Dynamically Stable and Impact controlled Humanoid Kick

By

Saifullah 201200561

Supervised By

Dr. Yasar Ayaz

School of Mechanical and Manufacturing Engineering

National University of Sciences and Technology (NUST)

Islamabad, Pakistan

June, 2016

Dynamically Stable and Impact controlled Humanoid Kick

By

Saifullah 201200561

Supervised By

Dr. Yasar Ayaz

A thesis submitted in partial fulfillment of the requirements for the degree of Bachelors of Engineering in Mechanical Engineering

School of Mechanical and Manufacturing Engineering

National University of Sciences and Technology (NUST)

Islamabad, Pakistan

National University of Sciences & Technology

FINAL YEAR PROJECT REPORT

We hereby recommend that the dissertation prepared under our supervision by: Saifullah NUST01200561 Titled: "Dynamically Stable and Impact controlled Humanoid Kick" be accepted in partial fulfillment of the requirements for the award of Bachelors of Engineering in Mechanical Engineering degree with (_____ grade)

English and format checked by Ms Aamna Hassan,

Signature: _____

Guidance Committee Members

1. Name:_____ Signature:_____

2. Name:_____ Signature:_____

3. Name:_____ Signature:_____

Supervisor's Name:_____ Signature:_____

Date:_____

_____ _____

Head of Department Date

COUNTERSIGNED

Date:_____ _____

Dean/Principal

Declaration

I declare that this project report entitled “Dynamically Stable and Impact controlled Humanoid Kick”, submitted as a requirement for the award of BE (ME) degree, does not contain any material previously submitted for a degree in any university; and that to the best of my knowledge it does not contain any material previously published or written by another person except where due reference is made in the text.

_____

Saifullah

NUST201200561

Copyright Statement

Acknowledgement

I am very thankful to my Almighty Allah for giving me a lot of courage to choose such an immense project and for completing it with great dignity and grace. Lots of people have played a significant role in the completion of this project.

I am extremely grateful to Mr. Yasar Ayaz and Mr. Fahad Islam for their valuable suggestions and their support to perform such a valuable research and implementation work and study in the field of Robotics and Artificial Intelligence. I truly admire their dedication to my project and for all the help and guidance that they have offered. A great deal of our project has been possible because of both of them.

I am very thankful to all of my teachers who had been guiding me throughout my project work. Their knowledge, guidance and training enables me to carry out this research work more efficiently.

I would like to thank all my friends, colleagues for fiving me fruitful suggestions about increasing the efficiency of my project. They are too many in number that I cannot thank implicitly. God bless them! Ameen.

Saifullah

Abstract

Research in the field of Robotics and Artificial Intelligence has become very popular, specifically in the area of multi-degree of freedom articulated robots such as humanoids. These robots are mostly preferred due to their capability to perform human-like complicated tasks. There are many complex humanoid dynamic motions that are required in RoboCup robot soccer competition such as walking, kicking and static and dynamic balancing, but to execute these tasks, all the environment parameters have to be modelled and accounted for during the planning phase. This thesis studies the problems of kick motion generation for the Aldebaran NAO humanoid robot and for the first time includes the model for dynamic impact control. The proposed model is made possible by using the principals of effective mass and applying the momentum conservation theories on the impact. Furthermore, to provide the stability or balance to the robot during the motion, zero moment point (ZMP) stability criteria is used with an optimal controller designed for its implementation. The thesis has been successfully implemented to create robust and strong kick motions on a real robot and verified through various experiments. The resulting product is currently being used as a kick engine for the team-NUST competing in Standard Platform League (SPL) of the RoboCup 2016 competition.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

With the recent advancements in the field of robotics and artificial intelligence, research in the area of multi-degree of freedom articulated robots, such as humanoids, has gained immense popularity. Humanoids are generally getting more attention because of their capabilities to perform the intricate human-like motions that cannot be performed by other kinds of robots such as walking up the stairs, opening a valve, kicking a ball, etc. To make this research more appealing, an organization called “RoboCup Federation” has introduced a robot soccer competition called “RoboCup”, which is held every year and welcomes all to demonstrate the newest research advancements in the field of humanoids in the form of a competition. The competition organizes various leagues with different rules and environments.

This thesis is focused on the Standard Platform League (SPL), in which all the teams compete against each other using the standard Aldebaran NAO humanoid robot. NAO is a mid-sized humanoid which shows a wide range of capabilities in terms of its task execution. For the purpose of competing in a SPL, every team has to create an autonomous humanoid team to play a soccer match in which different skills such as walking, kicking, dribbling, etc. are required. Almost every team has designed their own motion generation

engines for these complex tasks, and since our team from NUST is taking part in this year's competition, there was a need to design a kick engine for our team.

1.2 Aims and Objectives

The aim of this research is to design a new kick engine for "team-NUST" for which the kicks used before were based on simple predefined key-frame motions without any control on the velocity or direction. Kick engines designed by other teams have allowed the robot to kick the ball at various speeds and in several directions while allowing the robot to stay dynamically stable during the entire motion but as of yet, the impact of the kick with the ball has never been modelled or controlled by any of the teams or research groups and it has also been a big challenge before to be able to send the ball at required distances. By modelling the impact, we can not only send the ball at the required speeds but also gain the ability to maximize its effect and thus bring out best output from the kick. Further integration of the model with ball dynamics can shift the dependency of the kick engine input from the desired foot velocity to desired displacement of the ball and in this way it acts as a criteria for providing the artificial intelligence to the robot for the decision making during its kick.

The major improvements that have been made as compared to previous kick motions used for ROBOCUP team include:

1. Multidirectional kick motions (0 – 300 angle kicks and sidekicks).
2. More powerful (longer distances of ball) and stable kicks.

3. Intelligent trajectory planning which creates continuous and smooth trajectories in any direction.
4. Controlled impact using the concept of effective mass and momentum control which can be used to give the desired velocity to the ball after impact.
5. Addition of ball dynamics and effects of friction for accurate ball distances.
6. The total time of the kicks have been decreased by 50%

1.3 Thesis Contribution

This thesis provides a complete solution for the generation of dynamically stable kick motions on the Aldebaran NAO humanoid and provides the necessary kinematic, dynamic and control analysis required for the execution of this task while taking into account all the constraints on the robot.

The solution requires the kinematic analysis of the whole humanoid by dividing it into separate chains (Head, arms and legs). For the motion planning of the leg chain manipulators, iterative inverse kinematics algorithms are used and then smooth and continuously differentiable trajectories are used for trajectory generation. Dynamic analysis is performed for both legs and the resultant mass matrices are used to find the effective mass at the final configuration. Using the effective mass in combination with the momentum conservation principles, the impact situation is controlled.

Finally the ball dynamics and floor friction models are introduced in the algorithm to cater for the desired ball displacements.

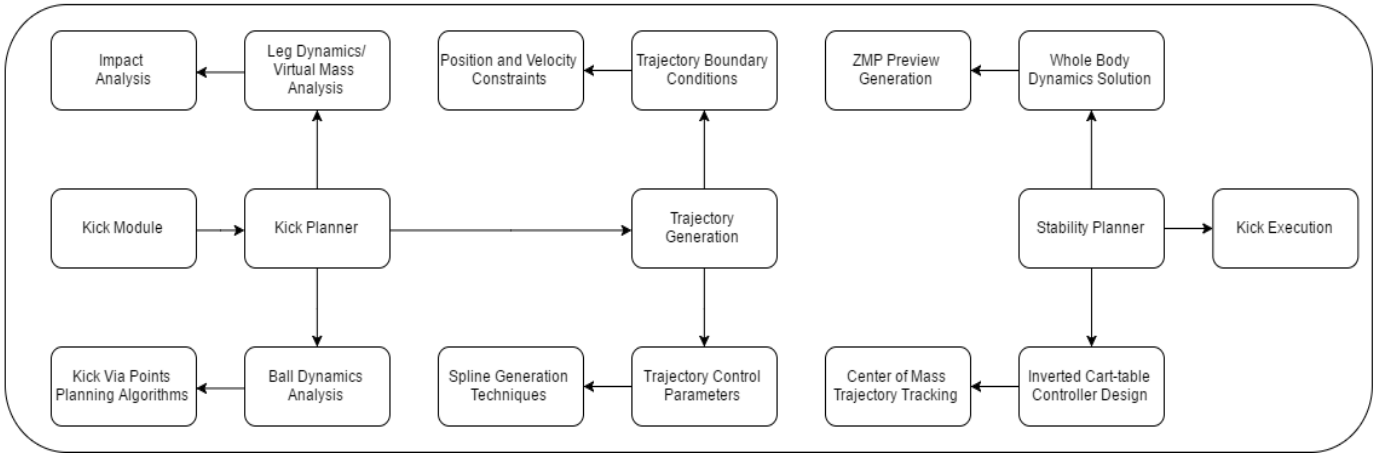


Figure 1.1 Schematic of the kick engine design

To provide the balance to the robot while it is performing the kick, a zero-moment point (ZMP) based criteria has been used. The criteria requires the control of the movement of the robot's ZMP and the implementation of a preview controller to control it. As a requirement for the preview controller, the previews of the ZMP movement are determined its projected movement during the kick trajectory using the whole body dynamics solution.

Furthermore, results through real-time implementation of the algorithm on the robot and several MATLAB codes used for debugging are provided and explained for our teams that will be participating in the coming years. The thesis will also contribute towards the implementation, compiling and building of the C++ source codes and various problems that arise while interfacing with NaoQi software on the robot.

1.4 Thesis Outline

Chapter 1 Introduction

This chapter gives the background introduction and describes the objectives of the research.

Chapter 2 Literature Review

This chapter summarizes all the literature that has been reviewed and identified that has be relevant to this research.

Chapter 3 Methodology:

This chapter provides the overall scheme and analysis that has been carried out for this research. The chapter has several sections explaining in detail the methods and concepts used throughout the research.

Chapter 4 Code Compilation:

This chapter provides a detailed information about the software architecture and the coding issues. It also lists the necessary steps to be able to successfully compile our code for execution.

Chapter 5 Results and Discussion

This chapter provides the results from all the experiments that were carried out to verify our research and discusses the problems that were faced.

Chapter 6 Conclusion

This chapter presents the conclusion of the conducted research along with future recommendations.

Chapter 2

Literature Review

2.1 RoboCup

The idea of autonomous robot soccer was first mentioned by Professor Alan Mackworth (University of British Columbia, Canada) in 1992. Later on, a group of Japanese researchers organized a robotic competition named J-League which by gaining international attention resulted in the formation of RoboCup Federation, "RoboCup" for short. The competition has a challenging mission: "By 2050, a team of fully autonomous humanoid robot soccer players shall win a soccer game against the winner of the most recent World Cup." Therefore, the goal of the competition is provide a publicly appealing platform for the advancement in research on Robotics. Every year, the competition provides certain challenges related to different modules (perception, motion, planning, etc.) and expects from the participating teams to give solutions to those challenges. All the modules of RoboCup (RoboCup@Home, RoboRescue, etc.) are designed such that the proposed solutions from various teams can be tested in real-time. So far, the participating teams have made considerable

progress in solving the real-world problems that show up in the various RoboCup leagues.

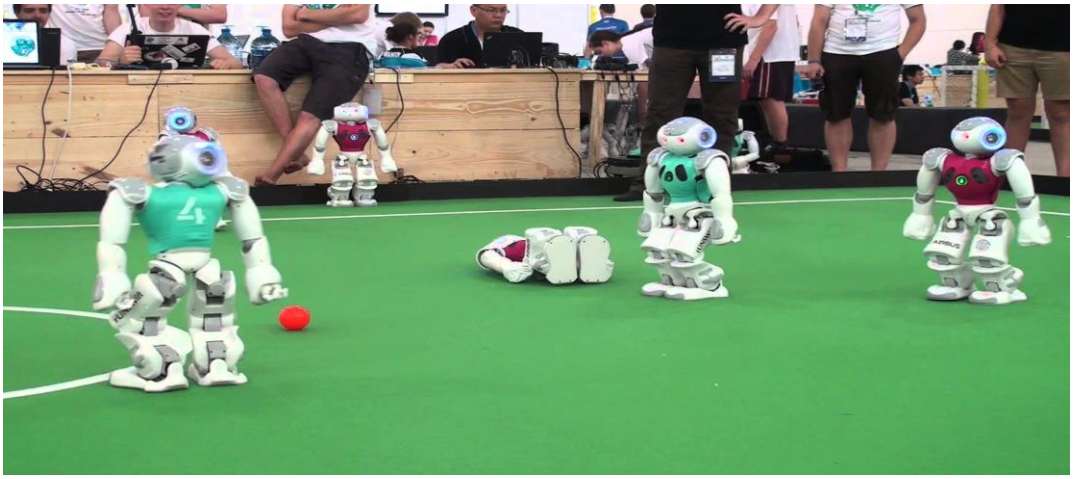
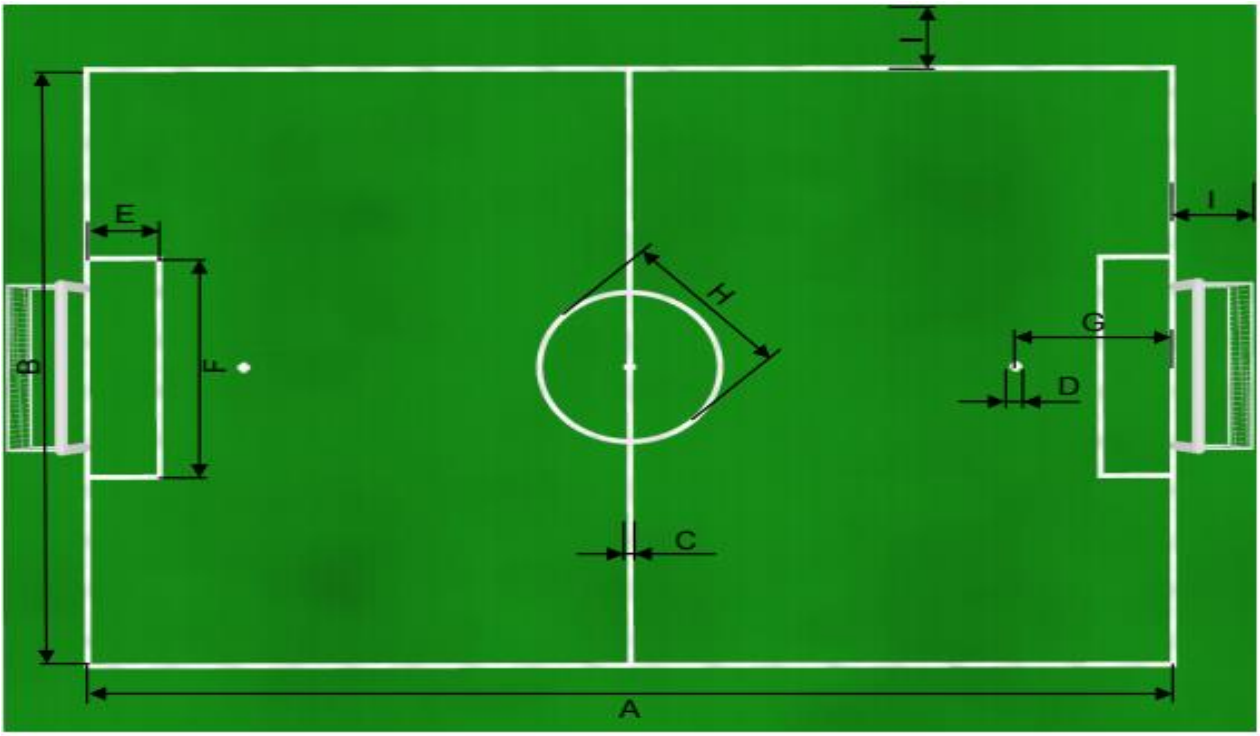


Figure 2.1 2014 RoboCup SPL Pool D: rUNSWift vs HULKs

2.1.1 Standard Platform League

The Standard Platform League (SPL) is one of the major leagues of RoboCup. In this league all the teams are required to use a standard robot, which is chosen to be Aldebaran NAO humanoid robot, and the focus of the teams is only on the algorithm design and software architecture development for this robot. For this reason, it is prohibited for any team to make any modifications to the hardware of the robot. During the gameplay, the robots are completely autonomous and no human intervention from team members is permissible. The robots interact with the environment through their software and the reception of any kind of data from the outside world is through the Game Controller. Game Controller is a computer that broadcasts the information about the state of the game (time, score, penalties, etc.).



ID	Description	Length (in mm)	ID	Description	Length (in mm)
A	Field length	9000	E	Penalty area length	600
B	Field width	6000	F	Penalty area width	2200
C	Line width	50	G	Penalty mark distance	1300
D	Penalty mark size	100	H	Center circle diameter	1500
			I	Border strip width	700

Figure 2.2 Field Dimensions from RoboCup Rulebook 2016

Currently, the field dimensions for the SPL gameplay are 4m×6m (Figure 2.2). It consists of a green carpet marked with white lines and two white goals. The ball is a spotted black and white ball similar to a real football. During the match, each team consists of four robots and each robot carries a colored waist band (blue or pink) that differentiates the teams. Total time for the match is 20 minutes with 10 minutes for each half.

2.1.2 Team-NUST

Team-NUST is the RoboCup team of the National University of Sciences and Technology (NUST). The team was founded in 2014 and is looking forward

to participate in RoboCup '16 SPL competition. The team has been developing its own software for NAO robots since 2014. The software includes a graphical interface, a perception framework for object recognition and state estimation, localization, and a behavior execution architecture. This year's team consists of four postgraduate students and one undergraduate student.

2.2 NAO Humanoid Robot

NAO is a medium-sized humanoid robot developed by Aldebaran Robotics in Paris, France. It officially substituted Sony's AIBO quadruped robot in the RoboCup SPL in August 2007.

Figure 2.3 Humanoid NAO

NAO (version V3.3) is a 58cm, 5kg humanoid robot. It carries a fully capable on-board computer with an AMD Geode processor at 500 MHz, 256 MB SDRAM, and 2 GB flash memory running an Embedded Linux distribution. Power is provided by a 6-cell Lithium-Ion battery which allows about half an hour of continuous operation and the communication of the robot is made possible via IEEE 802.11g wireless or a wired ethernet link.

NAO has a total of 22 degrees of freedom; 2 in the head, 5 in each arm, 5 in each leg and 1 in the pelvis (the two pelvis joints are joined together on a single servo motor and cannot move independently). It also consists of a variety of sensors. Each servo motor is accompanied by a magnetic rotary

encoder with an accuracy of $\pm 1^{o}$ which records the actual values of all joints and continuously updates it in the robot memory. It has two cameras mounted on the head in vertical alignment providing the views of the lower and distant frontal areas. Each camera is a 640×480 resolution device operating at 30fps. Four sonars (two emitters and two receivers) on the chest allow NAO to sense obstacles in front of it. One of the most promising is the inertial unit, with a 3-axis accelerometer and a 2-axis gyroscope in the torso which can instantaneously provide real-time information about its body. Two bumpers located at the front of each foot are simple switches and can provide information about collisions of the feet with obstacles. Last of all, four force sensitive resistors are provided on each foot to deliver the feedback of the forces applied on the feet.

2.3 Robot Kinematics

A humanoid is typically defined as a combination of 'n' number of articulated manipulators joined together on a single base link which is usually the main body or torso of the robot. An articulated manipulator is defined by a kinematic chain, which is a set of links subsequently joined together by a certain type of joint of which the final operational point is called an end effector. Generally, two types of joints are used in articulated manipulators, prismatic (translational motion) or revolute (rotational motion) while in case of NAO, only revolute joints will be considered. The number of joints in the chain defines the number of degrees of freedom (DOF). Depending upon the DOFs the mobility and redundancy of the chain is described. If an analysis of the whole robot is required then each of the 'n' chains have to be analyzed

separately and then combined at the base link thus the total degrees of freedom of the robot are equal to the sum of degrees of freedom of the individual kinematic chains.

The kinematics analysis of the robot means to study and define the relationship between the movement of the links and the joints for each of the kinematic chains. In particular, this analysis defines a mapping from the joint space to the Cartesian space or vice versa. This analysis is not only needed for the positioning and trajectory planning of the manipulator but also to construct the mathematical model of the robot.

2.3.1 Kinematic Chain

A kinematic chain with 'n' number of joints is defined by a set of 'n' frames subsequently attached to each joint. The relationship between a pair of frames is defined by the relative rotation or translation of those two frames with respect to one another, whereas it is modelled in the form of a transformation matrix. A transformation matrix is defined as a mapping that transforms points, vectors or frames from one space to another all while preserving the distance ratios.

For an n-dimensional space, the matrix defining the transformation from one frame to another is defined as an $(n + 1) \times (n + 1)$ matrix of the form:

$$T = \begin{bmatrix} R & A \\ [0,0,0] & 1 \end{bmatrix}$$

Where R is a (n×n) rotation matrix defining the rotation between the two frames, A is a (n×1) translation vector and the last line of T contains (n−1) zeros followed by a 1. For a series of transformations, the transformation matrices can be multiplied subsequently to get a single resultant transformation:

$$T = T_3T_2T_1 = \begin{bmatrix} R_3 & A_3 \\ [0,0,0] & 1 \end{bmatrix} \times \begin{bmatrix} R_2 & A_2 \\ [0,0,0] & 1 \end{bmatrix} \times \begin{bmatrix} R_1 & A_1 \\ [0,0,0] & 1 \end{bmatrix}$$

2.3.1.1 Translation:

If we need to translate a frame or a point with respect to a reference frame in three dimensional space, we can simply define a (4x4) transformation matrix of the 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. To translate a column vector $(p_1, p_2,..., p_n)^T$ along the distances d_x, d_y, and d_z, the translation matrix is just pre-multiplied by a vector $(p_1, p_2, ..., p_n, 1)^T$.

$$\begin{bmatrix} P_1' \\ P_2' \\ P_3' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ 1 \end{bmatrix}$$

2.3.1.2 Rotation:

If we need to rotate a frame about a given axis with respect to a reference frame in any n-dimensional space then a rotation matrix is described as an (n×n) orthogonal matrix R:

$$R^T = R^{-1} \qquad RR^T = R^T R = I$$

In the 3D Cartesian space there are generally three most important rotations, each of which performs a rotation of θ_x, θ_y, θ_z about the x, y, z axis respectively, assuming a right-handed coordinate system:

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

As an example, the rotation matrix that rotates a vector or a frame first about the x axis, then about the y axis, and then about the z axis will be defined as following:

$$R = R_z R_y R_x$$

The resultant form of the above matrix will be:

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

Finally, this rotation matrix can be substituted in the transformation matrix:

$$T = \begin{bmatrix} cos\theta_y cos\theta_x & -cos\theta_x sin\theta_z + sin\theta_x sin\theta_y cos\theta_z & sin\theta_x sin\theta_z + cos\theta_x sin\theta_y cos\theta_z & 0 \\ cos\theta_y sin\theta_x & cos\theta_x cos\theta_z + sin\theta_x sin\theta_y sin\theta_z & -sin\theta_x cos\theta_z + cos\theta_x sin\theta_y sin\theta_z & 0 \\ -sin\theta_y & sin\theta_x cos\theta_y & cos\theta_x cos\theta_y & 0 \\ & [0,0,0] & & 1 \end{bmatrix}$$

2.3.2 Forward Kinematics

The forward kinematics gives a transformation from the joint space to the 3D Cartesian space. For a certain kinematic chain with 'n' number of joints and a group of joint values (θ_1, θ_2,…,θ_n), the forward kinematics can be used to find the position (p_x ,p_y, p_z) and the orientation (a_x, a_y, a_z) of the final operational point or end effector of the kinematic chain in the X-Y-Z - Cartesian space. Forward kinematics can be solved for any simple or complex kinematic chain resulting in a closed-form, analytical solution.

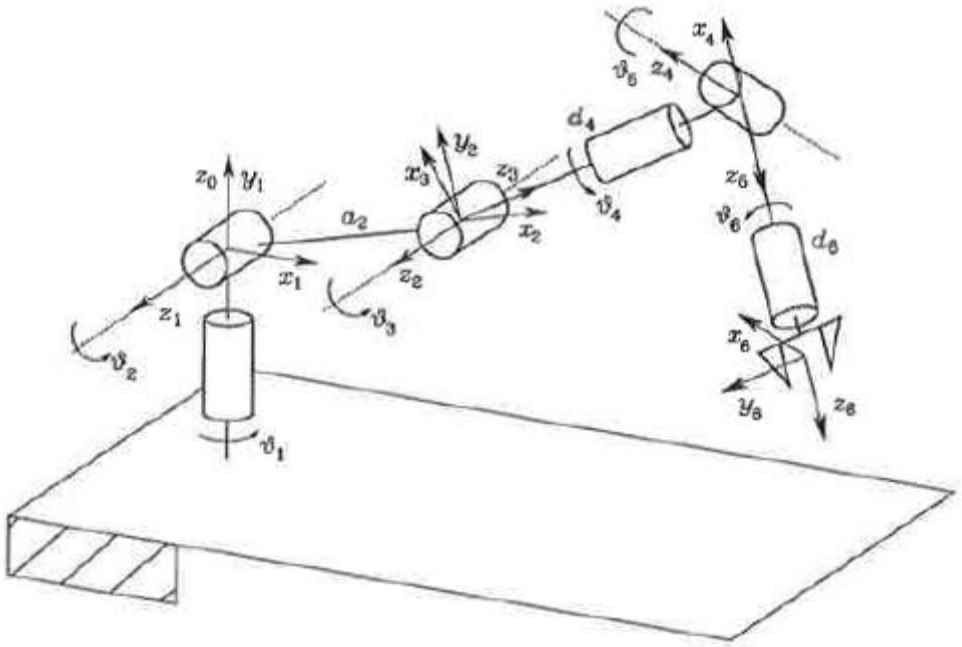


Figure 2.4 Forward Kinematics of a standard 6 DOF Robot

2.3.3 Inverse Kinematics

The end effectors of robotic manipulators are usually needed to reach certain target points or follow predefined trajectories in the 3D Cartesian space. To do that, a set of suitable values for the joints of the kinematic chain are needed. More specifically, the inverse kinematics defines a relationship between position (p_x, p_y, p_z) and orientation (a_x, a_y, a_z) of the end effector in the three-dimensional space and the joint positions (θ_1, θ_2,...,θ_n) in the joint space of a kinematic chain with 'n' joints. The problem with the inverse kinematics is that it is that a single configuration of the end effector can be achieved through multiple sets of the joint positions. This lack of uniqueness in the solution denies the direct transformation of Cartesian configuration (p_x, p_y, p_z, a_x, a_y, a_z) into joint space configurations (θ_1, θ_2,...,θ_n). Therefore the solution to the inverse kinematics problem can be achieved either through a numerical, iterative approach or a closed-form analytical method with certain constraints.

2.3.4 Denavit-Hartenberg (DH) Parameters

Denavit and Hartenberg have introduced a simple and effective method for investigating the kinematics of a manipulator. They concluded that we can fully define a relationship between any two coordinate frames attached to different joints of a manipulator by using only four parameters, known as Denavit-Hartenberg (DH) parameters: a, α, d, and θ (Figure 2.5)

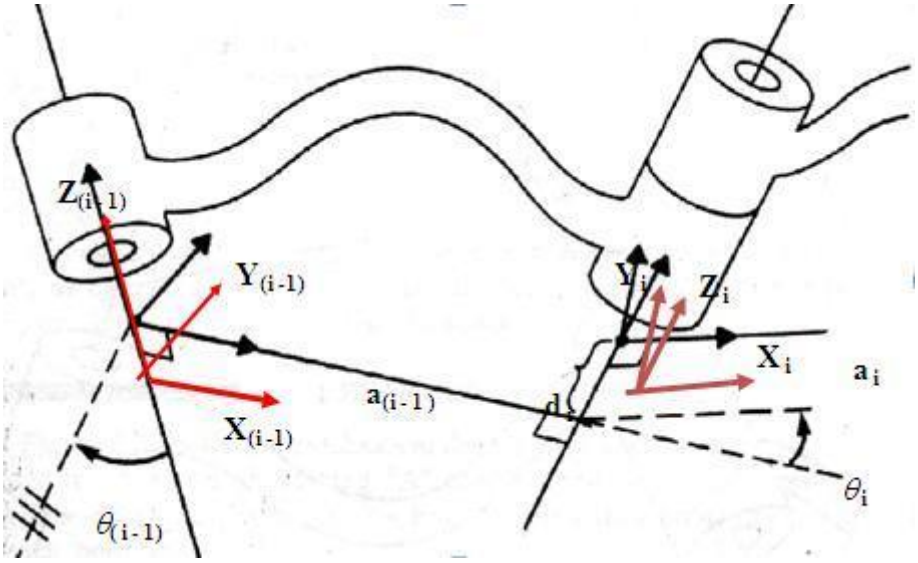


Figure 2.5 DH Parameters defined for the transformation from joint i-1 to i

To define these parameters, first we need to define the axis of rotation or translation of the concerned joint. This is taken as the Z axis of the joint frame. Once the individual joint axis *i* and *i* -1 are determined, a perpendicular is drawn from the axis *i* -1 to *i*. Then we attach the coordinate frames with Z axes of the frames along the previously defined joint axes and the X axis of the frame *i* is defined along the perpendicular. Finally, the DH parameters for the propagation from frame *i* -1 to frame *i* are defined as following:

1. a_i : distance (Z_i, Z_{i+1}) along X_i
2. α_i : angle (Z_i, Z_{i+1}) along X_i
3. d_i : distance (X_{i-1},X_i) along Z_i
4. θ_i : angle (X_{i-1},X_i) about Z_i

For the first frame, an initial reference frame is required to define it while for the last frame, an additional frame is required to which it is

propagated. Finally, the relation between a pair of links is defined on the basis of a transformation matrix:

$$T_i^{i-1} = \begin{bmatrix} cos\theta_i & -sin\theta_i & 0 & a_{i-1} \\ sin\theta_i cos\alpha_{i-1} & cos\theta_i cos\alpha_{i-1} & -sin\alpha_{i-1} & -d_i sin\alpha_{i-1} \\ sin\theta_i sin\alpha_{i-1} & cos\theta_i sin\alpha_{i-1} & cos\alpha_{i-1} & d_i cos\alpha_{i-1} \\ & [0,0,0] & & 1 \end{bmatrix}$$

In this way, a whole kinematic chain can be defined by choosing appropriate DH parameters between each link based on the transformation matrices.

2.3.5 Instantaneous Kinematics:

By this point, we have determined the position of the subsequent link frames with respect to the base frame but to delve into the dynamic analysis of the robot, we first need to take into account the motion of each link. As described, each link is connected by the previous link by a joint. To determine the effect of instantaneous movement of that joint (rotation for revolute and translation for prismatic) on the following kinematic chain, we need to define a relationship that transforms the velocities of the joint movements $(\dot{\theta}_1, \dot{\theta}_2, \dots, \dot{\theta}_n)$ in the joint space into the linear and angular velocities $(\dot{x}_1, \dot{x}_2, \dots, \dot{x}_m)$ of the link in Cartesian space. The required relationship is defined on the basis of a matrix J:

$$dX = Jd\theta \qquad or \qquad \dot{X} = J\dot{\theta}$$

Where J is called the Jacobian matrix which is an $(m \times n)$ matrix defined as,

$$J = \begin{pmatrix} \frac{\partial x_1}{\partial \theta_1} & \cdots & \frac{\partial x_1}{\partial \theta_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial x_m}{\partial \theta_1} & \cdots & \frac{\partial x_m}{\partial \theta_n} \end{pmatrix}$$

To define the linear and angular velocities ($\dot{x}$, $\dot{y}$, $\dot{z}$, w_x, w_y, w_z)of a point in Cartesian space, we can divide J into two parts; linear Jacobian J_v and angular Jacobian J_w which are defined as:

$$J_v = \begin{pmatrix} \frac{\partial x}{\partial \theta_1} & \cdots & \frac{\partial x}{\partial \theta_n} \\ \frac{\partial y}{\partial \theta_1} & \cdots & \frac{\partial y}{\partial \theta_n} \\ \frac{\partial z}{\partial \theta_1} & \cdots & \frac{\partial z}{\partial \theta_n} \end{pmatrix} \qquad J_w = \begin{pmatrix} \frac{\partial a_x}{\partial \theta_1} & \cdots & \frac{\partial a_x}{\partial \theta_n} \\ \frac{\partial a_y}{\partial \theta_n} & \cdots & \frac{\partial a_y}{\partial \theta_n} \\ \frac{\partial a_z}{\partial \theta_n} & \cdots & \frac{\partial a_z}{\partial \theta_n} \end{pmatrix}$$

Therefore, to find the total Jacobian, we first need to determine the Cartesian space position (x, y, z) and orientation (a_x, a_y, a_z) of the operational point in terms of joint space coordinates ($\theta_1, \theta_2, \ldots, \theta_n$) which is achieved by the kinematic analysis. Then, by differentiating the parameters with respect to the joint space parameters as shown above, we can find the linear and angular Jacobians. Finally the total Jacobian J is given by:

$$J = \begin{bmatrix} J_v \\ J_w \end{bmatrix}$$

2.4 Robot Dynamics

In the last section, we saw how to describe the effect of motion of the joints on the motion of the manipulator chains that make up the robot. In that section, we did not talk about the effects of the inertial properties, external forces or the forces arising from the individual joint accelerations or velocities on the manipulator. In this chapter, we will look more closely at the effects of dynamics on execution of manipulator trajectories and how the trajectories are closely followed by applying control principles.

Generally, two approaches are used for obtaining the dynamic model of a robotic manipulator; Langrangian formulation and Newtom-Euler Formulation. Both of these methods result in a dynamic equation:

$$\Gamma = M(\Theta)\ddot{\Theta} + V(\Theta, \dot{\Theta}) + G(\Theta)$$

Where $M(\Theta)$ is the mass matrix which defines the mass distribution of the robot for any joint angles configuration $(\theta_1, \theta_2, \dots, \theta_n)$. $V(\Theta, \dot{\Theta})$ is the term defining the centrifugal and Coriolis forces. The term $G(\Theta)$ defines the gravity forces while Γ is the vector for resultant joint torques.

The difference between the two methods is that the Langrangian formulation provides an explicit form for the dynamic equation in terms of the joint positions, velocities and accelerations while Newton-Euler approach is solved recursively and directly gives the resultant joint torques. A brief explanation is provided for both of these approaches below.

2.4.1 Lagrangian Formulation:

This approach is based on the well-known Lagrange equation of motion:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = \Gamma_i$$

Where L is the 'lagrangian' which is defined as the difference of kinetic and potential energies:

$$L = K.E - P.E$$

For the kinetic energy calculation of a serial manipulator with n joints, the kinetic energy of each link is found and then added to find the total. To find the individual link kinetic energy, its center of mass velocity is needed with respect to the base frame in terms of joint space coordinates. Thus for this purpose, we need to find the center of mass linear and angular velocity jacobians for the transformation. For any link i in serial chain a center of mass vector in its own frame is given by:

$$P_{ci} = (c_{xi}, c_{yi}, c_{zi})^T$$

A transformation T_i^{Base} is found through the kinematic analysis to convert the vector from frame i to the base frame:

$$P_{ciBase} = T_i^{Base} P_{ci}$$

Similarly, the orientation A_i of the link i can be transformed into the base frame coordinates by:

$$A_{iBase} = T_i^{Base} A_i$$

Using the center of mass position P_{ciBase} and orientation A_{iBase} of the link i, the linear and angular jacobian matrices for the center of mass velocities can be found by:

$$J_{vi} = \begin{bmatrix} \frac{\partial P_{ci}}{\Theta_1} & \frac{\partial P_{ci}}{\Theta_2} & \frac{\partial P_{ci}}{\Theta_3} & 0 & \dots & 0 \end{bmatrix}$$

$$J_{wi} = \begin{bmatrix} \frac{\partial A}{\partial \theta_1} & \frac{\partial A}{\partial \theta_2} & \frac{\partial A}{\partial \theta_i} & 0 & \dots & 0 \end{bmatrix}$$

The kinetic energy of the center of mass of link i in Cartesian space (in matrix form) resulting from the linear and angular motions is defined by the following equation:

$$K.E_i(X, \dot{X}) = \frac{1}{2}(m_i v_{ci}^T v_{ci} + w_{ci}^T I_{ci} w_{ci})$$

Where m_i is the mass of the link i and I_{ci} is its inertia tensor defined at the center of mass. v_{ci} and w_{ci} are the center of mass linear and angular velocities. For the kinetic energy in joint space, following substitutions are made:

$$v_{ci} = J_{vi}\,\dot{\theta}$$

$$w_{ci} = J_{wi}\,\dot{\theta}$$

Thus, $$K.E_i(\Theta, \dot{\Theta}) = \frac{1}{2}(m_{ci}\dot{\theta}^T {J_{vi}}^T J_{vi}\,\dot{\theta} + \dot{\theta}^T {J_{wi}}^T I_{ci} J_{wi}\,\dot{\theta})$$

$$K.E_i(\Theta, \dot{\Theta}) = \frac{1}{2}\dot{\theta}^T(m_{ci}{J_{vi}}^T J_{vi} + {J_{wi}}^T I_{ci} J_{wi})\dot{\theta}$$

Or, $$K.E_i(\Theta,\dot{\Theta}) = \frac{1}{2}\dot{\theta}^T M(\Theta)_i \dot{\theta}$$

Where $M(\Theta)_i$ is the generalized inertia matrix for the link i. The total kinetic energy can now be written as:

$$K.E(\Theta,\dot{\Theta}) = \sum_i^n K.E_i(\Theta,\dot{\Theta}) = \frac{1}{2}\dot{\theta}^T M(\Theta)\dot{\theta}$$

The manipulator inertia matrix:

$$M(\Theta) = \sum_i^n (m_{ci} J_{vi}{}^T J_{vi} + J_{wi}{}^T I_{ci} J_{wi})$$

To complete the lagrangian formulation, we also need to find the potential energy of each link of the manipulator. If $h_i(\Theta)$ is the height of the center of mass of link i defined by the z-component of P_{ciBase}, then the potential energy of link i is defined as:

$$P.E_i(\Theta) = m_i g h_i(\Theta)$$

$$P.E(\Theta) = \sum_i^n P.E_i(\Theta) = \sum_i^n m_i g h_i(\Theta)$$

Thus, the langrangian is,

$$L(\Theta,\dot{\Theta}) = \frac{1}{2}\dot{\theta}^T M(\Theta)\dot{\theta} - P.E(\Theta)$$

The lagrangian $L(\Theta,\dot{\Theta})$ is then substituted into the lagrange equation of motion with joint parameters $(\theta_1, \theta_2, \ldots, \theta_n)$ to find the resultant joint toques.

$$\frac{d}{dt}\frac{\partial L(\Theta,\dot{\Theta})}{\partial \dot{\Theta}_i} - \frac{\partial L(\Theta,\dot{\Theta})}{\partial \Theta_i} = \Gamma_i$$

2.4.2 Newton-Euler Formulation:

For the derivation of this method, two fundamental physical relationships are used, namely, Newton's Equation and Euler's Equation:

$$F = m\dot{v}_c$$

$$N = I_c\dot{w} + w \times I_c . w$$

The method is made up of two parts: A forward recursion and a backward recursion. In the first part, the velocities and accelerations (linear and angular) are computed for each joint. In case of rotational joints, the velocities and accelerations for the joint $i+1$ are given by:

$$^{i+1}_{i+1}w = {}^{i+1}_{i}R \,.\, ^{i}_{i}w + \dot{\Theta}_{i+1} . {}^{i+1}_{i+1}Z$$

$$^{i+1}_{i+1}\dot{w} = {}^{i+1}_{i}R \,.\, ^{i}_{i}\dot{w} + {}^{i+1}_{i}R \,.\, ^{i}_{i}\mathrm{w} \times \dot{\Theta}_{i+1} . {}^{i+1}_{i+1}Z + \ddot{\Theta}_{i+1} . {}^{i+1}_{i+1}Z$$

The linear accelerations,

$$^{i+1}_{i+1}\dot{v} = {}^{i+1}_{i}R({}^{i}_{i}\dot{w} \times {}_{i+1}{}^{i}P + {}^{i}_{i}\mathrm{w} \times ({}^{i}_{i}\mathrm{w} \times {}_{i+1}{}^{i}P) + {}^{i}_{i}\dot{v})$$

It is necessary to find the accelerations of the center of masses of each link which is given by:

$$^{i}_{i}\dot{v}_{c} = {}^{i}_{i}\dot{w} \times {}^{i}_{i}P_c + {}^{i}_{i}\mathrm{w} \times ({}^{i}_{i}\mathrm{w} \times {}^{i}_{i}P_c) + {}^{i}_{i}\dot{v}$$

The next step is to find forces and moments being applied at the center of mass of each link from the accelerations and velocities being produced.

$$ {}^{i}_{i}F = m_i . {}^{i}_{i}\dot{v}_c $$

$$ N = {}^{i}_{i}I_c . {}^{i}_{i}\dot{w} + {}^{i}_{i}w \times {}^{i}_{i}I_c . {}^{i}_{i}w $$

Finally, the backward recursion is applied for finding the forces and moments on the joints of the robot

$$ {}^{i}_{i}f = {}^{i}_{i+1}R \cdot {}^{i+1}_{i+1}f + {}^{i}_{i}F $$

$$ {}^{i}_{i}n = {}^{i}_{i}N + {}^{i}_{i+1}R . {}^{i+1}_{i+1}n + {}^{i}_{i}P_c \times {}^{i}_{i}F + {}^{i}_{i+1}P \times {}^{i}_{i+1}R . {}^{i+1}_{i+1}f $$

Finally, the values of Γ_i are computed as $\Gamma_i = {}^{i}_{i}n^T . {}^{i}_{i}Z$.

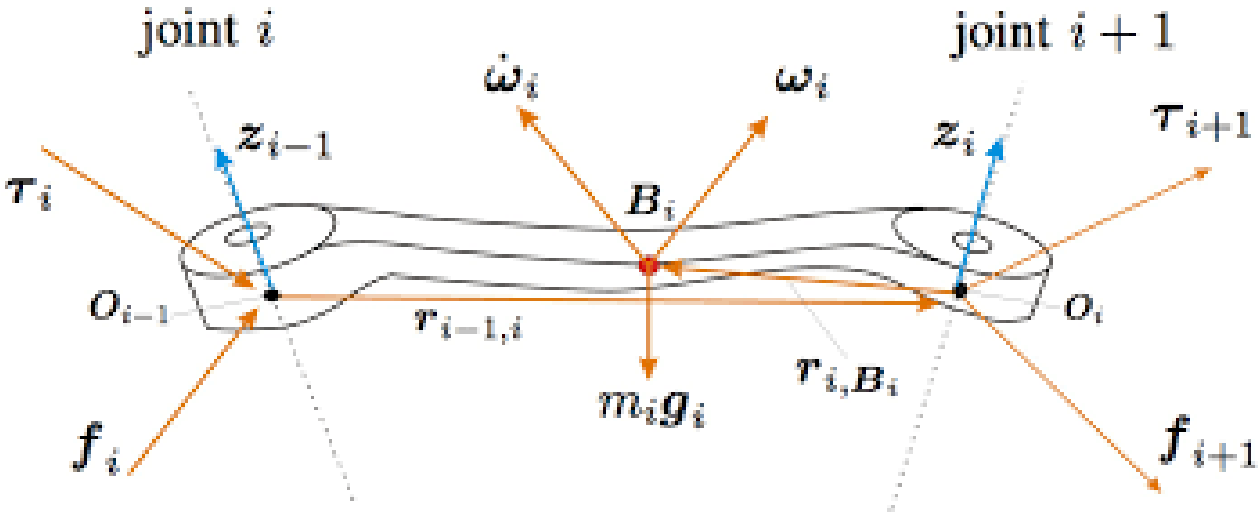


Figure 2.6 Forces and torques at the end-points and at the center of mass of a single link are shown

2.4.3 Effective Mass

For the analysis of inertial properties of the articulated robot manipulators, two types of tasks are generally studied; operational point translational task and operational point rotational tasks. Given the manipulator redundancy with respect to the required task, the dynamic behaviour of the operational point end-effect can be described in the form of a matrix:

$$M(\Theta)_{op} = J^{-T}(\Theta)\, M(\Theta)\, J^{-1}(\Theta)$$

Where $M(\Theta)_{op}$ is the operational space kinetic energy matrix with the kinetic energy defined by '$\frac{1}{2}\dot{x}^T M(\Theta)_{op}\dot{x}$'.

Let us consider a task of positioning an end-effector with only its linear velocity under consideration. The Jacobian in this case is the matrix $J(\Theta)_v$ associated with the linear velocity at the operational point. The inverse of $M(\Theta)_{op}$ is then defined by:

$$M(\Theta)^{-1}{}_{op} = J_v(\Theta)\, M(\Theta)^{-1}\, J_v(\Theta)^T$$

The matrix $M(\Theta)^{-1}{}_{op}$ is called the pseudo kinetic energy matrix and provides a description of the end-effector translational response to a force. For instance, if the response of the end-effector is considered along the direction specified by a unit vector $\boldsymbol{u}$, the response is perceived as:

$$\frac{1}{m_{eff}} = \boldsymbol{u}^T M(\Theta)^{-1}{}_{op}\boldsymbol{u}$$

Where, the term m_{eff} is the effective mass of the operational point in the direction $\boldsymbol{u}$ while its inverse represents the component of linear acceleration which results in response to a unit force applied along $\boldsymbol{u}$.

2.5 Humanoid Stability

Unlike most of the articulated robots with a fixed base, a humanoid is self-contained and thus it has no support by any means. The robot needs to maintain its balance while performing complicated tasks and therefore it requires some kind of criteria on the basis of which we can define its stability. The stability of a body is generally defined in two types:

1. Static stability
2. Dynamic stability

2.5.1 Static stability

Static stability of a body is usually defined on the basis of the position of the center of mass (COM) of the body with respect to the body's support polygon. The support polygon is an area defined by the contact points of the body with the ground. For example, a robot standing on one feet as shown in Figure 2.7, the support polygon will be defined as contour of the footprint on the ground. Finally, when a body is under static equilibrium such that there is no acceleration at the center of mass (COM), the body remains in balance as long as the COM is positioned inside the support polygon.

Figure 2.7 Static Balance

2.5.2 Dynamic stability

To perform complicated human-like motions, a humanoid is sometimes needed to be in acceleration. In such cases, the dynamic counter part of the center of mass, namely, zero moment point (ZMP) is used to define the stability.

Zero Moment Point:

The ZMP is a point where the total torque acting on the body is zero. The position of the ZMP can be defined by a point in Cartesian space (X, Y, Z). Since we are only interested in the positioning of ZMP in the support polygon on the ground plane, let us assume that Z = 0. To facilitate the analysis, let us consider the foot of a robot (Figure2.8) under the action of a force F_A and moment M_A. While, the weight of the foot acts at the center of gravity (G). The foot is also experiencing the ground reaction at P, which is keeping the whole body under equilibrium.

By analysing the situation, we can see that the ground reaction has three components of the force R (R_x, R_y, R_z) and moment M (M_x, M_y and M_z). For the foot to be at rest, the components of the force F_A and moment M_A (M_{Az}) that act in the horizontal plane will be balanced by the frictional forces. This friction is represented by the horizontal reaction forces (R_x, R_y) and the vertical reaction moment (M_z). Thus, with the assumption that the foot-floor contact is not undergoing sliding motion, the friction components (R_x, R_Y, M_{AZ}) will cancel out the horizontal force components (F_{Ax}, A_y) and the vertical torque component (M_{Az}). Therefore, only the vertical component R_z of the reaction force is left to balance the vertical forces and the moments acting

along the horizontal axis (M_{Ax}, M_{Ay}). As the force is in upward direction, the moments can be balanced out by changing the position of the reaction force component R_Z within the support polygon. However, if the support polygon is not large enough to compensate for the acting moments on the body, the position of the reaction force component will move towards the edge of the foot and the part of the moment (M_{Ax}, M_{Ay}) which is left unbalanced will overturn the body about the edge of the foot. Therefore, it can be said that the necessary condition for the body to stay in dynamic stability is that the total moment acting at the point P on the ground is zero, that is,

$$M_x = 0, \qquad M_y = 0$$

Thus this point with zero moments in both directions is the zero moment point (ZMP). With the origin of the coordinate system placed at point P, the necessary conditions for finding out the ZMP are then defined by:

$$R + F_A + m_{foot}g = 0 \qquad OP \times R + OG \times m_{foot}g + M_A + OA \times F_A = 0$$

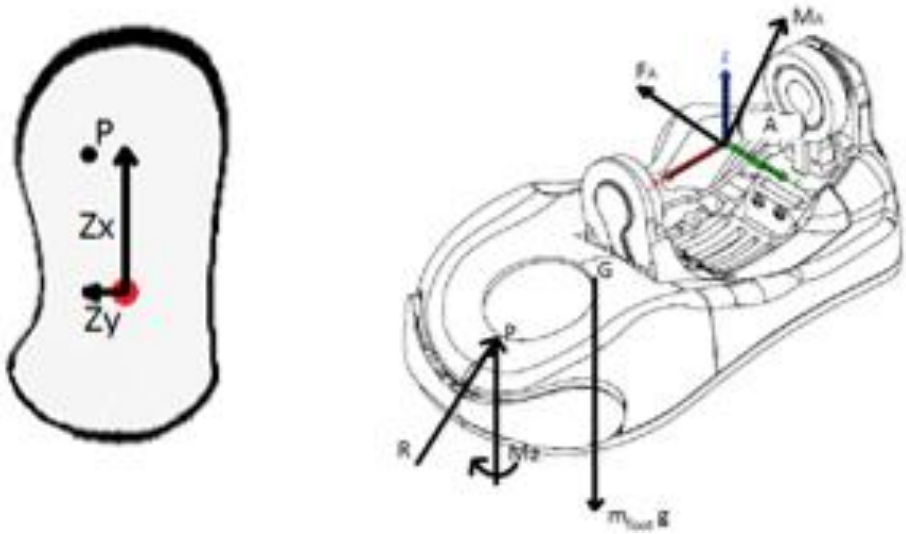


Figure 2.8 Forces and ZMP Position on Foot

2.6 Collision Models

An event in which two or more bodies exert forces on each other for relatively small time is called collision. In collisions, momentum is always conserved but kinetic energy may or may not be conserved.

2.6.1 Elastic and Inelastic Collisions

There are two different types of collision, elastic in which they conserve both kinetic energy and momentum while inelastic collision is when they conserve momentum but no kinetic energy. Inelastic collision is also sometimes referred as plastic collision.

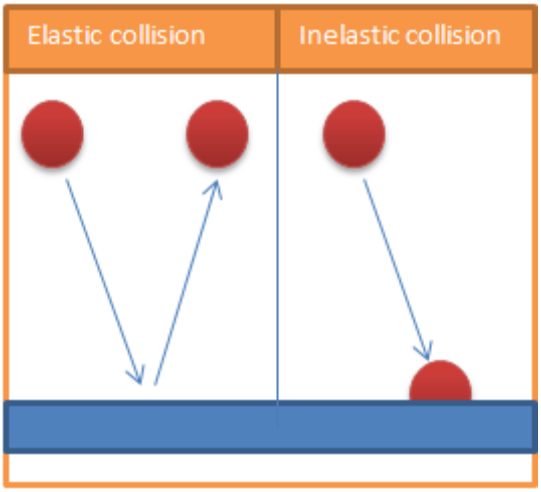
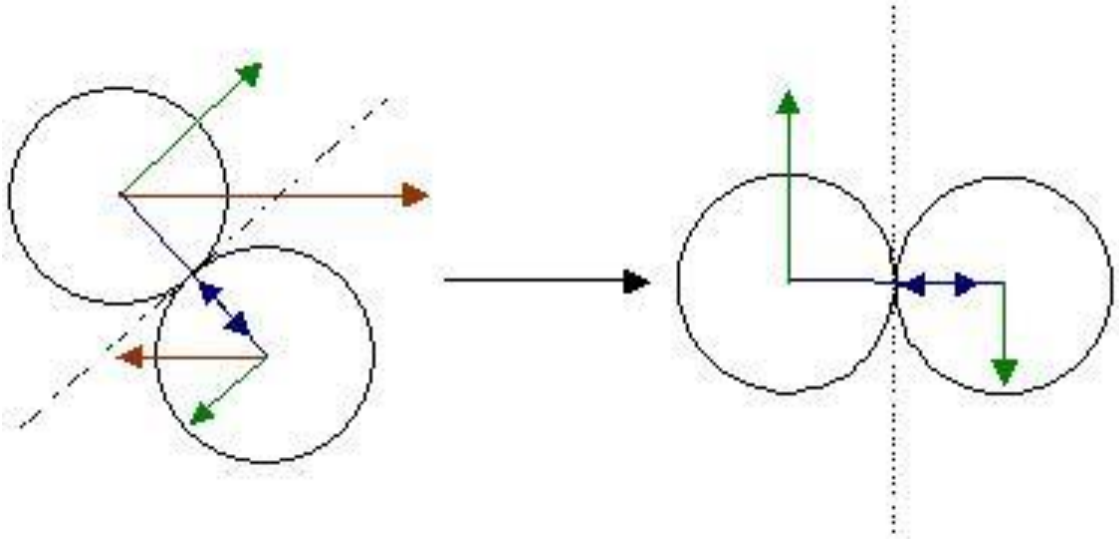


Figure 2.9 Elastic and Inelastic Collisions

2.6.2 Collision Dimensionality

There can be different types of collisions between two bodies. A Head on collisions also known as one-dimensional collisions is the one in which the velocity of each body is along the line of impact before the impact. Other type is non-head on collisions, oblique collisions or two-dimensional collisions in

which the velocity of each body is not along the line of impact before the impact.



(a) Oblique Collision (b) Head-on Collision

Figure 2.10 Head on and Oblique Collisions

2.6.3 Coefficient of Restitution

The amount to which a collision is elastic or inelastic is defined by the coefficient of restitution. This value ranges between zero and one. A perfectly elastic collision has a coefficient of restitution of one and a perfectly inelastic collision has a coefficient of restitution of zero.

Chapter 3

Methodology

3.1 Robot Specifications

The version of the robot we worked on is NAO H25 with 23 DOFs. The robot has two DOFs in the head, five DOFs in each arm, five DOFs in 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:

1. Head: HeadYaw, HeadPitch
2. Left Arm: LShoulderPitch, LShoulderRoll, LElbowYaw, LElbowRoll, LWristYaw
3. Right Arm: RShoulderPitch, RShoulderRoll, RElbowYaw, RElbowRoll, LWristYaw
4. Left Leg: LHipYawPitch, LHipRoll, LHipPitch, LKneePitch, LAnklePitch, LAnkleRoll
5. 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 whole body joint arrangement.

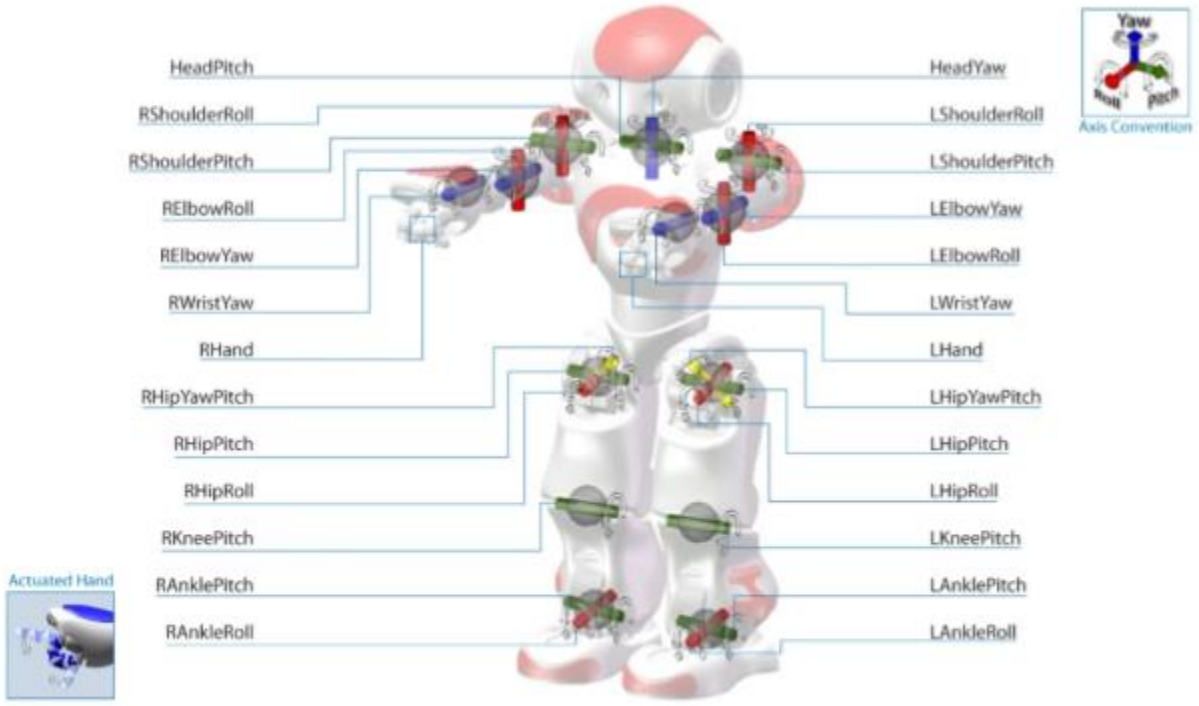


Figure 3.1 NAO Robot Joint Arrangement

To fully specify the joints of the robot, we have provided the links lengths (Table 3.2), the operational range in degrees of the upper body (Figure 3.3) and lower body joints (Figure 3.4), the mass and center of mass (Table 3.1) and the intertial properties (Table 3.2) of each joint. All these values have been taken from the NaoQi 2-1 Documentation provided by Aldebaran Robotics.

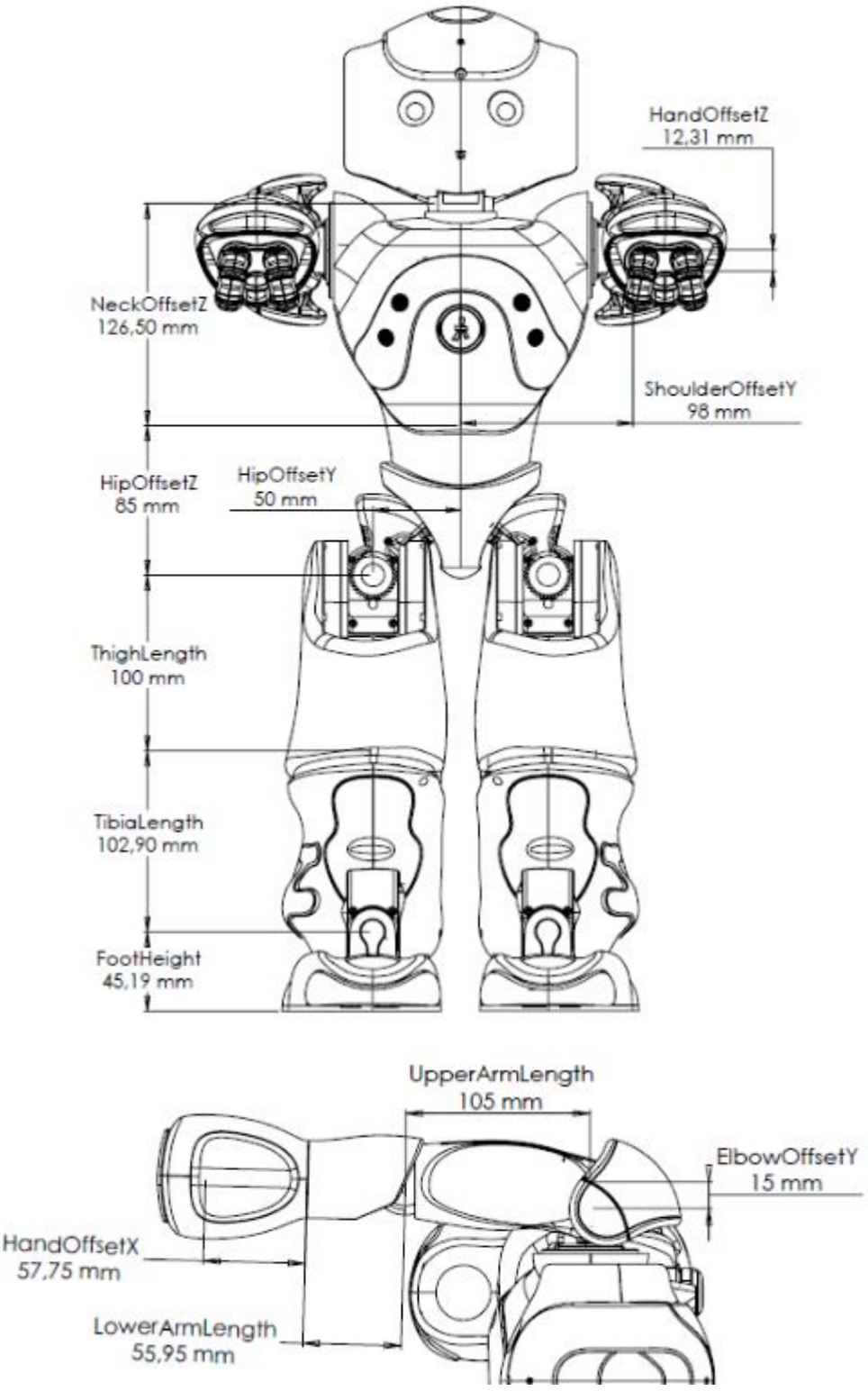
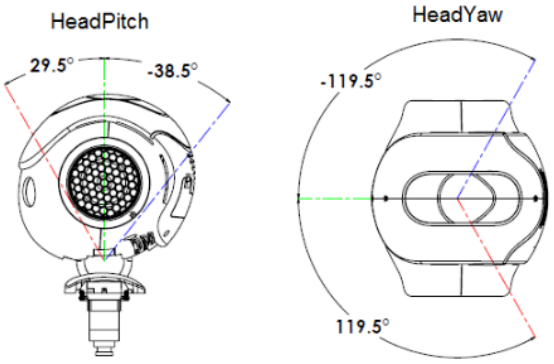
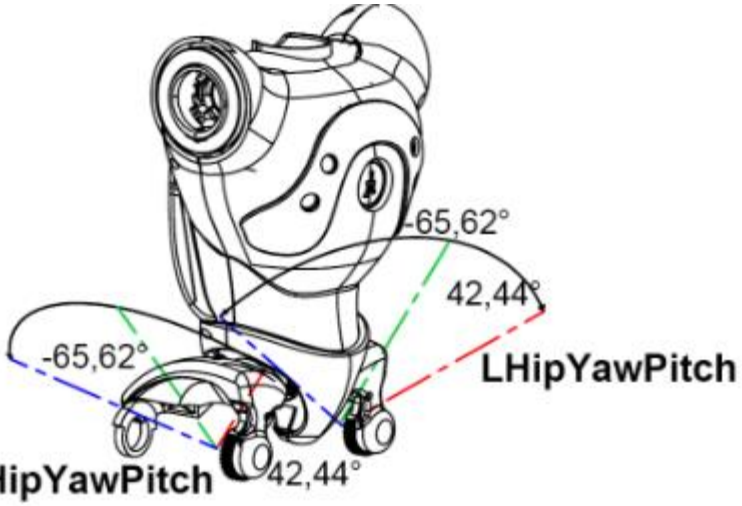


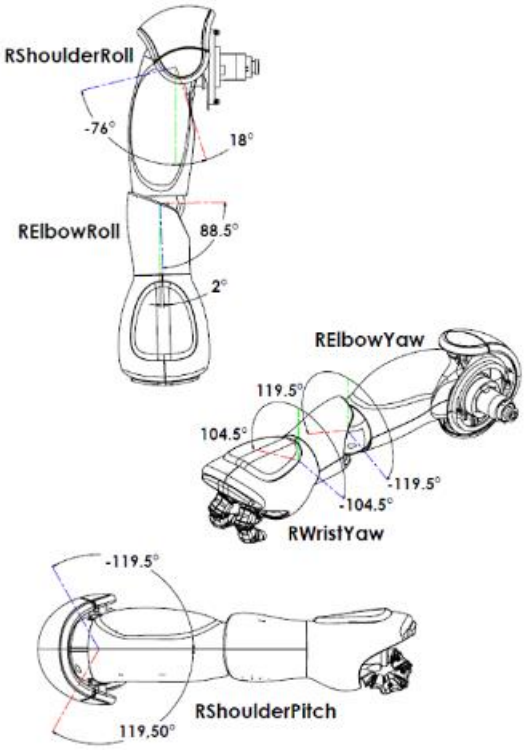
Figure 3.2 NAO Link Specifications



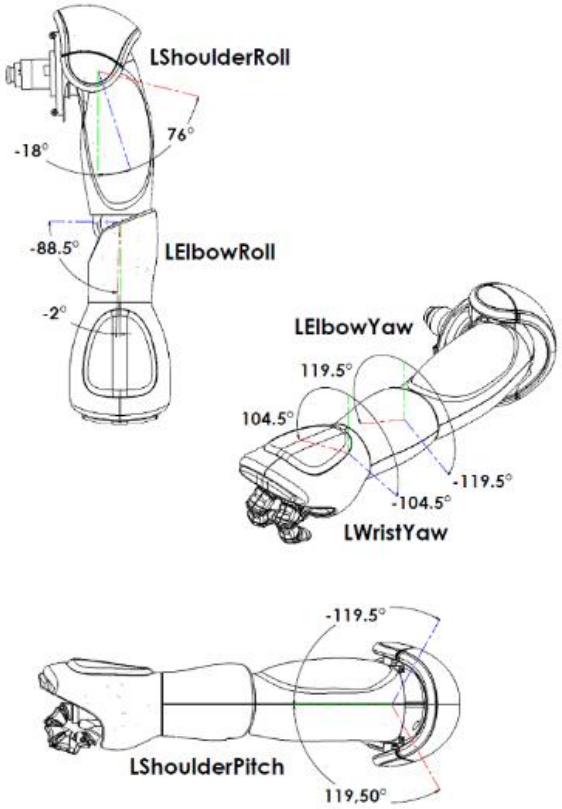
(a) Head



(b) Torso

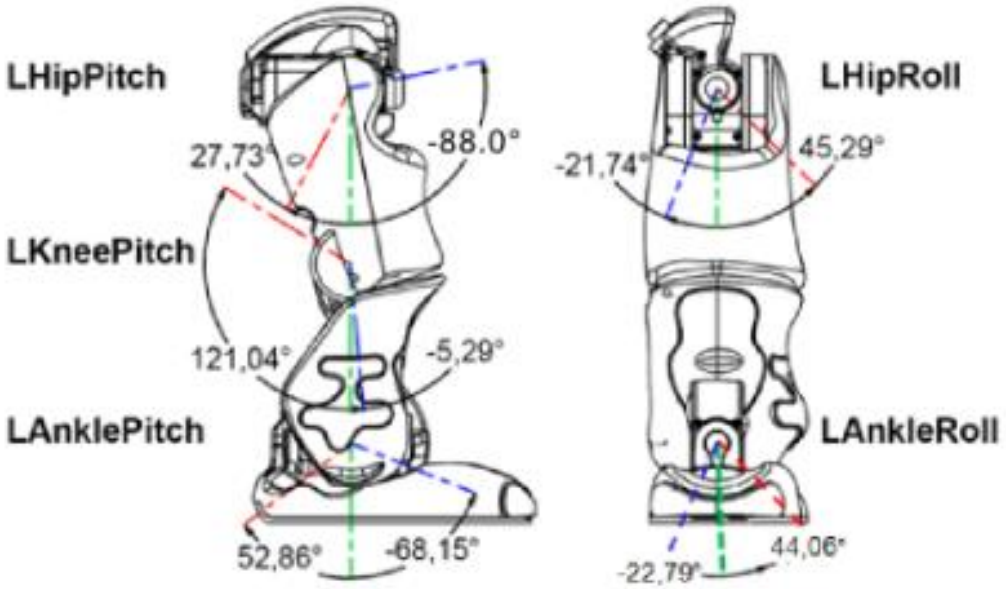


(c) Right Arm

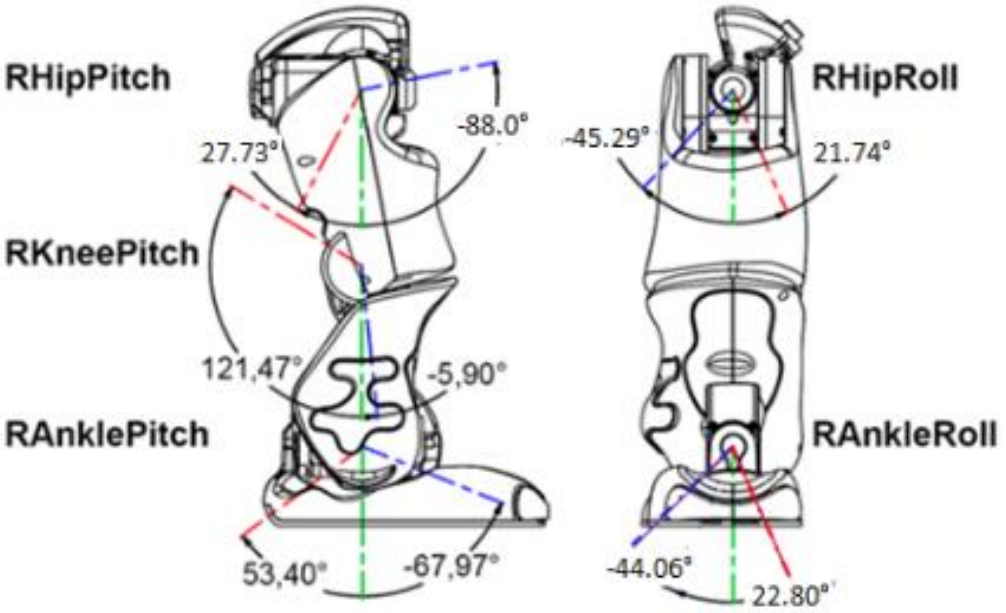


(d) Left Arm

Figure 3.3 Upper Body Joints Description



(a) Left Leg



(b) Right Leg

Figure 3.4 Lower Body Joints Description

Mass properties for NAO H25				
Frame of Reference	Mass (kg)	COMx (m)	COMy (m)	COMz (m)
Torso	1.0496	-0.00413	0	0.04342
HeadYaw	0.07842	-1e-5	0	-0.02742
HeadPitch	0.60533	-0.00112	0	0.05258
RShoulderPitch	0.09304	-0.00165	0.02663	0.00014
RShoulderRoll	0.15777	0.02455	-0.00563	0.0033
RElbowYaw	0.06483	-0.02744	0	-0.00014
RElbowRoll	0.07761	0.02556	-0.00281	0.00076
RWristYaw	0.18533	0.03434	0.00088	0.00308
LShoulderPitch	0.09304	-0.00165	-0.02663	0.00014
LShoulderRoll	0.15777	0.02455	0.00563	0.0033
LElbowYaw	0.06483	-0.02744	0	-0.00014
LElbowRoll	0.007761	0.02556	0.00281	0.00076
LWristYaw	0.18533	0.03434	-0.00088	0.00308
RHipYawPitch	0.06981	-0.00781	0.01114	0.02661
RHipRoll	0.14053	-0.01549	-0.00029	-0.00515
RHipPitch	0.38968	0.00138	-0.00221	-0.05373
RKneePitch	0.30142	0.00453	-0.00225	-0.04936
RAnklePitch	0.13416	0.00045	-0.00029	0.00685
RAnkleRoll	0.17184	0.02542	-0.0033	-0.03239
LHipYawPitch	0.06981	-0.00781	-0.01114	0.02661
LHipRoll	0.14053	-0.01549	0.00029	-0.00515
LHipPitch	0.38968	0.00138	0.00221	-0.05373
LKneePitch	0.30142	0.00453	0.00225	-0.04936
LAnklePitch	0.13416	0.00045	0.00029	0.00685
LAnkleRoll	0.17184	0.02542	0.0033	-0.03239

Table 3.1 Joint Masses and Center of Mass Coordinates

Inertial matrix properties for NAO H25						
Frame of Reference	Ixx	Iyy	Izz	Ixy	Ixz	Iyz
Torso	0.00506	0.00488	0.00161	1.43e-5	0.00015	-2.7e-5
HeadYaw	7.49e-5	7.59e-5	5.53e-6	1.57e-9	-1.83e-8	-5.29e-8
HeadPitch	0.00263	0.00249	0.000985	8.78e-6	4.09e-5	-2.99e-5
RShoulderPitch	8.428e-5	1.4155e-5	8.64e-5	2.028e-6	2.338e-8	1.971e-8
RShoulderRoll	0.00011	0.000367	0.000354	7.66e-5	-2.6e-5	1.209e-5
RElbowYaw	5.597e-6	7.54e-5	7.64e-5	4.209e-9	4.318e-8	-1.84e-9
RElbowRoll	2.539e-5	8.922e-5	8.72e-5	2.33e-6	-6.011e-7	2.69e-8
RWristYaw	7.05e-5	0.000356	0.000351	5.715e-6	-2.247e-5	3.177e-6
LShoulderPitch	8.428e-5	1.4155e-5	8.64e-5	-2.028e-6	2.338e-8	-1.971e-8
LShoulderRoll	9.389e-5	0.0003715	0.000341	-4.714e-5	-2.699e-5	-2.4599e-5
LElbowYaw	5.597e-6	7.54e-5	7.64e-5	4.209e-9	4.318e-8	-1.84e-9
LElbowRoll	2.5332e-5	8.913e-5	8.7287e-5	2.342e-6	7.458e-8	2.654e-8
LWristYaw	7.05e-5	0.000356	0.000351	5.715e-6	-2.247e-5	3.177e-6
RHipYawPitch	8.99e-5	0.0001055	6.6887e-	5.002e-6	1.273e-5	-2.77e-5
RHipRoll	2.7586e-5	9.826e-5	8.8103e-5	-1.919e-8	-4.108e-6	2.5099e-9
RHipPitch	0.001637	0.001592	0.000303	-8.395e-7	8.588e-5	-3.917e-5
RKneePitch	0.00118	0.001128	0.000191	-8.965e-7	2.799e-5	-3.84e-5
RAnklePitch	3.85e-5	7.43e-5	5.491e-5	6.433e-8	3.87e-6	-4.57e-9
RAnkleRoll	0.000269	0.0006434	0.000525	5.875e-6	0.000139	-1.884e-5
LHipYawPitch	8.15e-5	0.0001013	6.2623e-	-4.99e-6	1.2748e-5	2.345e-5
LHipRoll	2.7586e-5	9.826e-5	8.8099e-	-2.23e-8	-4.081e-6	-4.189e-9
LHipPitch	0.00163	0.00159	0.000303	9245e-7	8.53e-5	3.836e-5
LKneePitch	0.00118	0.001128	0.000191	6.336e-7	3.64e-5	-3.94e-5
LAnklePitch	3.85e-5	7.42e-5	5.486e-5	-2.634e-8	3.86e-6	1.833e-9
LAnkleRoll	0.000269	0.000644	0.000525	-5.69e-6	0.0001393	1.874e-5

$$[I_o(S)]_R = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix}_R (\mathrm{kg} * \mathrm{m}^2)$$

Table 3.2 Joint Inertia Matrices

3.2 Kinematic Model

The kinematic model of the robot is based on the five kinematic chains as described in section 3.1. As we need to solve the model both for dynamic stability and for kick motion generation, the center of mass positioning for each link becomes necessary. Therefore, we will consider the center of masses of each link to be the end effector while solving the kinematic chain. For all the kinematics calculations, the base pose is the 'StandZero' pose in which all the angles of NAO are at zero degrees and the base frame is the X-Y-Z coordinate frame called the 'TorsoFrame' as shown in Figure 3.5.

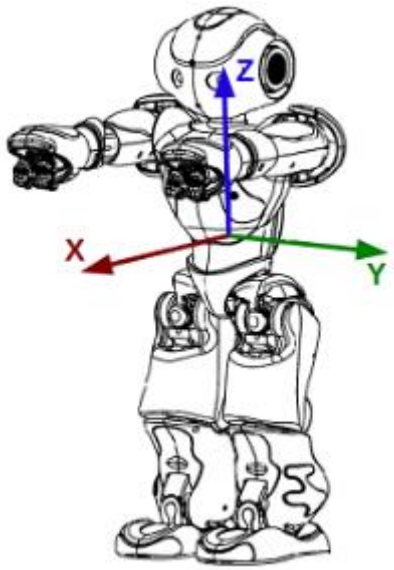


Figure 3.5 NAO in StandZero Pose

3.2.1 Head Kinematics:

The head is the simplest kinematic chain of the robot with only two joints; HeadPitch and HeadYaw. Keeping our base frame at the center of the NAO

torso where the “TorsoFrame" is defined, the kinematic chain for the head joints can be constructed in the form of Denavit-Hartenberg parameters as explained in the previous chapter. The DH parameters for the head chain are defined as follows:

Frame (T_i^{i-1})	a_{i-1}	α_{i-1}	$\mathbf{d}_i$	θ_i
HeadBasetoTorso (T_0^{Base})	0	0	NeckOffsetZ	0
HeadYawtoBase (T_1^0)	0	0	0	θ_1
HeadYawCOM (T_{1COM}^1)	$A_{(HY_{CX},HY_{CY},HY_{CZ})}$			
HeadPitchtoYaw (T_2^1)	0	$-\frac{\pi}{2}$	0	θ_2
HeadPitchCOM (T_{2COM}^2)	$R_x(\frac{\pi}{2})\ A_{(HP_{CX},HP_{CY},HP_{CZ})}$			

The resulting transformation matrices can then be used to find out the frame of end effector (Center of mass frames) in terms of the torso frame as following:

$$T_{1\text{COM}}^{\text{Base}} = T_0^{\text{Base}} T_1^0 T_{1\text{COM}}^1 = T_0^{\text{Base}} T_1^0 A_{(HY_{CX},HY_{CY},HY_{CZ})}$$

$$T_{2\text{COM}}^{\text{Base}} = T_0^{\text{Base}} T_1^0 T_2^1 R_x(\frac{\pi}{2})\ A_{(HP_{CX},HP_{CY},HP_{CZ})}$$

Where T_1^0 and T_2^1 are the transformation matrices of the joints (HeadYaw, HeadPitch). The terms $A_{(HY_{CX},HY_{CY},HY_{CZ})}$ and $A_{(HP_{CX},HP_{CY},HP_{CZ})}$ are the translation matrices defining the translation of the joint frame towards the center of mass based on the center of mass position vectors HY and HP.

3.2.2 Right Arm Kinematics:

The right arm chain consists of 5 joints therefore we need five sets of DH parameters to construct the whole kinematic chain. To define the chain starting from the torso frame, we first define the translation from the torso to the base of the right shoulder in terms of a translation matrix in the y and z directions. From the base towards the wrist we make following DH parameters to define the kinematic chain:

Frame (T_i^{i-1})	a_{i-1}	α_{i-1}	d_i	θ_i
ArmBasetoTorso (T_0^{Base})	A(0,-ShoulderOffsetY,ShoulderOffsetZ)			
RSPtoBase (T_1^0)	0	$-\frac{\pi}{2}$	0	θ_1
RSPCOM (T_{1COM}^1)	$R_x(\frac{\pi}{2})\, A_{(RSP_{CX},RSP_{CY},RSP_{CZ})}$			
RSRtoRSP (T_2^1)	0	$+\frac{\pi}{2}$	0	$\theta_2 + \frac{\pi}{2}$
RSRCOM (T_{2COM}^2)	$R_z(-\frac{\pi}{2})\, A_{(RSR_{CX},RSR_{CY},RSR_{CZ})}$			
REYtoRSR (T_3^2)	-ElbowOffsetY	$+\frac{\pi}{2}$	UpperArmLength	θ_3
REYCOM(T_{3COM}^3)	$R_y(-\frac{\pi}{2})\, R_x(-\frac{\pi}{2})\, A_{(REY_{CX},REY_{CY},REY_{CZ})}$			
RERtoREY (T_4^3)	0	$-\frac{\pi}{2}$	0	θ_4
RERCOM(T_{4COM}^4)	$R_Z(-\frac{\pi}{2}) A_{(RER_{CX},RER_{CY},RER_{CZ})}$			
RWYtoRER (T_5^4)	$R_Z\left(-\frac{\pi}{2}\right) A_{(LowerArmLength,0,0)}$			

Similar to the case in head chain, the right arm chains are then constructed for finding the center of mass frames in terms of the torso frames:

$$T_{1COM}^{Base} = T_0^{Base} T_1^0 T_{1COM}^1 \qquad T_{2COM}^{Base} = T_0^{Base} T_1^0 T_2^1 T_{2COM}^2$$

$$T_{3COM}^{Base} = T_0^{Base} T_1^0 T_2^1 T_3^2 T_{3COM}^3 \qquad T_{4COM}^{Base} = T_0^{Base} T_1^0 T_2^1 T_3^2 T_4^3 T_{4COM}^4$$

$$T_{5COM}^{Base} = T_0^{Base} T_1^0 T_2^1 T_3^2 T_4^3 T_5^4 A_{(RWY_{CX}, RWY_{CY}, RWY_{CZ})}$$

3.2.3 Left Arm Kinematics:

The left arm chain is nearly identical to the right arm chain with the joints LShoulderPitch, LShoulderRoll, LElbowYaw, LElbowRoll and LWristYaw. The base frame is again defined at the torso frame and we first define a translation from the torso to the base of the left shoulder in terms of a translation matrix. From the base of the left arm towards the wrist the following DH parameters are used to define the kinematic chain:

Frame (T_i^{i-1})	$\mathbf{a_{i-1}}$	$\boldsymbol{\alpha_{i-1}}$	$\mathbf{d_i}$	Θ_i
ArmBasetoTorso (T_0^{Base})	A(0,ShoulderOffsetY,ShoulderOffsetZ)			
LSPtoBase (T_1^0)	0	$-\frac{\pi}{2}$	0	Θ_1
LSPCOM (T_{1COM}^1)	$R_x(\frac{\pi}{2})\, A_{(LSP_{CX}, LSP_{CY}, LSP_{CZ})}$			
LSRtoLSP (T_2^1)	0	$+\frac{\pi}{2}$	0	$\Theta_2 + \frac{\pi}{2}$
LSRCOM (T_{2COM}^2)	$R_z(-\frac{\pi}{2})\, A_{(LSR_{CX}, LSR_{CY}, LSR_{CZ})}$			
LEYtoLSR (T_3^2)	ElbowOffsetY	$+\frac{\pi}{2}$	UpperArmLength	Θ_3
LEYCOM(T_{3COM}^3)	$R_y(-\frac{\pi}{2})\, R_x(-\frac{\pi}{2})\, A_{(LEY_{CX}, LEY_{CY}, LEY_{CZ})}$			
LERtoLEY (T_4^3)	0	$-\frac{\pi}{2}$	0	Θ_4

LERCOM(T^4_{4COM})	$R_Z(-\frac{\pi}{2})A_{(LER_{CX},LER_{CY},LER_{CZ})}$
LWYtoLER (T^4_5)	$R_Z\left(-\frac{\pi}{2}\right)A_{(LowerArmLength,0,0)}$

The center of masses of each link are then defined by the following recursive transformations:

$$T^{\text{Base}}_{1\text{COM}} = T^{\text{Base}}_0 T^0_1 T^1_{1\text{COM}} \qquad T^{\text{Base}}_{2\text{COM}} = T^{\text{Base}}_0 T^0_1 T^1_2 T^2_{2\text{COM}}$$

$$T^{\text{Base}}_{3\text{COM}} = T^{\text{Base}}_0 T^0_1 T^1_2 T^2_3 T^3_{3\text{COM}} \qquad T^{\text{Base}}_{4\text{COM}} = T^{\text{Base}}_0 T^0_1 T^1_2 T^2_3 T^3_4 T^4_{4\text{COM}}$$

$$T^{\text{Base}}_{5\text{COM}} = T^{\text{Base}}_0 T^0_1 T^1_2 T^2_3 T^3_4 T^4_5 A_{(RWY_{CX},RWY_{CY},RWY_{CZ})}$$

3.2.4 Right Leg Kinematics:

The kinematic chain for the right leg is the longest chain in the robot and has six joints. The DH parameters for the right leg joints are given below:

Frame (T^{i-1}_i)	a_{i-1}	α_{i-1}	d_i	θ_i
RLegBasetoTorso (T^{Base}_0)	A(o,-HipOffsetY,-HipOffsetZ)			
RHYPtoBase (T^0_1)	o	$-\frac{\pi}{4}$	o	$\theta_1 - \frac{\pi}{2}$
RHYPCOM (T^1_{1COM})	$R_y(\frac{\pi}{4})R_z(-\frac{\pi}{2})\,A_{(RSP_{CX},RSP_{CY},RSP_{CZ})}$			
RHRtoRHYP (T^1_2)	o	$-\frac{\pi}{2}$	o	$\theta_2 - \frac{\pi}{4}$
RHRCOM (T^2_{2COM})	$R_y(\frac{\pi}{2})\,R_z(\pi)A_{(RSR_{CX},RSR_{CY},RSR_{CZ})}$			
RHPtoRHR (T^2_3)	o	$+\frac{\pi}{2}$	o	$\theta_3 + \pi$
RHPCOM(T^3_{3COM})	$R_y(-\frac{\pi}{2})\,R_z(-\frac{\pi}{2})\,A_{(REY_{CX},REY_{CY},REY_{CZ})}$			
RKPtoRHP (T^3_4)	ThighLength	0	o	θ_4

RKPCOM(T_{4COM}^{4})	$R_y(-\frac{\pi}{2})\,R_z(-\frac{\pi}{2})A_{(RER_{CX},RER_{CY},RER_{CZ})}$			
RAPtoRKP (T_5^4)	TibiaLength	o	o	θ_5
RAPCOM (T_{5COM}^{5})	$R_y(-\frac{\pi}{2})\,R_z(-\frac{\pi}{2})A_{(RAP_{CX},RAP_{CY},RAP_{CZ})}$			
RARtoRAP (T_6^5)	o	$+\frac{\pi}{2}$	o	θ_6
RARCOM (T_{6COM}^{6})	$R_y(-\frac{\pi}{2})\,A_{(RAR_{CX},RAR_{CY},RAR_{CZ})}$			

The kinematic chains for each center of mass frame with respect to the torso frame is then found out by recursively multiplying the transformations defined above:

$$T_{1COM}^{Base} = T_0^{Base}T_1^0T_{1COM}^1 \qquad T_{2COM}^{Base} = T_0^{Base}T_1^0T_2^1T_{2COM}^2$$

$$T_{3COM}^{Base} = T_0^{Base}T_1^0T_2^1T_3^2T_{3COM}^3 \qquad T_{4COM}^{Base} = T_0^{Base}T_1^0T_2^1T_3^2T_4^3T_{4COM}^4$$

$$T_{5COM}^{Base} = T_0^{Base}T_1^0T_2^1T_3^2T_4^3T_5^4T_{5COM}^5 \qquad T_{6COM}^{Base} = T_0^{Base}T_1^0T_2^1T_3^2T_4^3T_5^4T_6^5T_{6COM}^6$$

3.2.5 Left Leg Kinematics:

Identical to the right leg, the left leg parameters are defined as:

Frame (T_i^{i-1})	a_{i-1}	α_{i-1}	d_i	θ_i
LLegBasetoTorso (T_0^{Base})	A(o,HipOffsetY,-HipOffsetZ)			
LHYPtoBase (T_1^0)	o	$\frac{\pi}{4}$	o	$\theta_1+\frac{\pi}{2}$
LHYPCOM (T_{1COM}^{1})	$R_y(\frac{\pi}{4})R_z(-\frac{\pi}{2})\,A_{(LSP_{CX},LSP_{CY},LSP_{CZ})}$			
LHRtoLHYP (T_2^1)	o	$\frac{\pi}{2}$	o	$\theta_2+\frac{\pi}{4}$
LHRCOM (T_{2COM}^{2})	$R_y(\frac{\pi}{2})\,R_z(\pi)A_{(LSR_{CX},LSR_{CY},LSR_{CZ})}$			

LHPtoLHR (T_3^2)	o	$+\frac{\pi}{2}$	o	$\Theta_3 + \pi$
LHPCOM(T_{3COM}^3)	$R_y(-\frac{\pi}{2})\, R_z(-\frac{\pi}{2})\, A_{(LEY_{CX}, LEY_{CY}, LEY_{CZ})}$			
LKPtoLHP (T_4^3)	ThighLength	0	o	Θ_4
LKPCOM(T_{4COM}^4)	$R_y(-\frac{\pi}{2})\, R_z(-\frac{\pi}{2}) A_{(LER_{CX}, LER_{CY}, LER_{CZ})}$			
LAPtoLKP (T_5^4)	TibiaLength	o	o	Θ_5
LAPCOM (T_{5COM}^5)	$R_y(-\frac{\pi}{2})\, R_z(-\frac{\pi}{2}) A_{(LAP_{CX}, LAP_{CY}, LAP_{CZ})}$			
LARtoLAP (T_6^5)	o	$+\frac{\pi}{2}$	o	Θ_6
LARCOM (T_{6COM}^6)	$R_y(-\frac{\pi}{2})\, A_{(LAR_{CX}, LAR_{CY}, LAR_{CZ})}$			

Similar to other cases, the center of mass for each link is then defined in terms of the torso frame as following:

$$T_{1\mathrm{COM}}^{\mathrm{Base}} = T_0^{\mathrm{Base}} T_1^0 T_{1\mathrm{COM}}^1 \qquad T_{2\mathrm{COM}}^{\mathrm{Base}} = T_0^{\mathrm{Base}} T_1^0 T_2^1 T_{2\mathrm{COM}}^2$$

$$T_{3\mathrm{COM}}^{\mathrm{Base}} = T_0^{\mathrm{Base}} T_1^0 T_2^1 T_3^2 T_{3\mathrm{COM}}^3 \qquad T_{4\mathrm{COM}}^{\mathrm{Base}} = T_0^{\mathrm{Base}} T_1^0 T_2^1 T_3^2 T_4^3 T_{4\mathrm{COM}}^4$$

$$T_{5\mathrm{COM}}^{\mathrm{Base}} = T_0^{\mathrm{Base}} T_1^0 T_2^1 T_3^2 T_4^3 T_5^4 T_{5\mathrm{COM}}^5 \qquad T_{6\mathrm{COM}}^{\mathrm{Base}} = T_0^{\mathrm{Base}} T_1^0 T_2^1 T_3^2 T_4^3 T_5^4 T_6^5 T_{6\mathrm{COM}}^6$$

3.2.6 Inverse Kinematics Solver

To solve the inverse kinematics problem, we need to find the solution to the equation:

$$\Delta\Theta = J^{-1}\Delta x$$

Where Θ represents the joint space coordinates while x represents the Cartesian coordinates.

As the Jacobian J, in our case is a 5×6 matrix, it cannot be inverted through normal means. As we cannot find its inverse, we will find a pseudo-inverse instead. That is, we set

$$\Delta\Theta = J^{+}e$$

Where the n × m matrix J^{+} is the pseudo inverse of J, also called the Moore-Penrose inverse of J. It is defined as,

$$J^{+} = J^{T}(JJ^{T})^{-1}$$

Now as the matrix JJ^{T} is always invertible, the pseudo-inverse of the Jacobian can be easily found but it is only valid for a small change e thus we need to reach the final position iteratively using the following equation,

$$\Theta_{i+1} = J^{+}e + \Theta_{i}$$

To avoid invalid configurations, maximum ranges of the joint space coordiantes are checked in each iteration and if the max value is reached, that particular coordinate is set to max.

3.3 Dynamics and Control

The dynamics of the robot has been solved twice, once for the whole body and once for the legs. The dynamic model for the legs is found by using the langrangian formulation and in this case only the kinetic energy matrix or mass matrix is considered so that the effective mass can be found for the impact. Whereas, the whole body dynamics model is found by using the newton-euler formulation and it is generally focused on the determination of zero-moment point for robot stability during the motion generation. The newton-euler approach is preferred in this case because of its computational ease and simplicity for solving the dynamics for a large number of DOFs. Before we move on to the detailed analysis of the robot dynamics, we first need to define the inertial parameters with respect to the required frames.

3.3.1 Inertial Parameters:

The inertial parameters consist of mass and moment of inertia of each link. When solving the dynamics of the robot, it is necessary to find the inertial matrix at the center of mass of each link. The inertial matrix properties (I_{xx}, I_{yy}, I_{zz}, I_{xy}, I_{zy}, I_{xz}) in section 3.1 are defined with respect to the base frame of each joint. For the dynamic analysis, we need to transformation the inertia matrix of each link to its center of mass. The necessary transformations for the inertial matrices of each kinematic chain are defined below.

3.3.1.1 Head:

The head chain has two inertial frames, therefore we need two transformations. First, we need to rotate the inertial matrix from its frame

defined in section 3.1 to our joint coordinate frame. The frame of the first joint collides with the base frame, therefore it does not require a rotation. The rotation for the second joint (HeadPitch) is given as:

$$I_{HP}{}^{Joint} = R_x(\frac{\pi}{2})\ I_{HP}{}^{Base} R_x(\frac{\pi}{2})^T$$

The resulting inertial matrices in our coordinate frames are then translated to the center of mass of the link by using the 3-dimensional parallel-axis theorem. The inertial matrices at the center of masses are given as:

$$I_{HY}{}^{COM} = I_{HY}{}^{Joint} - mass_{HY}\left[Pcom_{HY}{}^{T} Pcom_{HY} Id(3) - Pcom_{HY} Pcom_{HY}{}^{T}\right]$$

$$I_{HP}{}^{COM} = I_{HP}{}^{Joint} - mass_{HP}\left[Pcom_{HP}{}^{T} Pcom_{HP} Id(3) - Pcom_{HP} Pcom_{HP}{}^{T}\right]$$

Where $Pcom_j$ is the position vector to the center of mass of the joint 'j' and Id(3) is a 3x3 identity matrix.

3.3.1.2 Arms:

Similar to the head chain, the transformations are defined for the right and left arms. The rotations in this case are identical for both arms.

Rotations:

$$I_{R/LSP}{}^{Joint} = R_x\left(\frac{\pi}{2}\right)\ I_{R/LSP}{}^{Base} R_x\left(\frac{\pi}{2}\right)^T$$

$$I_{R/LSR}{}^{Joint} = R_z(-\frac{\pi}{2})\ I_{R/LSR}{}^{Base} R_z(-\frac{\pi}{2})^T$$

$$I_{R/LEY}{}^{Joint} = R_y\left(-\frac{\pi}{2}\right) R_x\left(-\frac{\pi}{2}\right)\ I_{R/LEY}{}^{Base}\left[R_y\left(-\frac{\pi}{2}\right) R_x\left(-\frac{\pi}{2}\right)\right]^T$$

$$I_{R/LER}{}^{Joint} = R_y(-\frac{\pi}{2})\ R_z(-\frac{\pi}{2})\ I_{R/LER}{}^{Base}\left[R_y(-\frac{\pi}{2})\ R_z(-\frac{\pi}{2})\right]^T$$

$$I_{R/LWY}{}^{Joint} = I_{R/LWY}{}^{Base}$$

Translations:

The inertial matrix of the link '*i*' is then translated to the center of mass as following:

$$I_i{}^{COM} = I_i{}^{Joint} - mass_i\left[Pcom_i{}^T Pcom_i Id(3) - Pcom_i Pcom_i{}^T\right]$$

3.3.1.3 Legs:

The rotations for the base frames to our joint coordinate frames for both legs are also identical.

Rotations:

$$I_{L/RHYP}{}^{Joint} = R_y\left(\frac{\pi}{4}\right) R_z\left(-\frac{\pi}{2}\right)\ I_{L/RHYP}{}^{Base}\left[R_y\left(\frac{\pi}{4}\right)\ R_z\left(-\frac{\pi}{2}\right)\right]^T$$

$$I_{L/RHR}{}^{Joint} = R_y\left(\frac{\pi}{2}\right) R_z\left(\frac{\pi}{2}\right)\ I_{L/RHR}{}^{Base}\left[R_y\left(\frac{\pi}{2}\right)\ R_z\left(\frac{\pi}{2}\right)\right]^T$$

$$I_{L/RHP}{}^{Joint} = R_y\left(-\frac{\pi}{2}\right) R_z\left(-\frac{\pi}{2}\right)\ I_{L/RHP}{}^{Base}\left[R_y\left(-\frac{\pi}{2}\right)\ R_z\left(-\frac{\pi}{2}\right)\right]^T$$

$$I_{L/RKP}{}^{Joint} = R_y\left(-\frac{\pi}{2}\right) R_z\left(-\frac{\pi}{2}\right)\ I_{L/RKP}{}^{Base}\left[R_y\left(-\frac{\pi}{2}\right)\ R_z\left(-\frac{\pi}{2}\right)\right]^T$$

$$I_{L/RAP}{}^{Joint} = R_y\left(-\frac{\pi}{2}\right) R_z\left(-\frac{\pi}{2}\right) I_{L/RAP}{}^{Base} \left[R_y\left(-\frac{\pi}{2}\right) R_z\left(-\frac{\pi}{2}\right)\right]^T$$

$$I_{L/RAR}{}^{Joint} = R_y\left(-\frac{\pi}{2}\right) I_{L/RAR}{}^{Base} \left[R_y\left(-\frac{\pi}{2}\right)\right]^T$$

Translations:

Similarly, the translation for inertia matrix of link '*i*' of each leg to the center of mass is given by:

$$I_i{}^{COM} = I_i{}^{Joint} - mass_i \left[Pcom_i{}^T Pcom_i Id(3) - Pcom_i Pcom_i{}^T\right]$$

3.3.2 Leg Dynamics:

As mentioned above, the necessary dynamics of the leg have been solved by the Langrangian approach. In this section, the necessary steps and calculations for the finding the mass matrix of the NAO robot legs will be shown. As described in chapter 2, the mass matrix of an articulated robot manipulator is given by:

$$M(\Theta) = \sum_i^n (m_{ci} J_{vi}{}^T J_{vi} + J_{wi}{}^T I_{ci} J_{wi})$$

Therefore, we need to find the center of mass Jacobians J_v and J_w, the mass and the inertial matrix defined at the center of mass for each link '*i*'. The masses and the inertial matrices have already been defined at the center of mass of each link in the previous section, therefore, we now need to find the linear and angular velocity Jacobians from the robot kinematic equations. As

the two legs are identical, let us consider one of the two legs, say the right leg. For the sake of simplicity, we will not consider the HipYawPitch joint in our dynamic model because of its complexity and shared mechanism between the legs. The joints considered in the analysis are RhipRoll, RHipPitch, RKneePitch, RAnklePitch and RAnkleRoll. We first find the positions of the center of masses of each link with respect to the base of the right leg using the forward kinematics as described in section 3.2. The transformations defining the center of masses of links 1-5 are given below:

$$T_{2\mathrm{COM}}^{\mathrm{Base}} = T_0^{\mathrm{Base}} T_1^0 T_2^1 T_{2\mathrm{COM}}^2$$

$$T_{3\mathrm{COM}}^{\mathrm{Base}} = T_0^{\mathrm{Base}} T_1^0 T_2^1 T_3^2 T_{3\mathrm{COM}}^3$$

$$T_{4\mathrm{COM}}^{\mathrm{Base}} = T_0^{\mathrm{Base}} T_1^0 T_2^1 T_3^2 T_4^3 T_{4\mathrm{COM}}^4$$

$$T_{5\mathrm{COM}}^{\mathrm{Base}} = T_0^{\mathrm{Base}} T_1^0 T_2^1 T_3^2 T_4^3 T_5^4 T_{5\mathrm{COM}}^5$$

$$T_{6\mathrm{COM}}^{\mathrm{Base}} = T_0^{\mathrm{Base}} T_1^0 T_2^1 T_3^2 T_4^3 T_5^4 T_6^5 T_{6\mathrm{COM}}^6$$

Where the position of each center of mass is defined by the fourth column of the transformation matrix. Let each transformation matrix T be defined by the notation:

$$T(\,[rows]x[columns]\,)$$

Then the linear and angular jacobians for each link can be defined as:

$$J_{v2} = \begin{bmatrix} \frac{\partial T_{2COM}^{Base}([1,2,3],4)}{\Theta_2} & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$J_{w2} = \begin{bmatrix} T_{2COM}^{Base}([1,2,3],3) & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$J_{v3} = \begin{bmatrix} \frac{\partial T_{3COM}^{Base}([1,2,3],4)}{\Theta_2} & \frac{\partial T_{3COM}^{Base}([1,2,3],4)}{\Theta_3} & 0 & 0 & 0 \end{bmatrix}$$

$$J_{w3} = \begin{bmatrix} T_{2COM}^{Base}([1,2,3],3) & T_{3COM}^{Base}([1,2,3],3) & 0 & 0 & 0 \end{bmatrix}$$

$$J_{v4} = \begin{bmatrix} \frac{\partial T_{4COM}^{Base}([1,2,3],4)}{\Theta_2} & \frac{\partial T_{4COM}^{Base}([1,2,3],4)}{\Theta_3} & \frac{\partial T_{4COM}^{Base}([1,2,3],4)}{\Theta_4} & 0 & 0 \end{bmatrix}$$

$$J_{w4} = \begin{bmatrix} T_{2COM}^{Base}([1,2,3],3) & T_{3COM}^{Base}([1,2,3],3) & T_{4COM}^{Base}([1,2,3],3) & 0 & 0 \end{bmatrix}$$

$$J_{v5} = \begin{bmatrix} \frac{\partial T_{5COM}^{Base}([1,2,3],4)}{\Theta_2} & \frac{\partial T_{5COM}^{Base}([1,2,3],4)}{\Theta_3} & \frac{\partial T_{5COM}^{Base}([1,2,3],4)}{\Theta_4} & \frac{\partial T_{5COM}^{Base}([1,2,3],4)}{\Theta_5} & 0 \end{bmatrix}$$

$$J_{w5} = \begin{bmatrix} T_{2COM}^{Base}([1,2,3],3) & T_{3COM}^{Base}([1,2,3],3) & T_{4COM}^{Base}([1,2,3],3) & T_{5COM}^{Base}([1,2,3],3) & 0 \end{bmatrix}$$

$$J_{v6} = \begin{bmatrix} \frac{\partial T_{6COM}^{Base}([1,2,3],4)}{\Theta_2} & \frac{\partial T_{6COM}^{Base}([1,2,3],4)}{\Theta_3} & \frac{\partial T_{6COM}^{Base}([1,2,3],4)}{\Theta_4} & \frac{\partial T_{6COM}^{Base}([1,2,3],4)}{\Theta_5} & \frac{\partial T_{6COM}^{Base}([1,2,3],4)}{\Theta_6} \end{bmatrix}$$

$$J_{w6} = \begin{bmatrix} T_{2COM}^{Base}([1,2,3],3) & T_{3COM}^{Base}([1,2,3],3) & T_{4COM}^{Base}([1,2,3],3) & T_{5COM}^{Base}([1,2,3],3) & T_{6COM}^{Base}([1,2,3],3) \end{bmatrix}$$

The mass matrix for the link 'i' is then defined as:

$$M_i(\Theta) = m_{ci} {J_{vi}}^T J_{vi} + {J_{wi}}^T I_{ci} J_{wi}$$

Finally the total kinetic energy matrix is found to be:

$$M(\Theta) = \sum_{i}^{n} M_i(\Theta)$$

3.3.3 Virtual Mass Analysis

To find the virtual mass at the end effector point EE (Figure 3),, we first need to find the inverse inertia matrix at that point. As we are only concerned with the linear velocity of the foot, therefore we will use the linear jacobian of the end effector point given by:

$$J_{vEE} = \left[\frac{\partial T_{EE}^{Base}([1,2,3],4)}{\Theta_2} \quad \frac{\partial T_{EE}^{Base}([1,2,3],4)}{\Theta_3} \quad \frac{\partial T_{EE}^{Base}([1,2,3],4)}{\Theta_4} \quad \frac{\partial T_{EE}^{Base}([1,2,3],4)}{\Theta_5} \quad \frac{\partial T_{EE}^{Base}([1,2,3],4)}{\Theta_6}\right]$$

Then, the cartesian space inertia matrix is given by:

$$M(\Theta)_{EE} = {J_{vEE}}^{-T}(\Theta)\, M(\Theta)\, {J_{vEE}}^{-1}(\Theta)$$

The pseudo kinetic energy matrix is then defined by:

$$M(\Theta)^{-1}{}_{EE} = J_{vEE}(\Theta)\, M(\Theta)^{-1}\, J_{vEE}(\Theta)^T$$

Finally, the effective mass of the end effector is found in the direction specified by the desired unit vector $\boldsymbol{u}$,

$$\frac{1}{m_{eff}} = \boldsymbol{u}^T M(\Theta)^{-1}{}_{EE}\boldsymbol{u} \qquad m_{eff} = \frac{1}{\boldsymbol{u}^T M(\Theta)^{-1}{}_{EE}\boldsymbol{u}}$$

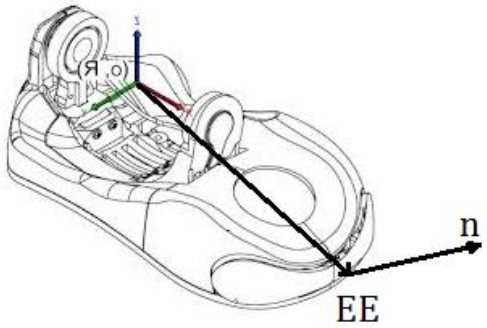


Figure 3.6 Virtual Mass Analysis

3.3.4 Whole Body Dynamics:

The recursive Newton-Euler algorithm (RNEA) is generally preferred to get the direction solution to the desired forces/torques instead of the creation of explicit equations to find the dynamic model. Thus, the direct solution requires the knowledge of joint positions, velocities and accelerations at the desired time and results in the values of forces and torques acting on point on the body given that the transformation to that point is available. Our goal here is to model the whole robot so that we can find the forces and torques at the support foot of the robot when its other leg is in motion. Thus, having prior knowledge about the torques and forces at the support foot, we can find the projected movement of the zero-moment point during the motion generation.

3.3.4.1 Analysis:

To solve the dynamics, we first assumed that the whole body is attached at the support foot ankle which is acting as a fixed support. From that point onwards, a new kinematic chain was constructed which started from the ankle and reached the torso as the final end effector. The force/moment effects applied by head, left arm, right arm and kicking leg chains on the torso are found separately by considering each chain as a fixed end manipulator. The forward and backward recursion gives a resultant force 'f_i' and moment 'm_i' for each chain 'i'. These forces/moment effects are then superimposed on the torso chain as external forces/moments. The new kinematic chain that was constructed for the two legs is defined by the joints AnkleRoll, AnklePitch, KneePitch, HipPitch, HipRoll, HipYawPitch in the respective

order. The DH parameters and the center of mass transformations for the support leg chain are:

Frame (T_i^{i-1})	a_{i-1}	α_{i-1}	d_i	θ_i
Rotation	$R_y(\frac{\pi}{2})$			
ARtoAP (T_1^0)	o	0	o	θ_1
APCOM (T_{1COM}^1)	$R_y(-\frac{\pi}{2})\, A_{(AP_{CX},AP_{CY},AP_{CZ})}$			
KPtoAP (T_2^1)	o	$\frac{\pi}{2}$	o	θ_2
KPCOM (T_{2COM}^2)	$R_y(-\frac{\pi}{2})\, R_z(\frac{\pi}{2})\, A_{(KP_{CX},KP_{CY},KP_{CZ})}$			
HPtoKP (T_3^2)	-TibiaLength	o	o	θ_3
HPCOM(T_{3COM}^3)	$R_y(-\frac{\pi}{2})\, R_z(\frac{\pi}{2})\, A_{(HP_{CX},HP_{CY},HP_{CZ})}$			
HRtoHP (T_4^3)	-ThighLength	0	o	θ_4
HRCOM(T_{4COM}^4)	$R_y(-\frac{\pi}{2})\, R_z(\frac{\pi}{2})\, A_{(HR_{CX},HR_{CY},HR_{CZ})}$			
HYPtoHR (T_5^4)	o	$-\frac{\pi}{2}$	o	$\theta_5+\frac{\pi}{4}$
HYPCOM (T_{5COM}^5)	$R_z(-\frac{\pi}{4})\, R_y(-\frac{\pi}{2})\, A_{(HYP_{CX},HYP_{CY},HYP_{CZ})}$			
TorsotoHYP (T_6^5)	o	$-\frac{\pi}{2}$	o	θ_6
TorsoCOM (T_{6COM}^6)	$R_y(-\frac{\pi}{4})\, R_z(-\frac{\pi}{2})\, A_{(Torso_{CX},Torso_{CY},Torso_{CZ})}$			

Finally, this support leg kinematic chain is then solved by RNEA, first by forward recursion to find the velocities and accelerations and then by the backward recursion with external forces/moments on the torso included from the previous analysis. At the end we find the total resultant force and moment applied at the ankle given by F_{Ankle} and M_{Ankle}.

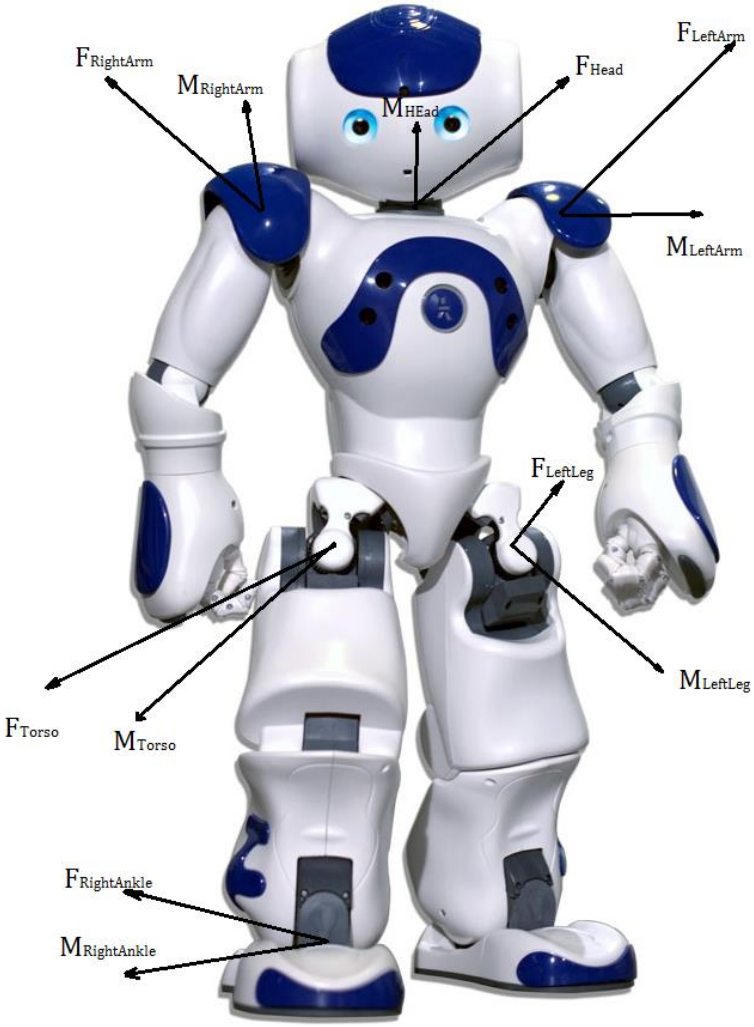


Figure 3.7 Whole Body Dynamics Solution

3.3.4.2 ZMP Calculation:

Substituting the resultant force F_{Ankle} and moment M_{Ankle} in the equations defined in section 2.5.3, the reaction force is found by

$$R + F_{Ankle} + m_{ankle}g = 0$$

And the ZMP x and y coordinates are given by:

$$OP \times R + OG \times m_{foot}g + M_{Ankle} + OA \times F_A = 0$$

3.3.5 ZMP Controller Design

For the ZMP controller design, the inverted cart-table model was chosen according to which the zero-moment point for the humanoid can be expressed for both x and y directions in terms of the its center of mass as following:

$$P_x = x - \frac{H}{g}\ddot{x}$$

$$P_y = y - \frac{H}{g}\ddot{y}$$

where $\ddot{x}$ and $\ddot{y}$ are the center of mass accelerations in x and y directions, H is the height of the center of mass which is kept constant at the starting height when the controller is initiate and g is the gravitational acceleration.

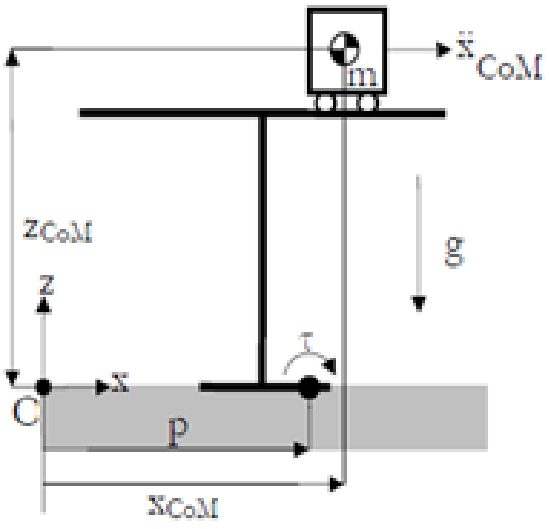


Figure 3.8 Inverted Cart-table Model

As the two equations are same for both directions, a controller with same parameters can be used. Therefore, the specifications of the controller will now be described with respect to x direction. The above mentioned equation can be described in the form of a state-space model,

$$\frac{d}{dt}\begin{pmatrix} x \\ \dot{x} \\ \ddot{x} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} x \\ \dot{x} \\ \ddot{x} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} u$$

$$P = \begin{pmatrix} 1 & 0 & -H/g \end{pmatrix}\begin{pmatrix} x \\ \dot{x} \\ \ddot{x} \end{pmatrix}$$

As the robot operations are carried out in discrete space in real-time, a digital controller model can be described in terms of the sampling rate '*T*',

$$x(k+1) = AX(k) + Bu(k)$$

$$P(k) = CX(k)$$

$$A = \begin{pmatrix} 1 & t & t^2/2 \\ 0 & 1 & t \\ 0 & 0 & 1 \end{pmatrix} \qquad B = \begin{pmatrix} t^3/6 \\ t^2/2 \\ t \end{pmatrix} \qquad C = \begin{pmatrix} 1 & 0 & -H/g \end{pmatrix}$$

By assuming the incremental state $\Delta X(k) = X(k) - X(k-1)$, the state can be augmented as $X_{Aug}(k) = \begin{bmatrix} p(k) \\ \Delta X(k) \end{bmatrix}$, and therefore the above equation can be rewritten as,

$$X_{Aug}(k+1) = A_{Aug}X_{Aug}(k) + B_{Aug}u(k)$$

$$P(k) = C_{Aug}X_{Aug}(k)$$

Where

$$A_{Aug} = \begin{bmatrix} 1 & CA \\ 0 & A \end{bmatrix} \qquad B_{Aug} = \begin{bmatrix} CB \\ B \end{bmatrix} \qquad C_{Aug} = [1\ 0\ 0\ 0]$$

In this case, the control input u can be define as,

$$u(k) = -G_i \sum_{i=0}^{k} e(i) - G_x x(k) - \sum_{j=1}^{N} G_p P_{des}(k+j)$$

Here, G_i and G_x are ZMP tracking error gain and the state feedback gain respectively. The ZMP tracking error is $e_i = P - P_{des}$ and k is the k_{th} sample time. The third term consists of the planned ZMP reference trajectory up to N samples in future. Since this controller uses future information, it is called a preview controller and the gain G_p is called the preview gain. The parameter N can be calculated based on the incremental time step as,

$$N = 1/Sampling\ rate = 1/t$$

The gain parameters G_i and G_x must be found using the discrete algebraic Riccati equation,

$$P = A^T PA - A^T PB(R + B^T PB)\, B^T PA + Q$$

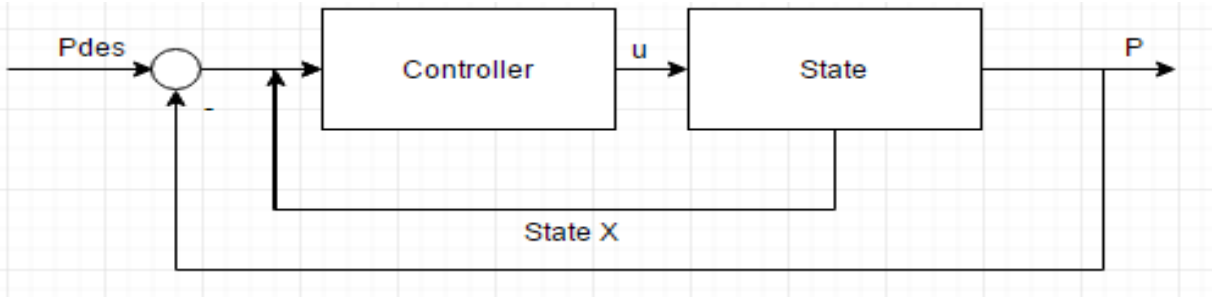


Figure 3.9 Preview Control Block Diagram

3.5 C++ Algorithm:

The C++ algorithm designed for the whole kick module is explained in the following sections:

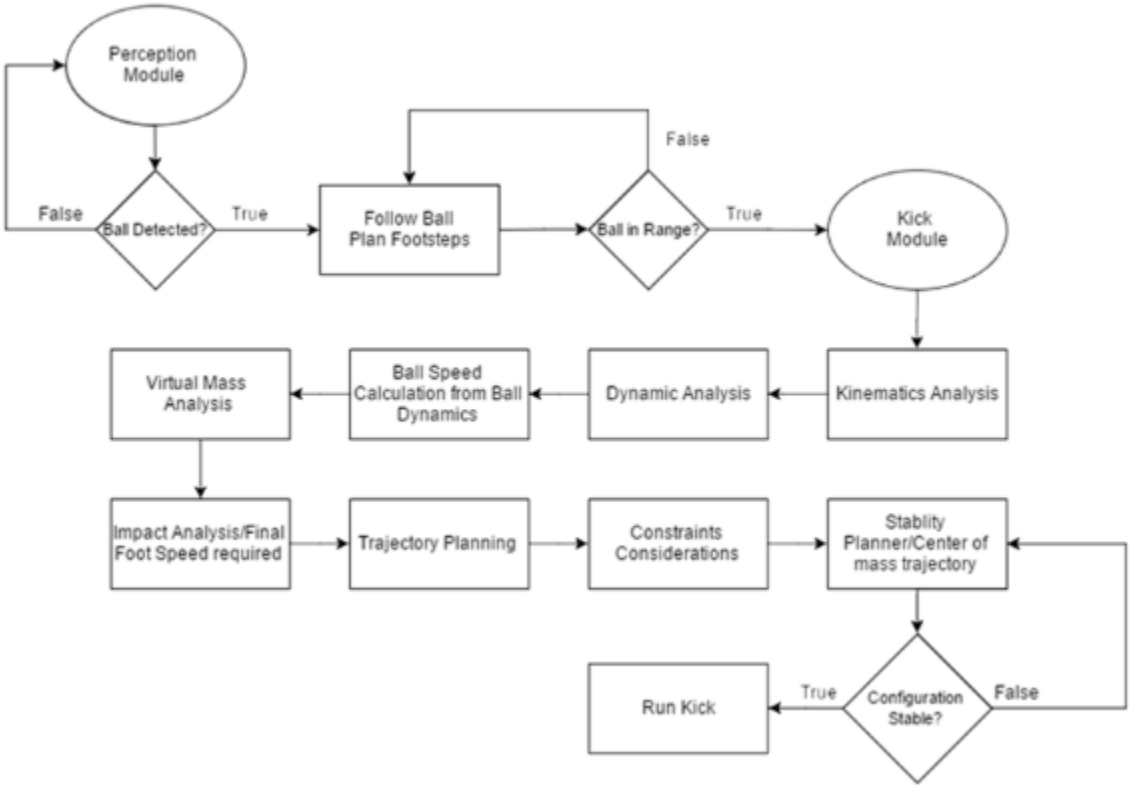


Figure 3.10 Schematic of Kick Algorithm integrated with RoboCup Team Code

3.5.1 Motion Planner:

The kick module starts with the motion planner, which takes the input regarding the position of the ball and the goal in terms of their x-y coordinates. Depending upon the position of the ball, the planner checks whether the ball is within the range to kick. The total kicking range is manually defined as shown in the figure:

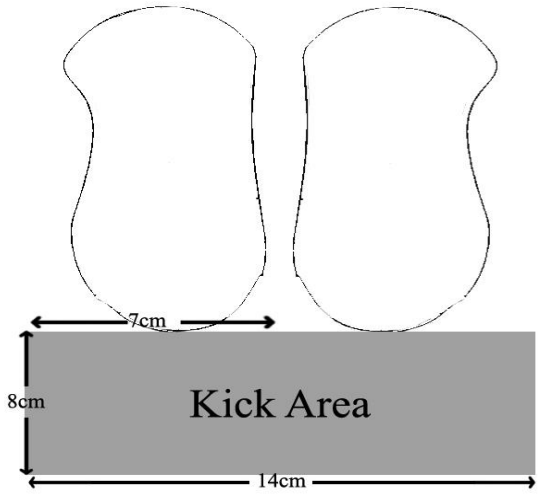


Figure 3.11 Optimum kicking range

If the ball is in the range, the planner then selects the kicking leg depending upon the ball position and goal direction. The kick selecting criteria is also manually defined by taking into account the reachable space of each leg and if the kick is infeasible, the kick module stops and returns to the footsteps planning. Once the kick leg is selected, the planner moves onto finding the end effector point on the foot.

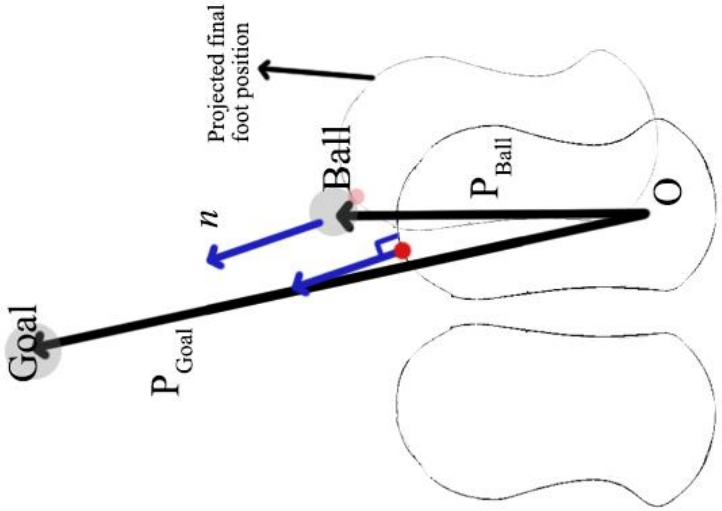


Figure 3.12 Motion planning

Let the position vectors for the ball and the goal be '$\vec{\boldsymbol{P}}_{\boldsymbol{Ball}}$' and '$\vec{\boldsymbol{P}}_{\boldsymbol{Goal}}$'. Based on the position vectors '$\vec{\boldsymbol{P}}_{\boldsymbol{Ball}}$' and '$\vec{\boldsymbol{P}}_{\boldsymbol{Goal}}$' the distance '$S$' and the direction unit vector '$\hat{\boldsymbol{n}}$' are found as follows:

$$S = \left|\vec{\boldsymbol{P}}_{\boldsymbol{Goal}} - \vec{\boldsymbol{P}}_{\boldsymbol{Ball}}\right|$$

$$\hat{\boldsymbol{n}} = (\vec{\boldsymbol{P}}_{\boldsymbol{Goal}} - \vec{\boldsymbol{P}}_{\boldsymbol{Ball}}) / \left|\vec{\boldsymbol{P}}_{\boldsymbol{Goal}} - \vec{\boldsymbol{P}}_{\boldsymbol{Ball}}\right|$$

The unit vector '$\hat{\boldsymbol{n}}$' is used to find the point on the foot contour which is perpendicular to the direction. This is achieved by finding the solution to the equation:

$$\hat{\boldsymbol{n}}.\, C_{\,foot} = 0$$

Where $C_{\,foot}$ is the foot contour defined by two continuous bezier curves as shown (Figure 3.12).

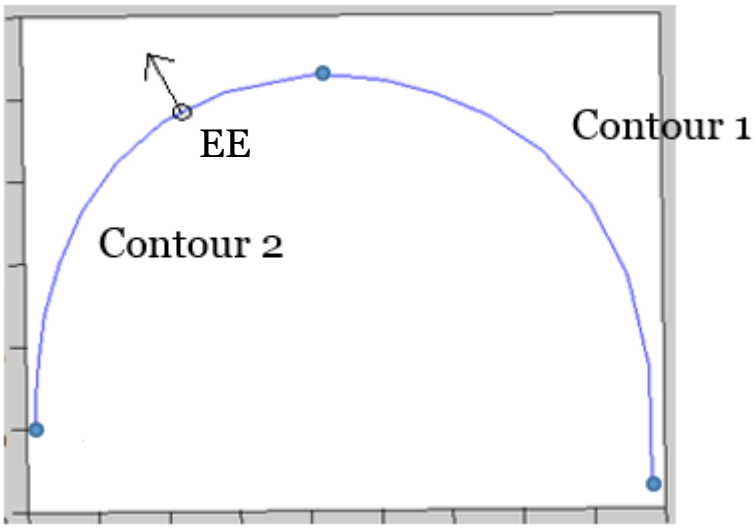


Figure 3.13 Foot Contour

The resulting point on the foot contour is chosen as the final end effector 'EE' which is supposed to have the impact with the ball.

3.5.2 Ball Dynamics:

The ball dynamics and friction analysis is required for the calculation of desired ball velocity for the required distance. The ball during the motion goes through two phases, sliding and rolling (Figure 3.13).

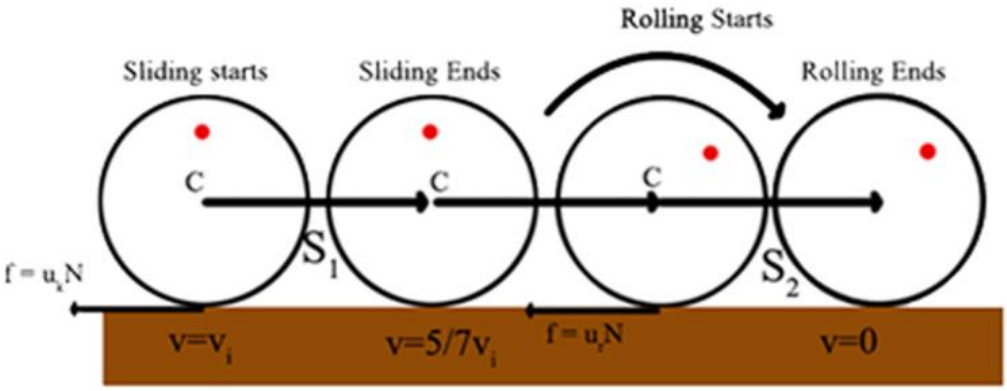


Figure 3.14 Ball Motion Phases

During the sliding motion, ball is affected by kinetic friction 'u_k' and the force of friction is given by:

$$F_k = u_k m g$$

Which can be related with ball velocity 'v' as:

$$F_k = m\left(\frac{dv}{dt}\right) - (1)$$

The moment applied on the ball by the friction force is:

$$M = I\alpha$$

In our case, $I = \frac{2}{5}mR^2$ for the spherical ball and 'α' the angular acceleration can be written in the form of rate of change of angular velocity 'w'. The above equation can then be written as:

$$F_k R = \frac{2}{5}mR^2\left(\frac{dw}{dt}\right)$$

Or $$F_k = \frac{2}{5}mR\left(\frac{dw}{dt}\right) - (2)$$

Combining (1) and (2):

$$\left(\frac{dv}{dt}\right) = \frac{2R}{5}\left(\frac{dw}{dt}\right)$$

Integrating the above equation and considering the fact that the increase in angular velocity is caused by the decrease in linear velocity of the ball:

$$v_f - v_i = -\frac{2R}{5}(w_f - w_i)$$

This equation defines the relationship between the two phases the ball undergoes; the sliding phase with linear velocity and the rolling phase with angular velocity. In our case, the ball starts with an initial velocity 'v_i' and ends with a rolling velocity v_r after which rolling phase starts. Using the relation $v_r = wr$:

$$v_r - v_i = -\frac{2R}{5}\left(\frac{v_r}{R} - 0\right) - (3)$$

$$v_r = \frac{5}{7}v_i$$

Thus the ball starts with v_i and moves against the force F_k until it starts to roll where it encounters the rolling friction F_r. Therefore, using the equations of motions we can find out the distance $'S'$ for an initial velocity of the ball as follows:

$$S = S1 + S2$$

$$S = \frac{\left(\frac{5}{7}v_i\right)^2 - {v_i}^2}{-2u_k g} + \frac{0 - \left(\frac{5}{7}v_i\right)^2}{-2u_r g}$$

The required ball velocity is then found in terms of 'S' as follows:

$$v_i = \sqrt{S / \left(\frac{\left(\frac{5}{7}\right)^2 - 1^2}{-2u_k g} + \frac{-\left(\frac{5}{7}\right)^2}{-2u_r g}\right)}$$

3.5.3 Virtual Mass:

The virtual mass of the end effector point 'EE' is needed to analyse the impact. To find it, the projected final configuration of the foot is taken where the impact is perceived and inverse kinematics is used to find the joint angles. Using the resulting joint angles of the kicking leg, the mass matrix of the leg and the linear Jacobian of the point 'EE' is found as explained in section 3.3.3. Then the effective mass m_{eff} is found in the direction of the vector $\hat{\boldsymbol{n}}$ as described in section 3.3.3.

3.5.4 Impact Consideration:

As the foot contour is almost spherical, the contact of the foot with the ball is almost linearly elastic and so a linear elastic collision model is used for impact

analysis. The virtual mass of the 'EE', ball mass, and the required ball velocity v_i are used with momentum and kinetic energy conservation principles to find the required end effector velocity at the final position 'FP'.

$$v_{EE}{}^{FP} = v_i * (m_{eff} + m_{ball})/(2m_{eff})$$

$$\vec{v}_{EE}{}^{FP} = v_{EE}{}^{FP}\hat{\boldsymbol{n}} = v_{EE}{}^{FP}\begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix}$$

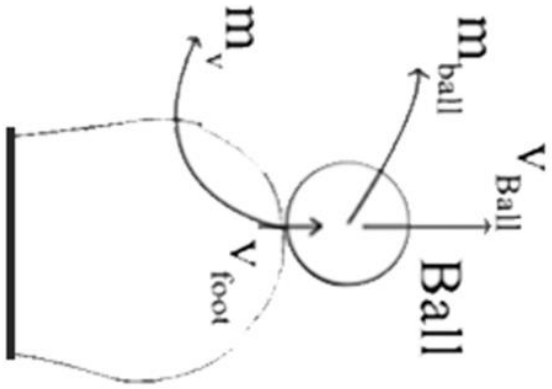


Figure 3.15 Impact Analysis

3.5.5 Trajectory planner:

The trajectory planner takes into account four via points:

1. Initial position of the end effector 'IP'.
2. The swing back point or retraction point 'RP'.
3. The ball hit point or final position 'FP'
4. The upswing point 'UP'.

3.5.5.1 Initial Position (IP)

The initial position of the end effector '*EE*' is found by the forward kinematics using the joint angle values from the position sensors.

3.5.5.2 Retraction Point (RP)

The retraction point is found by extending the direction vector $\hat{\boldsymbol{n}}$ in the opposite direction and intersecting it with the circle of radius '*r*', where r is manually set according to the reachable space and the required speed of the foot.

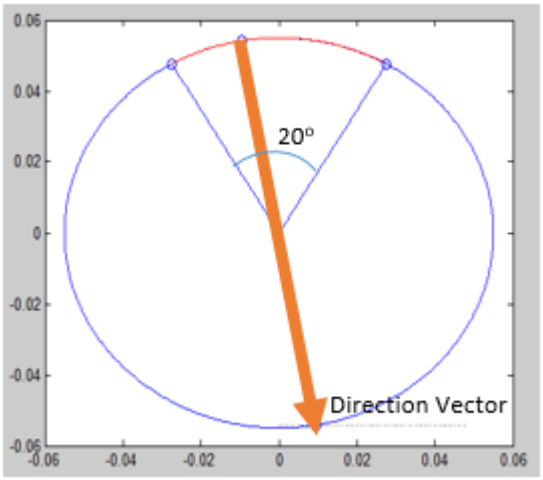


Figure 3.16 Retraction Point

The maximum angle for the position of the retraction point is set to 20°.

3.5.5.3 Final Position (FP)

The final position of the foot is found with respect to the position of the ball by:

$$\overrightarrow{FP} = \vec{\boldsymbol{P}}_{\boldsymbol{Ball}} - ballradius.\hat{\boldsymbol{n}}$$

3.5.5.4 The upswing point (UP)

The upswing point is the final point set after the ball is hit so that all of the energy from the foot is passed into the ball. The upswing point is found by the following equation:

$$\overrightarrow{UP} = \overrightarrow{FP} + Displacement.\hat{\boldsymbol{n}}$$

After finding all the via points, the inverse kinematics is to find the joint angles at each of the end effector positions.

3.5.6 Trajectory Generator:

The joint position configurations found in trajectory planning phase are now used to create the smooth trajectory splines with required velocity outputs. The Cartesian velocities for the points IP, UP, and RP are set to zero, whereas the velocity at the FP is ${\vec{v}_{EE}}^{FP}$ found from the impact analysis. The velocities are then converted from the Cartesian space into joint space velocities using the relation:

$$\dot{\theta} \;=\; J^{-1}v$$

The joint accelerations at each of the points are set to zero and the coefficients of each quintic curve are found with coinciding boundary conditions between the successive curves. The coefficients of each curve defined by the time interval $T \;=\; [T_i \; T_f]$ are given by the simultaneous equations in the form:

$$\begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = \begin{bmatrix} 1 & T_i & {T_i}^2 & {T_i}^3 & {T_i}^4 & {T_i}^5 \\ 0 & 1 & 2T_i & 3{T_i}^2 & 4{T_i}^3 & 5{T_i}^4 \\ 0 & 0 & 2 & 6T_i & 12{T_i}^2 & 20{T_i}^3 \\ 1 & T_f & {T_f}^2 & {T_f}^3 & {T_f}^4 & {T_f}^5 \\ 0 & 1 & T_f & 3{T_f}^2 & 4{T_f}^3 & 5{T_f}^4 \\ 0 & 0 & 2 & 6T_f & 12{T_f}^2 & 20{T_f}^3 \end{bmatrix} \begin{bmatrix} \theta_i \\ \dot{\theta}_i \\ \ddot{\theta}_i \\ \theta_f \\ \dot{\theta}_f \\ \ddot{\theta}_f \end{bmatrix}$$

Where the time interval for the curves IP-RP and RP-FP is 0.25 seconds and 0.1 seconds for the final curve from FP-UP. The total time for the kick trajectory is then 0.6 seconds. The individual curves are generated by the following 5-degree polynomial:

$$F(x) = ax\text{^}5 + bx\text{^}4 + cx\text{^}3 + dx\text{^}2 + ex + f$$

3.5.7 Stability Module:

The stability module of the robot first shifts all the mass of the robot on the support leg and lifts the kicking leg to an initial height. It then takes as input the trajectory of each joint of the kicking leg from the trajectory planner and using the whole body dynamics solution as described in section 3.3.4, it finds the projected movement of the zero-moment point (ZMP) in x and y directions. Using the projected movement during the kicking, it finds the necessary previews of ZMP throughout the trajectory for the preview controller. The required tracking of the center of mass (COM) trajectory is accomplished by the two support leg joints, namely, HipPitch and HipRoll. Two preview controllers are used in parallel, for x and y directions. The

HipPitch joint provides the trajectory tracking in x direction while the HipRoll joint is used for y direction.

3.5.8 Execution:

The execution phase of the algorithm, initiates the stability module in one thread and executes the kick trajectory in another. The two threads run side by side and the kick trajectory is constantly updated accordingly. The kick is executed and the robot is shifted back to its initial standing position.

Chapter 4

Code Compilation

4.1 Working with NaoQi

Aldebaran Robotics have provided a built-in software architecture “NaoQi” for NAO which runs and controls the robot. Therefore, to program the robot, we need to interface with the NaoQi framework. This framework controls the communication between different modules (audio, video, and motion), the programming and the memory of the robot. The framework is cross-platform and cross-language therefore it is possible to develop with it on Windows, Linux or Mac and it works with C++ and Python.

4.2 Programming

To develop advanced programming codes on NAO according to our requirements, C++ is preferred. For our coding purposes, we have used Linux OS. Linux is generally preferred because of its simplicity, ease of debugging the codes and advanced code handling utilities.

4.2.1 Necessary Build Packages

The code has various dependencies which are required for building, compiling and running the code on NAO:

1. build-essential

2. cmake
3. git-core, gitk
4. python2.7-dev
5. qt4-dev-tools
6. libboost-all-dev
7. qibuid

These packages can be obtained by adding the following command in the Linux terminal:

```
$> sudo apt-get install <package-name>
```

4.2.2 NaoQi SDK

The NaoQi C++ SDK is required for code compilation and building to run the codes on Linux (programs built by this won't run on actual NAO). Get the latest SDK "naoqi-sdk-[version]-tar.gz." from the Aldebaran website. Extract the SDK into some folder:

```
$> tar -xvzf  naoqi-sdk-[version]-tar.gz.
```

4.2.3 Cross Toolchain

The Cross toolchain is required to compile the code for running on real NAO. NAO has the Atom system image and therefore to run the code compiled on Linux, we need cross-compilation. This helps in running the code on both the platforms. Get the latest atom toolchain "ctc-atom-[version]-tar.gz." from the Aldebaran website. Extract the toolchain into another folder:

```
$> tar -xvzf  ctc-atom-[version]-tar.gz.
```

5.2.4 Build and Compilation

To build the code, you will need to initiate the qibuild in a directory:

```
$> qibuild init
```

Then make a toolchain (Cross toolchain or SDK toolchain) by using the following command:

```
$> qitoolchain create [TOOLCHAINNAME] /path/to/SDK/toolchain.xml
```

Or,

```
$> qitoolchain create [TOOLCHAINNAME] /path/to/ctc/toolchain.xml
```

Once the toolchain is created, you will need to add its configs to your qibuild worktree. Use the following command:

```
$> qitoolchain add-config [TOOLCHAINNAME] –t [TOOLCHAINNAME]
```

To configure and build the project:

```
$> qibuild configure [ProjectName] –c [TOOLCHAINNAME]
```

```
$> qibuild make [ProjectName] –c [TOOLCHAINNAME]
```

The project will build in toolchain-build/sdk/bin or toolchain-build/sdk/lib folder according to your cmake config. To modify your cmake config, the necessary changes can be made in cmakelists.txt file.

4.2.5 Running the Code

The code compiled by using C++ SDK toolchain will only run on virtual robots. A virtual robot can be initiated by using running naoqi-executable in /path/to/SDK/ folder. To run the program on virtual nao, go to toolchain-build/sdk/bin and run the project executable. To run the code on real nao, just connect the nao on a private network with you PC and run the same executable with ip address,

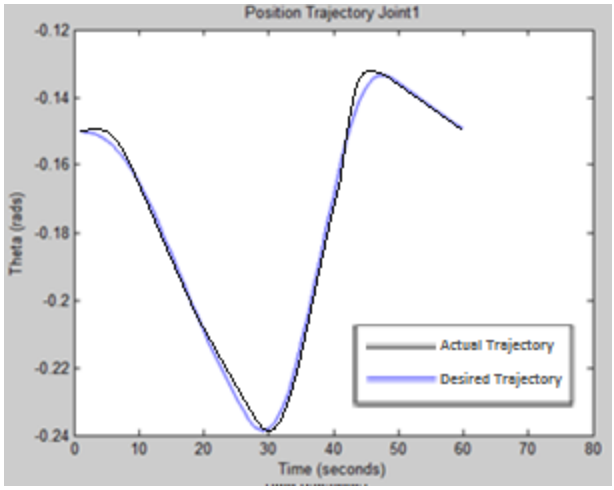
```
./ProjectName.exe –pip [robot-ip] –pport [robot-port][Default = 9559]
```

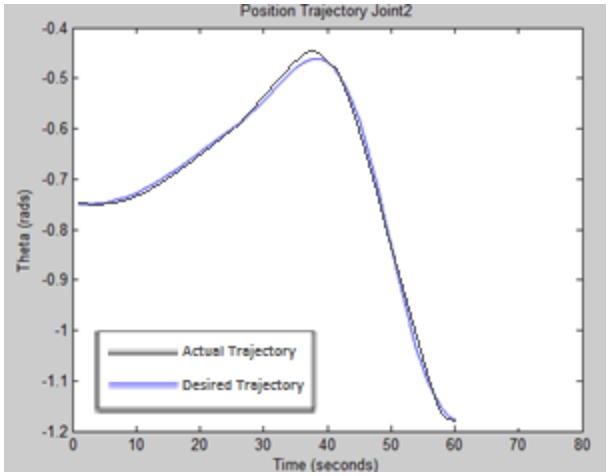
Chapter 5

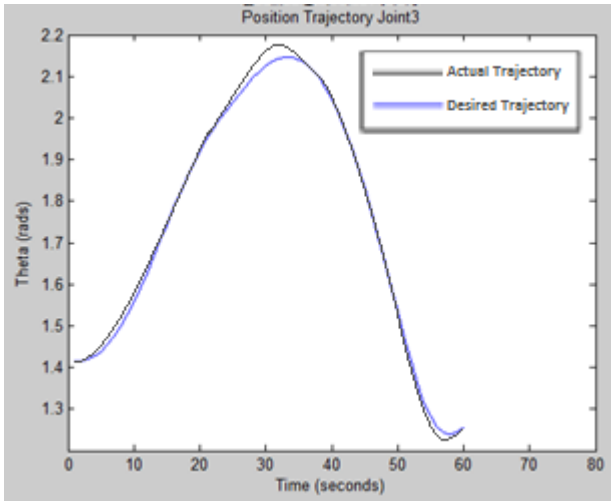
Results and Discussion

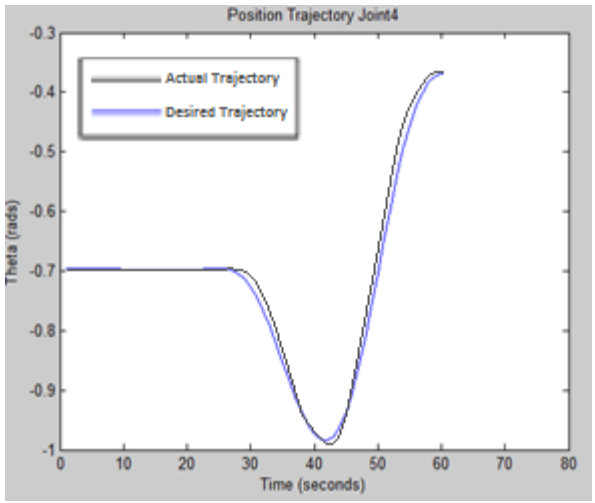
5.1 Joint Trajectory Output

The commanded trajectories that are made by the trajectory generator for each joint and the actual trajectory angles taken from the position sensors for a 6-meter straight kick are shown. The trajectory for joint 5 also distinguishes the separate regions of three curves.









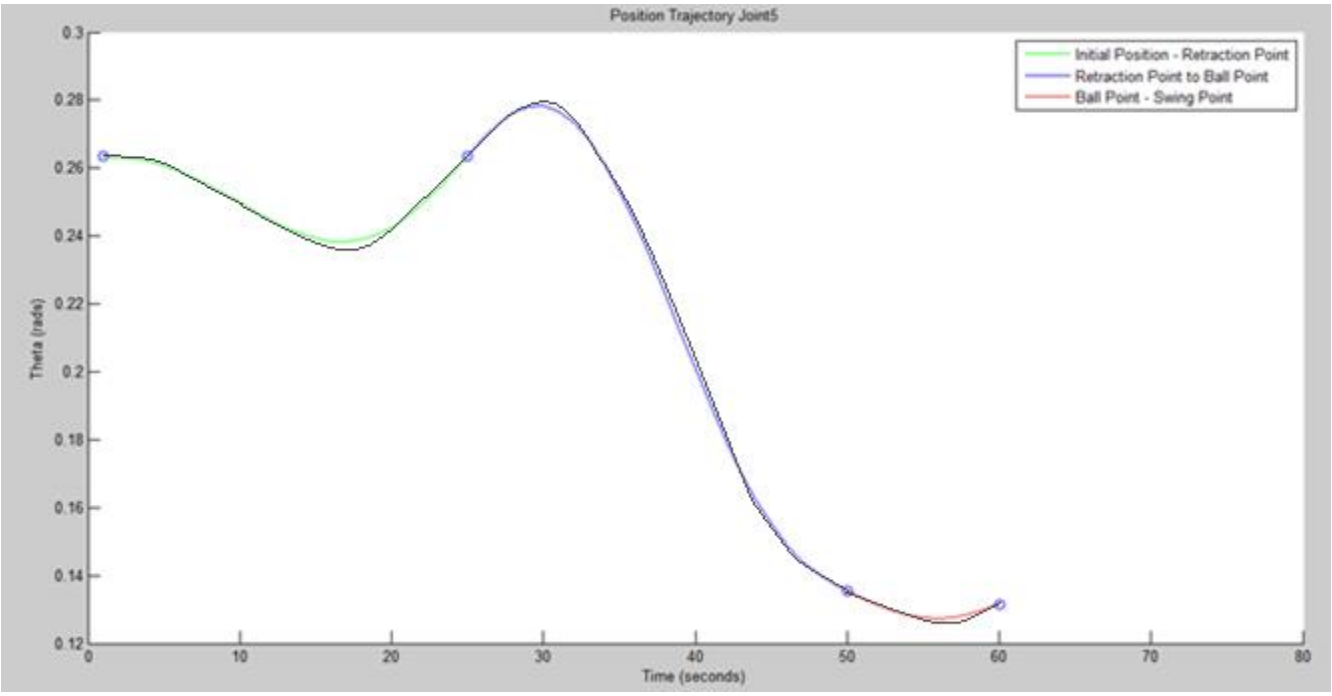


Figure 5.1 Actual and Desired Joint Trajectories

The trajectory planning algorithms that have been used here have provided us with smooth and continuous curves which is not only necessary for getting better results but also for the purpose of making our motions harmless for the motors, links and other parts of the humanoid. Sudden and jerky motions could result in damaging of motors which was one of the major problems faced in the beginning when the cubic splines interpolation was used instead of quintic splines. As our trajectories are fast and need a sudden velocity at the final point, cubic splines gave us the provision to define the velocities, but due to their high values, the resulting acceleration values seemed to be much higher than expected. High values of acceleration also needed higher torques and therefore our trajectories were far slower than expected. The problem was that to follow that trajectory, we needed much high torques, which was

not possible and the Device Communication Manager (DCM) of NAO which controls the PID control loop limited the trajectory at the max power and torque available. To solve this problem, we moved on to the use of quintic splines, and defined zero accelerations at each end point of the curves, which resulted in splines that were continuous and differentiable up to second order (smooth acceleration curves). Moreover, we can see that the trajectories are also being tracked quite precisely by the DCM.

5.2 Impact Results

To verify our system model, we performed an experiment to find out the actual ball velocity after the impact. In this experiment, we fixed a FireFly MV 60 fps camera over the robot and recorded the video of its kicks (Figure 5.2). After getting the data for all the kicks, we ran the videos through matlab image processing tools. We checked for the movement of the ball between the two frames just before and after the impact (Figure 5.3). Let the movement of the ball and diameter in pixels in between the frames be s_p and d_p then by assuming constant height for two frames, the displacement of the ball in meters can be found from the equation,

$$s_m = s_p * d_m/d_p$$

Where d_m= 0.065m. Then the velocity can be found as

$$v = s_m \times 60$$

Figure 5.2 Experimental Setup for Impact Verification

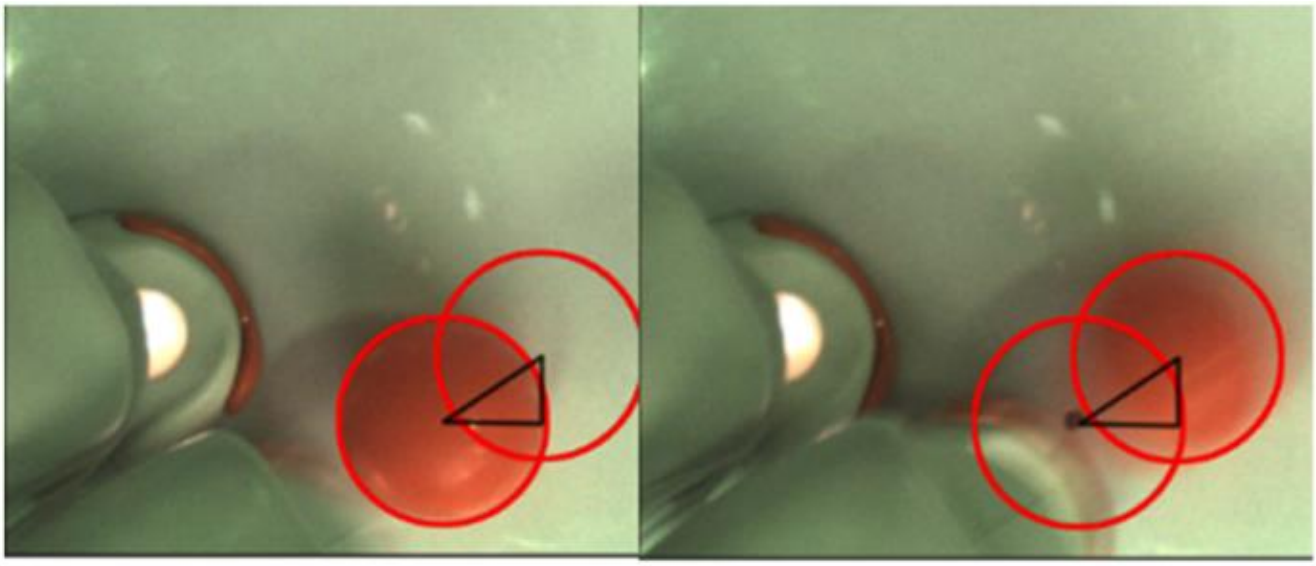


Figure 5.3 6-meter kick 60 fps frames just before and after the kick

The results of the experimental setup for the verifications of impact velocities is shown.

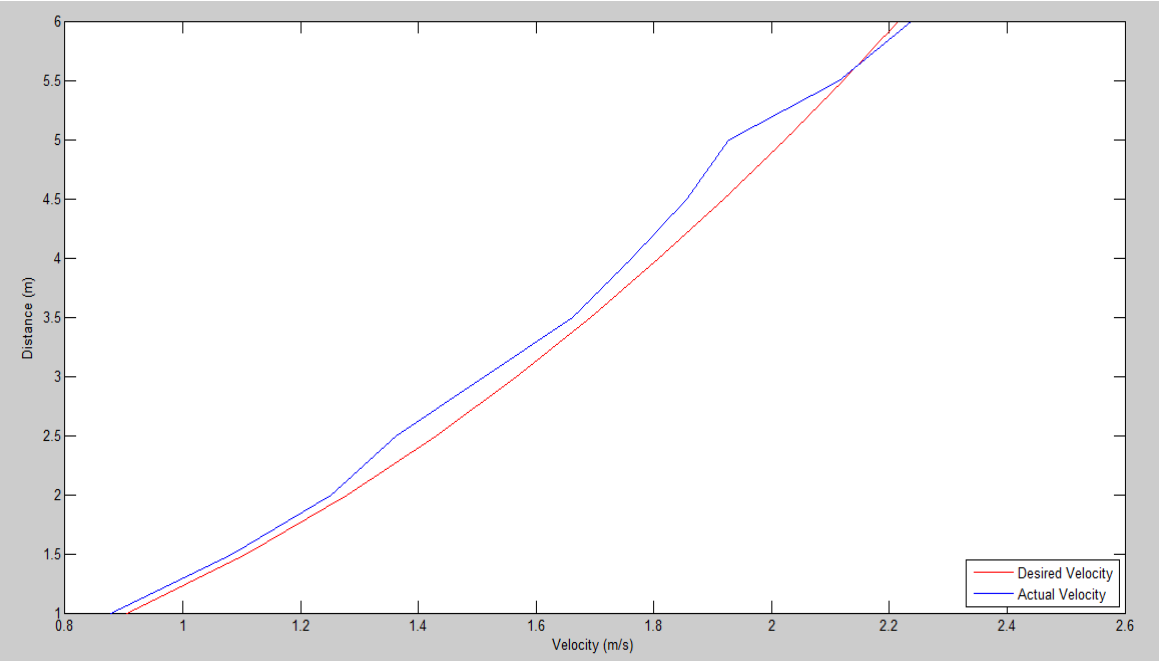


Figure 5.4 Actual and Desired Ball Velocity

The impact results experiment was necessary for the purpose of understanding the outcomes of our research. It was also very crucial in finding out the problems that were occurring related to our trajectory planning, because in our first batch of experiments, the speed of the ball was far lower than expected. This initially gave us the idea that maybe our virtual mass calculations and dynamics of the body have not been properly solved, but when we looked at the movement of the foot, we found out that the foot is not reaching the desired velocities and thus the problem of trajectory planning was identified. Once that problem was solved, we ran the second batch of experiments which resulted in very precise results as shown in Figure 5.2.

5.3 Kick Results

The results of the experiment for the verifications of friction model and ball dynamics is shown. The graph shows the results of different straight kicks with desired distances (1m, 2m, 3m, 4m, 5m, 6m).

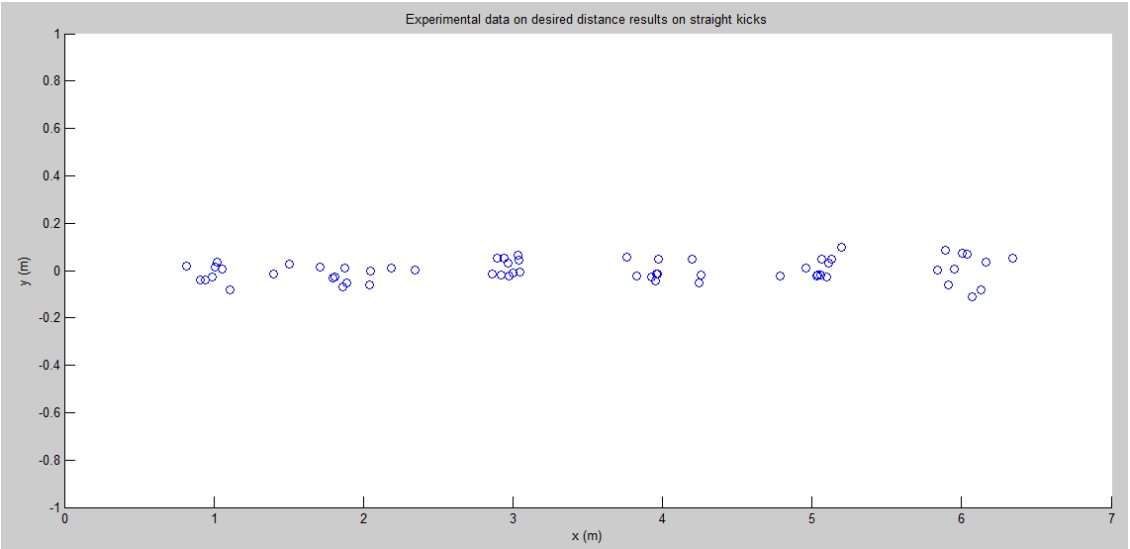


Figure 5.5 Measured Distances of the ball

Distance (m)	Standard error mean
1	6.8%
2	6.1 %
3	2.01 %
4	5.39 %
5	3.57 %
6	4.73 %

Table 5.1 SEM between measured and desired ball distances

As we can see from the Table 5.1, the results of our desired distances are also quite accurate having the error range within ~7%, but even though these results were accurate and commendable, we could not achieve the same in directional kicks. The main problem in directional kicks was that the foot contour of the robot is not as spherical from all sides, and only forms a conservative collision model up to the 10 degrees on both side from the middle. From that point onwards, its shape has sudden corners because of which the collision model does not apply as expected. Another problem was that in our iterative inverse kinematics solution the threshold is set to +-5 mm otherwise the solution does not converge. Due to this, our foot positioning was not as accurate as desired.

5.4 Stability Results

The movement of the ZMP in x and y directions during the kick trajectory is shown.

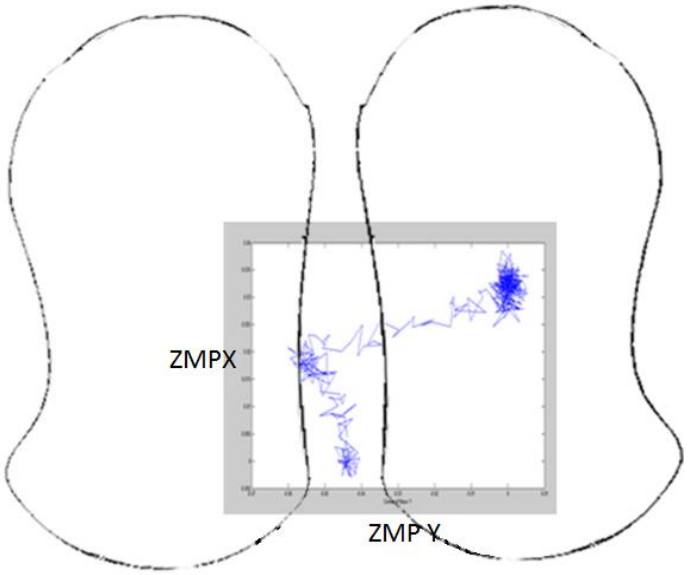


Figure 5.6 ZMP Movement Throughout the 6-meter kick

As we can see from the figure (5.4), the zero-moment point of the robot stays well within the support polygon and therefore, our robot is highly stable during the kick trajectory generation.

5.5 Issues

A few other problems that we came through were related to the integration of our program with NaoQi software. One of the problems was that NaoQi does not allow us to make changes to the PID control loop because of which the trajectories that were not being followed with desired ones as accurately as expected. Another problem was the interruption caused by the autonomous life module of the robot. The autonomous life robot has its own modules running in parallel and increases processing time for kick module which needs to be completely removed.

Chapter 6

Conclusion

As have been discussed and presented in the previous sections, despite the issues, the results of our research have been very precise and accurate for straight kicks within the angle range of 0~100 on both sides. To further increase the capability of our kick module, following work is suggested:

1. The trajectory planner of our kick module is constrained within the position and velocity ranges but it is not yet optimized with respect to any parameter such as distance, speed and if the optimization of the current planner can be performed for maximum effective mass and velocity, we can achieve the highest possible distances with increased accuracy.
2. Another very useful research that can be done is to include the coefficient of restitution in the impact model to get better accuracy for both multidirectional kicks.
3. As the kick module has infinite outcomes and the results are very precise, it is recommended that machine learning algorithms be applied to train the robot according to different environments.
4. The iterative inverse kinematics algorithms are time consuming and therefore it is recommended that they be replaced by an analytical solution.

References

[1] Teppei Tsujita, Atsushi Konno, Shunsuke Komuzunai, Yuki Nomura, Tomoya Myojin, Yasar Ayaz and Masaru Uchiyama, “Humanoid Robot Motion Generation Scheme for Tasks Utilizing Impulsive Force,” International Journal of Humanoid Robotics (IJHR), World Scientific Publishing Company, Vol. 9, No. 2, pp. 1250008-1 to 1250008-23, 2012.

[2] Teppei Tsujita, Atsushi Konno, Shunsuke Komizunai, Yuki Nomura, Takuya Owa, Tomoya Myojin, Yasar Ayaz and Masaru Uchiyama, "Analysis of Nailing Task Motion for a Humanoid Robot," Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1570-1575, France, September 2008.

[3] Teppei Tsujita, Atsushi Konno, Shunsuke Komizunai, Yuki Nomura, Takuya Owa, Tomoya Myojin, Yasar Ayaz and Masaru Uchiyama, "Humanoid Robot Motion Generation for Nailing Task," Proceedings of IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), pp. 1024-1029, China, July 2008.

[4] Felix Wenk and R¨ofer, T.: Online Generated Kick Motions for the NAO Balanced Using Inverse Dynamics: In RoboCup 2013: Robot World Cup XVII. Volume 8371 of the series Lecture Notes in Computer Science pp 25-36

[5] Inge Becht, Maarten de Jonge, and Richard Pronk. A Dynamic Kick for the Nao Robot. Project report (Universiteit van Amsterdam, 26 July 2012).

[6]Raul rojas, Mark simon: Like a rolling ball

[7] Shuuji KAJITA, Fumio KANEHIRO, Kenji KANEKO, Kiyoshi FUJIWARA, Kensuke HARADA, Kazuhito YOKOI and Hirohisa HIRUKAWA, Biped Walking Pattern Generation by using Preview Control of Zero-Moment Point

[8] Nikolaos Kofinas, Forward and Inverse Kinematics for the NAO Humanoid Robot

[9] Nima Shafii , Abbas Abdolmaleki , Rui Ferreira, Nuno Lau, Luis Paulo Reis, Omnidirectional Walking and Active Balance for Soccer Humanoid Robot

[10] MIOMIR VUKOBRATOVIC, ZERO-MOMENT POINT — THIRTY FIVE YEARS OF ITS LIFE

[11] Oussama Khatib, Inertial Properties in Robotic Manipulation: An Object-Level Framework

[12] M¨uller, J., Laue, T., R¨ofer, T.: Kicking a Ball – Modeling Complex Dynamic Motions for Humanoid Robots. In: del Solar, J.R., Chown, E., Ploeger, P.G. (eds.) RoboCup 2010: Robot Soccer World Cup XIV. Lecture Notes in Artificial Intelligence, vol. 6556, pp. 109–120. Springer (2011)