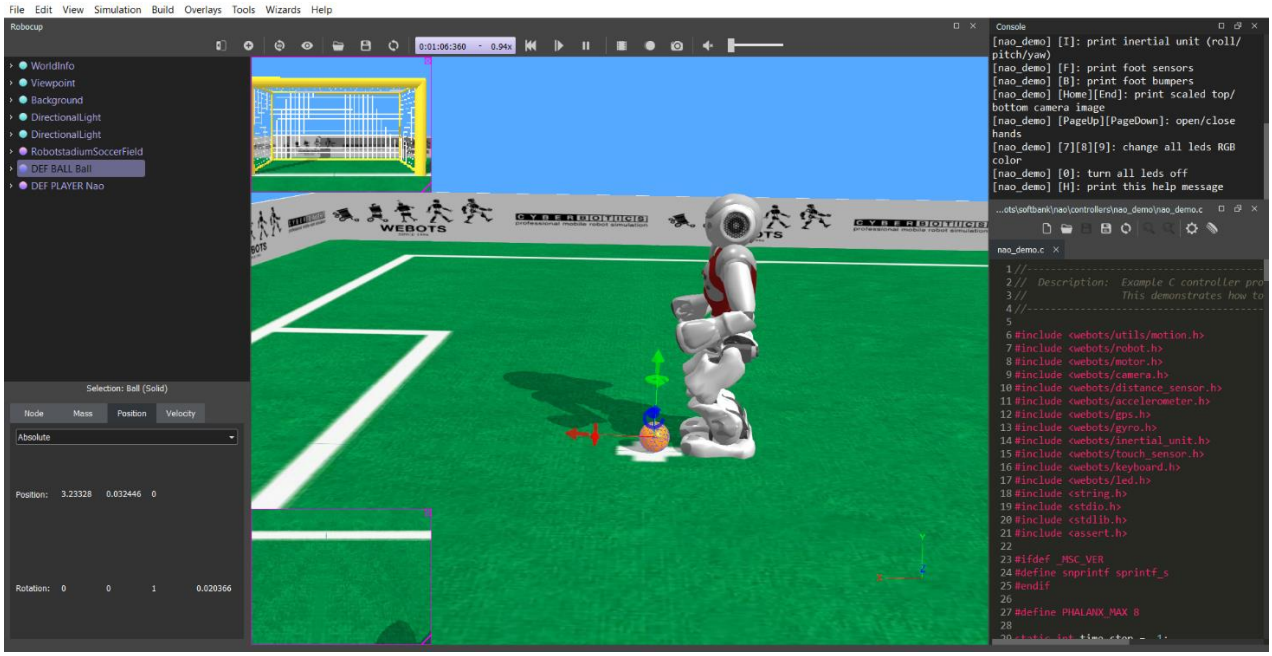


Figure-16: Joint Angles of Right Leg

Home Positions of Ball and Robot:

Ball: Position: (3.23439, 0.032446, 0),
Angles (in quaternion form): (0, 0, 1, 0.0167499)

Robot: Position: (3.06829, 0.305839, 0.0577693),
Angles (in quaternion form): (-0.999993, -0.00106701, -0.00352525, 1.57133)

Due to the difficulty of this project, as stated in the pre-proposal, it turned out impossible to execute the dance sequence and indicate the kinematics involved with all the links and joint and show them in real-time.

7. PROJECT LEARNING & FUTURE WORK

Project Learning

Learning from this modelling project helped me in learning the comprehensive implementation and of a robotics field. As my project was on the Kinematics of NAO Robot, I got an opportunity of implementing my class knowledge in a totally different world of NAO Robot having 25 DOF. I could learn the following things through this project-

A) Inverse position kinematics of a NAO robot
B) Forward position kinematics of a NAO robot
C) Using ROS, Gazebo and Rviz briefly (despite of a failed attempt at simulation)
D) Using the NAO simulation environment- Choregraphe and Webots
E) Simulating the NAO robot as desired
F) Using MATLAB for robotic system simulations

Future Work

Live Kinematics Validation

At present, due to some limitations, I couldn't perform the kinematics validation as per the requirement of the project. So, the future work would be to do a live validation of the model as per the kinematics calculated for it.

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.

8. CONCLUSION

My approach to NAO kinematics is based on standard principled methods for studying robot kinematic chains. The project was demanding and difficult to complete as it involved many unseen aspects. However, that resulted in the learning and derivations of Inverse position kinematics and Forward position kinematics of the NAO robot as the major and the primary outcome. But apart from that it also resulted in many sub outcomes which are also important for the robotics system implementation as explained in the learning section. The project outcomes were seen in the derivation results and the videos of simulations. The references used in the project were immensely useful and the project was inspired from the thesis on ‘Forward and Inverse Kinematics for the NAO Humanoid Robot’ by Nikolas Kofinas and an effort was made to represent his work on my own with the help of simulation

9. REFERENCES

1. Nikolas Kofinas, “Forward and Inverse Kinematics for the NAO Humanoid Robot”, Technical University of Crete, Greece
2. Alicia Bargar, Jillian Greczek, Dr. Maja Mataric, **“**A Comprehensive ROS Interface for the Aldebaran NAO”
3. Daniel P. Haggerty, Kshitij Ken Thapa, “Interface System for NAO Robots”, Worcester Polytechnic Institute
4. Nikos Kofinas, Emmanouil Orfanoudakis, and Michail G. Lagoudakis, “Complete Analytical Inverse Kinematics for NAO”
5. Nikolaos Kofinas, Emmanouil Orfanoudakis and Michail G. Lagoudakis, “Complete Analytical Forward and Inverse Kinematics for the NAO Humanoid Robot”
6. Arne Bockmann and Tim Laue, “Kick Motions for the NAO Robot using Dynamic Movement Primitives”
7. Shuhui Wen , Zhiyuan Ma , Shuhuan Wen, Yongsheng Zhao, Jiantao Yao, “The Study of NAO Robot Arm Based on Direct Kinematics by Using D-H Method”,
8. Felix Burget Armin Hornung Maren Bennewitz, “Whole-Body Motion Planning for Manipulation of Articulated Objects”
9. David Gouaillier, Vincent Hugel and Pierre Blazevic, “The NAO humanoid: A Combination of Performance and Affordability”
10. http://wiki.ros.org/nao
11. http://wiki.ros.org/nao/Tutorials