Kinematic Analysis, Motion Planning and Control of a Humanoid Robot

Project Report

Submitted in partial fulfillment of
the requirements for the award of
Bachelor of Technology Degree in
Electrical and Electronics Engineering
of the APJ Abdul Kalam Technological University

Submitted by

Aaron Xavier (TVE15EE002)
Adithya B Uday (TVE15EE007)
Amritha Rajeev (TVE15EE025)
Vaishnav J (TVE15EE112)

Guided By
Dr. Jisha V R

Department Of Electrical Engineering

College Of Engineering Trivandrum

Thiruvananthapuram-16

2019

DECLARATION

I undersigned hereby declare that the project report **KINEMATIC ANALYSIS, MOTION PLANNING AND CONTROL OF A HUMANOID ROBOT**, submitted for partial fulfillment of the requirements for the award of degree of Bachelor of Technology of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by me under the supervision of Dr. Jisha V R. This submission represents my ideas in my own words and where ideas or words of others have been included, I have adequately and accurately cited and referenced the original sources. I also declare that I have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University. .

Place: AARON XAVIER

Date: ADITHYA B UDAY

AMRITHA RAJEEV

VAISHNAV J

DEPARTMENT OF ELECTRICAL ENGINEERING

COLLEGE OF ENGINEERING TRIVANDRUM

2019

https://v2.overleaf.com/2243265127zxgxydsgshcg

CERTIFICATE

This is to certify that this project report entitled ***"Kinematic Analysis, Motion Planning and Control of a Humanoid Robot "*** is a bonafide record of the project work done by **Aaron Xavier, Adithya B Uday, Amritha Rajeev and Vaishnav J** of 8th semster B.Tech under our guidance towards the partial fulfillment of the requirements for the award of **B.Tech Degree in Electrical and Electronics Engineering** of the APJ Abdul Kalam Technological University during the year 2019.

Dr. Jisha V R
Associate Professor
Department of Electrical Engineering
College of Engineering Trivandrum

Prof. R. Harikumar
UG Co-ordinator
Department of Electrical Engineering
College of Engineering Trivandrum

Dr. P. Sreejaya
Head of the Department
Department of Electrical Engineering
College of Engineering Trivandrum

ACKNOWLEDGEMENT

First of all, we would like to express our deepest appreciation and gratitude to our guide **Dr. Jisha V R**, Associate Professor, Department of Electrical Engineering, College of Engineering Trivandrum, for all her guidance and support. We could not have completed this project if it wasn't for her constant guidance and advices.

We gratefully acknowledge **Dr. P. Sreejaya** , Head of the Department, Department of Electrical Engineering, for her whole hearted cooperation and encouragement.

We express our gratitude to **Prof. R. Harikumar**, UG coordinator, Department of Electrical Engineering, College of Engineering, Trivandrum, for all necessary help extended to us during the evaluation process in doing the project work.

We also acknowledge our gratitude to all other members of faculty in the Department of Electrical Engineering, our family and friends for their wholehearted cooperation and encouragement. Above all, we thank GOD Almighty, without whose help, we wouldn't have reached this far and completed the work.

ABSTRACT

Humans have always been extremely adroit and are adept at intricate motion which is nearly impossible for any other animal on the planet. The main objective of the project is to design and fabricate a humanoid robot with natural walking capabilities using foot-placement control and plan an obstacle-free path for the same in an indoor environment. The walking pattern is generated using the 3D Linear Inverted Pendulum Model(LIPM) based approach, by employing the ZMP Criterion. Inverse Kinematic Analysis is performed at the joint level and the resulting joint angles are used for generating a lookup table. This technique is tested through both simulation and hardware on a 12 DOF bipedal model and the results obtained are satisfactory. Mapping of an indoor environment is done using a laser rangefinder and the shortest path, avoiding obstacles, is obtained from the Probabilistic Roadmap Algorithm.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

A humanoid robot is a robot that has human-like shape. Humanoid robots are one of the captivating autonomous systems in human society as they can work well in indoor environment designed for humans. The environment of the modern society is designed for humans, the width of corridor, the height of stair and the position of a handrail are determined to fit the size and motions of humans. Thus a more economical solution is to develop humanoid robots than to modify the whole environment. Most of the tools for humans are designed to be used by humans.Thus anthropomorphism helps for proper interaction within human environment.

Bipedal robots are able to move in areas that are normally inaccessible to wheeled robots and areas filled with obstacles that make wheeled locomotion impossible. An uneven floor has to be made flat, a narrow passage should be removed and a lift must be available for a robot on wheels. Also it may be easier for people to interact with walking robots with human morphology rather than robots with non-human shape. Robots should be able to do tasks alongside humans being our partners enjoying communications. These days humanoids mainly finds scope in amusement and entertainment. Recently, many studies have focused on the development of humanoid biped robots. Among the robots HRP, WABIAN and ASIMO are the most famous ones.

1.2 Problem Statement

The design and development of human-like robots has been one of the main topics in advanced robotics research during the last years. The aim of the project is to

design and control a Humanoid with natural walking gait and human interaction capabilities. In order to mimic the human like walking, a dynamically stable model has to be realized. Dynamic equations and dynamic system modeling is used to predict the position, velocities and acceleration to achieve the same. Thereby obtaining the overall stability of the robot can be achieved.

1.3 Motivation

Humanoid robots are envisioned as the ideal but probably most complex service robot. Human-like structure allow them to perform well in human environments, they would be able to navigate anywhere inside a house, to go up and down by stairs, to use human tools and machines , or assist people in their daily life. An anthropomorphic design with human social characteristics is believed to increase the acceptance of robots for people. In some aspects humanoid robots can overperform human capabilities since they do not feel fatigue, sleepiness, boredom. This can act as an advantage in safety as disastrous scenarios need robots with capability of acting in dangerous places without risking human life.

1.4 Objectives

- To develop a humanoid robot with bipedal walking capabilities.
- Designing and implementing a dynamically stable humanoid robot
- Design of structure limbs which mimics human motions
- Realisation of kinematic model for a robot with natural walking gait
- Finding the optimum range of torques, speeds and angles for the joints, so that the humanoid is stable for a given range of positions and orientations.
- Indoor navigation using efficient path planning algorithms, avoiding obstacles.

1.5 Outline of the Project

Chapter 1 gives a brief introduction into our project. In chapter 2 a literature survey is conducted. Chapter 3 gives an idea about the proposed system used. Chapter 4 deals with the methodology of the system. The hardware design is given in chapter 5. The software design is discussed in chapter 6. Results and discussions are conducted in chapter 7. Finally, the report is concluded with the conclusions and future scope in chapter 8.

Chapter 2

Literature Review

The advances in robotics and the expanding significance of humanoid robots led to a growing business interest in this field. There are different control algorithms proposed to different type of robots. Each control algorithm has its own pros and cons. The selection of the control algorithm depends on the type of robot to be developed and the number of degrees of freedom(DOF). The three dimensional passive dynamic walking robots[4] utilises the natural dynamics of the body to walk passively down a slope using only the gravitational potential. But they were hard to control. Some of the control algorithms widely used are zero moment point method, minimal actuation method[6], foot capture point method[7] and hybrid state driven autonomous control(HYDAC). The control techniques were identified and a comparison study was conducted on their performance and feasibility.

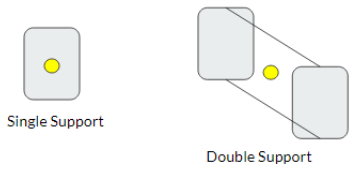


Figure 2.1: Zero Moment Point stability criteria

Zero moment point(ZMP) specifies the point with respect to which dynamic reaction force, at the contact of the foot with the ground, does not produce any moment. At ZMP, the total inertia force equals zero. It is an indicator of the stability of the robot. If ZMP point is in the foot shadow then the biped is stable and else unstable.So the key principle is that, in order to attain stability, the stationary leg should have no net torque acting on it.In static walking, the biped moves very slowly so that the dynamics can be ignored. Then biped projected centre of gravity (PCOG) must be within the supporting area.In dynamic walking the motion is fast

and hence dynamics cannot be negligible.Thus we should look at ZMP rather than PCOG.

Minimally actuated serial robot[6] is a serial rigid structure consisting of movable actuators and multiple links connected by passive joints. These actuators are freely moved through the links to desired positions. So, more the number of links, more is the area that can be covered by the robot. Minimally actuated serial robots(MASR) are practically limited to hyper-redundant robots like snake robots.They can navigate around obstacles in highly confined spaces.

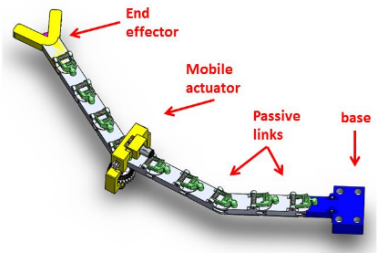


Figure 2.2: Minimal actuated robot

The foot capture method[7] was introduced by splitting the Center of Mass(CoM) dynamics into stable and unstable components. The capture point specifies when and where a humanoid must step to in order to maintain balance. This method helps in the push recovery of humanoid robots.Boston dynamics is applying this principle in the control of its robot named Atlas.This algorithm says when and where the foot should be placed when an external force is applied on the stable biped. When the Capture Region intersects the Base of Support, a humanoid can modulate its Center of Pressure to balance and does not need to take a step. When the Capture Region and Base of Support are disjoint, the humanoid must take a step to come to a stop. If the Capture Region is outside the kinematic workspace of the swing foot, the humanoid cannot stop in one step.

The Hybrid-state driven autonomous control(HYDAC) is an analytical approach in bipedal locomotion control which is superior in accuracy and efficiency compared to heuristic control methods like ZMP. Here, periodic stability of the walking gait is analysed on the basis of two level hierarchical control. They are task level control and supervisory level control. But HYDAC studies are limited to sagittal plane only and thus a practical application of this control strategy is not possible. The heuristically motivated 2-level hierarchical control framework provides a transparent and adaptable control architecture to suit for different locomotion patterns as well as for different terrain profiles.

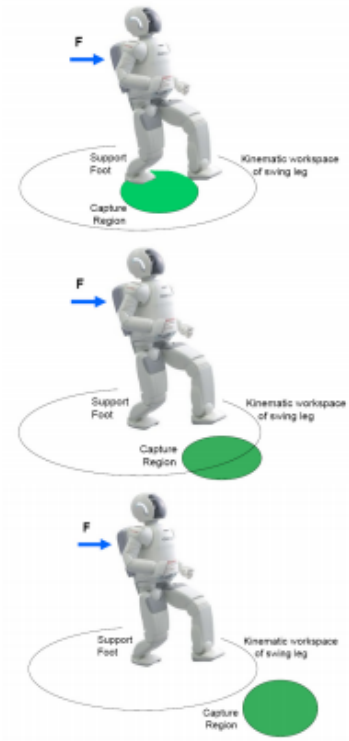


Figure 2.3: Foot Capture Method

The kinematic data required can be obtained using a computer program from relative motions as proposed by Zarrugh et al. Various photographic techniques were adopted for the same. Opto-electronic methods, interrupted light methods and cinematographic methods are widely adopted. A person is made to walk through a treadmill in two different ways. In the first method called the free walking, the self selected step rate varies around 85-130 steps/min and in second method called forced walking, step rate is forced to be constant at around 90-140 steps/min. Goniometric or pin study methods are used to measure the body rotations during walking. In goniometric methods, the angular rotation between two body segments are measured using an instrument called goniometer. Pin study methods provide more accurate readings, where pins are inserted directly into the bone, linked mechanically to a measuring device.

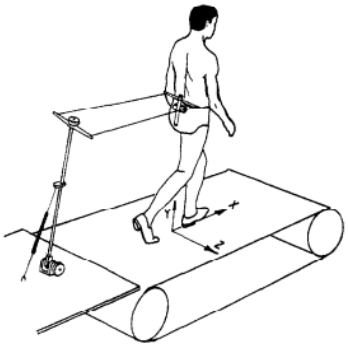


Figure 2.4: Generation of human gait kinematics

The Zero Moment Point(ZMP) concept is followed by most of the humanoid robots including Honda humanoid robot Asimo[3], HUBO[5] etc. Honda studied extensively about how humans walk. The Robot achieves its stability by integrating controls like ground reaction force, zero moment point control and foot landing position control. It is extremely stable and can perform various tasks like running, climbing steps and walking up a slope. It does not move quite like people do. Its knee is always bent and is energetically inefficient. The HUBO robots, compared to Asimo robots, have high rigidity of joints/links and proper selection of joint actuators.The increased stiffness improves the stability of the robot by minimizing the uncertainty of the joint positions and the link vibration control.

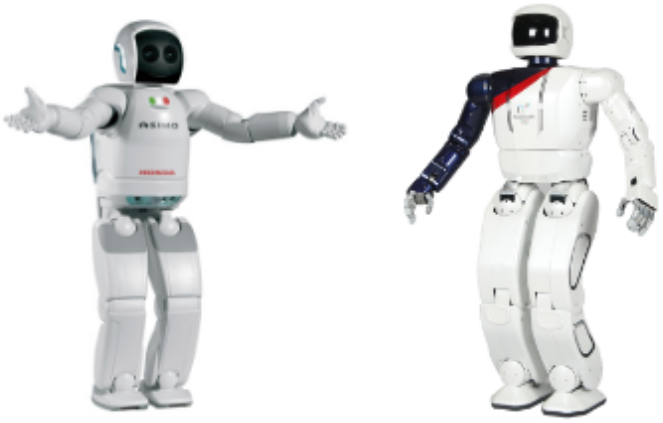


Figure 2.5: Honda Humanoid Robot Asimo and HUBO

Chapter 3

Proposed System Model

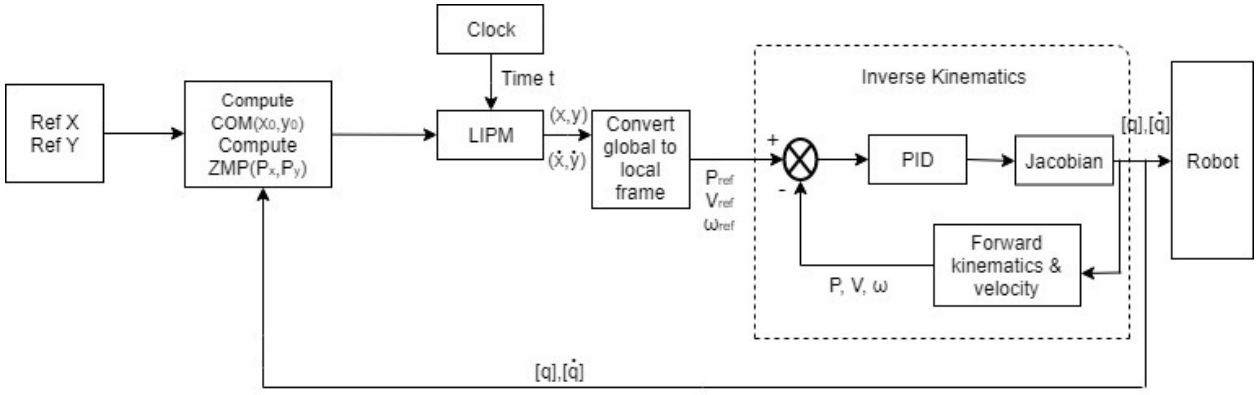


Figure 3.1: Block Schematic of Walking Pattern Generation

Figure 3.1 shows the different units of the proposed control system and the processes taking place. The input given to the system is a reference trajectory, say a line or a circle, as X and Y coordinates. Using this information, the Center of Mass and ZMP values are calculated at each time frame. The Linear Inverted Pendulum Model imbibes these inputs along with the instantaneous time t and outputs position and velocity of the body's center of mass, thereby tracking the reference trajectory. These values are then metamorphosed into the joint level local frame which yields the reference position, velocity and angular velocity for each servo actuator. The Inverse Kinematics block has an internal PID controller that minimises the error by reckoning the present state values, using Forward Kinematics and returns the joint angle and joint velocity which is given to the robot. The process is continued and the next set of values are calculated until the end location of the trajectory is

reached.

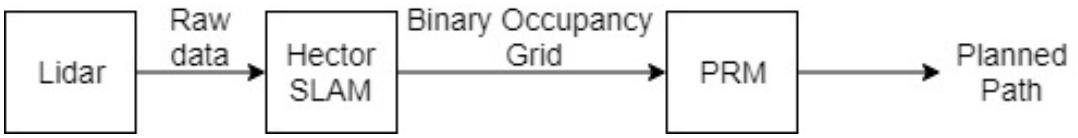


Figure 3.2: Block Schematic of Motion Planning

Figure 3.2 shows the block schematic of the motion planning unit which can be divided into two phases. The first phase is procuring the 2D map of an indoor environment by the SLAM approach, i.e Simultaneous Localisation and Mapping. For this, a Hokuyo URG-04 LX-UG01 Scanning Laser Rangefinder is used along with Hector SLAM package in ROS. A binary 2D occupancy grid map obtained from the first phase is then used in the path planning phase. Here Probabilistic Roadmap algorithm is used to obtain the most coherent and shortest path, avoiding obstacles, between the start and end locations that can be furnished as an input.

Chapter 4

Methodology

This chapter encompasses the theoretical analysis of the methods and concepts adopted in the project. Section 4.1 overviews the kinematics of humanoid robots. The concept of ZMP (Zero Moment Point) is described in section 4.2. Section 4.3 explains Linear Inverted Pendulum Method (LIPM) and the concepts of path planning is presented in section 4.4.

4.1 Kinematics

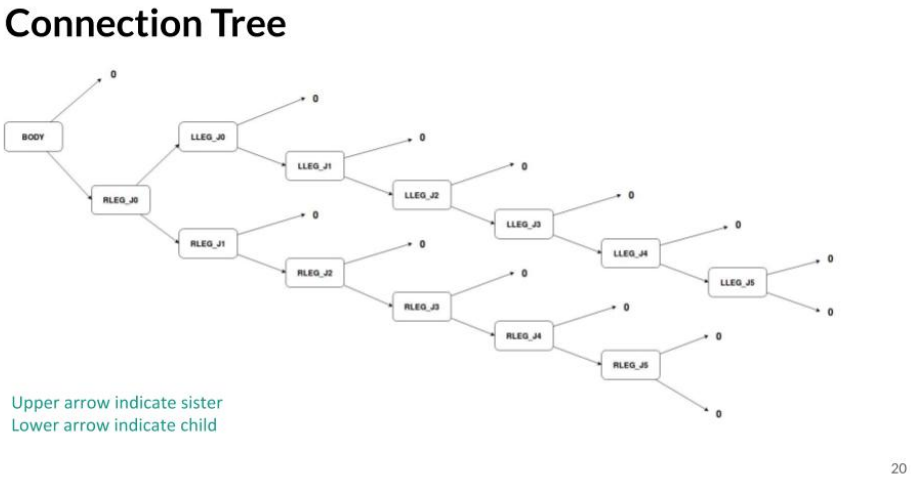


Figure 4.1: Humanoid Link Connection

In order to connect each link to get a whole humanoid robot, we use a connection rule as shown in Figure 4.1. The connection rule used here is similar to a family tree. Left arrow of each link indicates its child link and right arrow indicates the sister link. This family structure can make use of recursive programming to access all links.

4.1.1 Forward Kinematics and Forward Velocity

Forward kinematics is calculated using the chain rule of homogeneous transformation, the first step of which will be calculation of homogeneous transformation of a single link. The homogeneous transform of a link relative to the parent link is given by:

$$T_j^i = \begin{bmatrix} e^{\widehat{a}_j q_j} & & & b_j \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

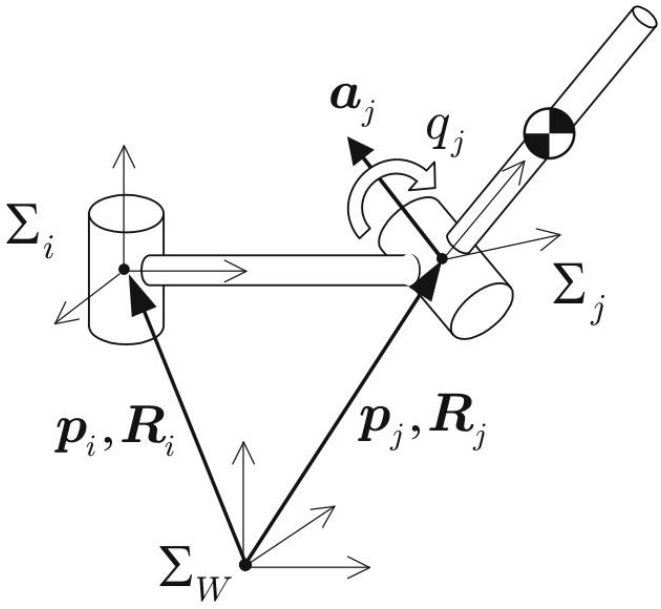


Figure 4.2: Relative position of 2 links

where a_j is the joint axis vector seen from the parent coordinates and b_j is the origin of Σ_j. q_j is the joint angle. Now considering two links as shown in Figure 4.2, the homogeneous transform of Σ_i becomes

$$T_i = \begin{bmatrix} R_i & & & p_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

where (p_i) and (R_i) are the position and attitude of the i^{th} link. From the chain rule of homogeneous transform Σ_j is

$$T_j = T_i T_j^i \tag{4.3}$$

The absolute position (p_j) and attitude (R_j) of Σ_j can be calculated using:

$$p_j = p_i + R_i b_j \tag{4.4}$$

$$R_j = R_i e^{\widehat{a}_j q_j} \tag{4.5}$$

Similar to Forward Kinematics, we can also compute the velocity of a link which is connected to its parent link. Given the joint speed $\dot{q}_j$, its linear velocity $(v_j)$and angular velocity (ω_j) can be calculated by:

$$v_j = v_i + \omega_i \times R_i b_j \tag{4.6}$$

$$\omega_j = \omega_i + R_i a_j \dot{q}_j \tag{4.7}$$

4.1.2 Inverse Kinematics

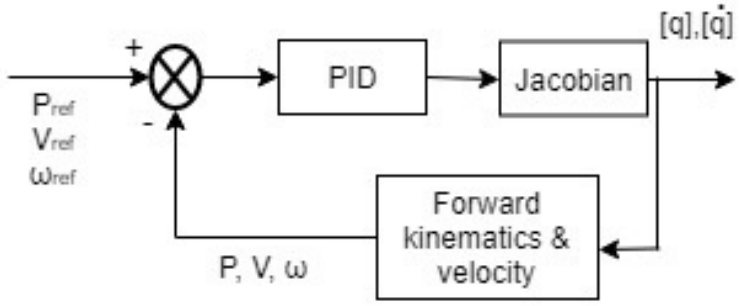


Figure 4.3: Control Schematic of Inverse Kinematics

Since analytical solution to Inverse Kinematics is too complex, a numerical solution strategy is being adopted. We use the Newton-Raphson method by considering what happens to the position and attitude (δp, $\delta\theta$) when there is a momentary change in joint angle δq, which can be mathematically expressed as:

$$\delta p = X_p(q, \delta q) \tag{4.8}$$

$$\delta\theta = X_\theta(q, \delta q) \tag{4.9}$$

Figure 5.3 shows the basic concept behind numerical approach to inverse kinematics. The desired configuration of the link P_{ref}, v_{ref} and ω_{ref} is given as a reference. Forward Kinematics and Forward velocity equations are used to obtain the current state of the link which can be expressed in terms of joint angle δq and a Jacobian matrix J as:

$$\begin{bmatrix} \delta p \\ \delta \theta \end{bmatrix} = J \delta q \tag{4.10}$$

The error which is the difference between the reference and current configurations is minimised using a PID controller and after a set of computations the reference configuration is reached. The required adjustment in joint angle to reach the target configuration can be calculated by taking inverse of the matrix as:

$$\delta q = \lambda J^{-1} \begin{bmatrix} \delta p \\ \delta \theta \end{bmatrix} \tag{4.11}$$

This adjustment is added to the old joint angle configuration which is then passed to the robot.

$$q_{new} = q_{old} + \delta q \tag{4.12}$$

4.2 Zero Moment Point

Fig. 5.4 shows an example of force distribution across the foot. The load has the same sign all over the surface, ergo it can be reduced to a resultant force R, the point of attack of which will be in the boundaries of the foot. The point on the surface of the foot, where the resultant R passed is defined as the zero-moment point, or ZMP in short.

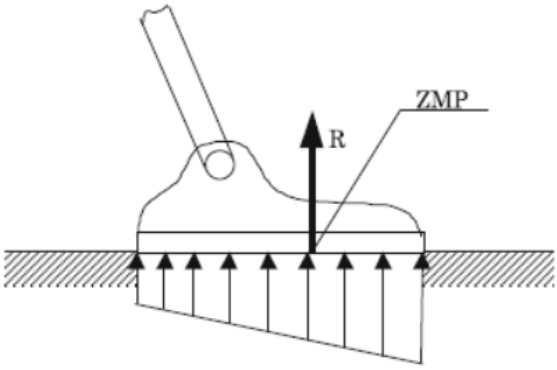


Figure 4.4: Understanding ZMP

4.2.1 Calculation of Center of Mass

If the center of mass of the j^{th} link in local frame is $\bar{c_j}$, then the center of mass of the j^{th} link in world frame is given by:

$$c_j = p_j + R_j \bar{c_j} \tag{4.13}$$

where (p_j,R_j) denotes the position and orientation of the link. We can obtain the body's center of mass by dividing the sum of moments by the total mass M as shown:

$$c = \sum_{j=1}^{N} m_j c_j / M \tag{4.14}$$

4.2.2 Calculation of Linear and Angular Momentum

The linear momentum of a robot having N links is given by:

$$P = \sum_{j=1}^{N} m_j \dot{c_j} \tag{4.15}$$

where velocity of the j^{th} link $\dot{c_j}$ is calculated as:

$$\dot{c_j} = v_j + \omega_j \times R_j \bar{c_j} \tag{4.16}$$

The angular momentum of a robot having N links is given by:

$$L = \sum_{j=1}^{N} L_j \tag{4.17}$$

where angular momentum of the j^{th} link L_j is calculated as:

$$L_j = c_j \times P_j + R_j \bar{I}_j {R_j}^T \omega_j \tag{4.18}$$

4.2.3 Calculation of ZMP

The ground reaction force can be expressed by using the ZMP (p), the force (f) and the moment (τ_p) about the vertical line including the ZMP as:

$$\tau = p \times f + \tau_p \tag{4.19}$$

Substituting the equations for linear and angular momentum, we get:

$$\tau = \dot{L} - c \times Mg + (\dot{P} - Mg) \times p \tag{4.20}$$

Solving for p_x and p_y, we get:

$$p_x = \frac{Mgx + p_z \dot{P}_x - \dot{L}_y}{Mg + \dot{P}_z} \tag{4.21}$$

$$p_y = \frac{Mgy + p_z \dot{P}_y - \dot{L}_x}{Mg + \dot{P}_z} \tag{4.22}$$

4.3 Linear Inverted Pendulum Model

A humanoid robot in 3D space can be approximated as an imaginary inverted pendulum[8] consisting of the COM of the robot and a massless leg connecting the COM and the supporting point. The pendulum is free to rotate about the supporting point and the leg can alter its length by using a kick force f which can be decomposed into 3 components(as shown in Figure 4.5):

$$f_x = \frac{x}{r} f \tag{4.23}$$

$$f_y = \frac{y}{r} f \tag{4.24}$$

$$f_z = \frac{z}{r} f \tag{4.25}$$

where r is the distance between the supporting point and the CoM.

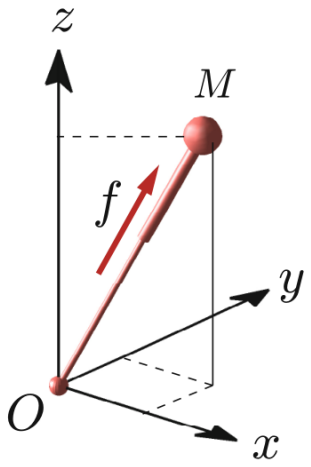


Figure 4.5: Resolving kick force vector

The motion equation of the CoM is given by:

$$M\ddot{x} = \frac{x}{r} f \tag{4.26}$$

$$M\ddot{y} = \frac{y}{r} f \tag{4.27}$$

$$M\ddot{z} = \frac{z}{r} f - Mg \tag{4.28}$$

We can obtain a solution for f in terms of a constraint plane z_c (as shown in Figure 4.6) whose normal vector is orthogonal to the acceleration vector of the COM as:

$$f = \frac{Mgr}{z_c} \tag{4.29}$$

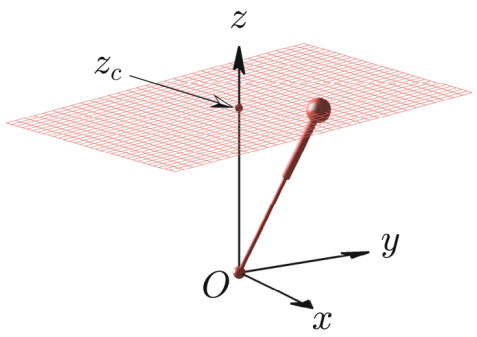


Figure 4.6: 3D Linear Inverted Pendulum

The horizontal dynamics of the CoM can be obtained by substituting equation (4.29) in equations (4.26) and (4.27), where g is acceleration due to gravity.

$$\ddot{x} = \frac{g}{z_c}x \tag{4.30}$$

$$\ddot{y} = \frac{g}{z_c}y \tag{4.31}$$

4.3.1 Modification of Foot Placements

Figure 4.7 shows the foot placement approach for walking primitives generation where p_x^* is the modified foot placement. The dynamics equation is:

$$\ddot{x} = \frac{g}{z_c}(x - p_x^*) \tag{4.32}$$

This equation can be solved analytically to obtain the instantaneous position and velocity of the center of mass as:

$$x(t) = (x_i^{(n)} - p_x^*)cosh(t/T_c) + T_c\dot{x_i}^{(n)}sinh(t/T_c) + p_x^* \tag{4.33}$$

$$\dot{x}(t) = \frac{(x_i^{(n)} - p_x^*)}{T_c}sinh(t/T_c) + \dot{x_i}^{(n)}cosh(t/T_c) \tag{4.34}$$

where $T_c = \sqrt{\frac{z_c}{g}}$

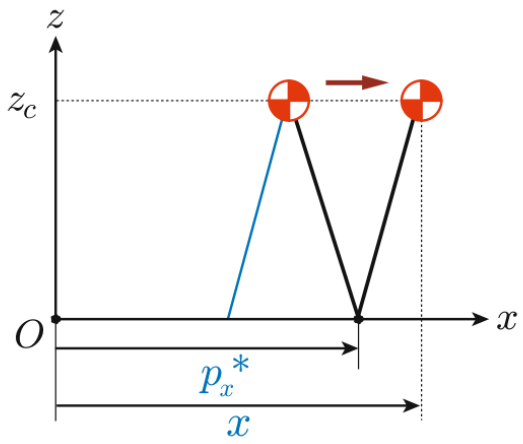


Figure 4.7: Foot Placement

4.4 Motion Planning

One of the indispensable tasks of an autonomous system is to procure knowledge about its environment. The objective of motion planning is to produce a continuous motion that connects a start configuration and a goal configuration, avoiding obstacle collision. From the sensor data(LiDAR scans) over discrete time steps, an estimate of the robot's location and a map of the surrounding environment is developed. The next step is to find an optimum, collision free path for the robot which is attained by the implementation of optimization methods like PRM algorithm or RRT algorithm on the obtained Binary Occupancy Grid map.

4.4.1 Simultaneous Localization and Mapping

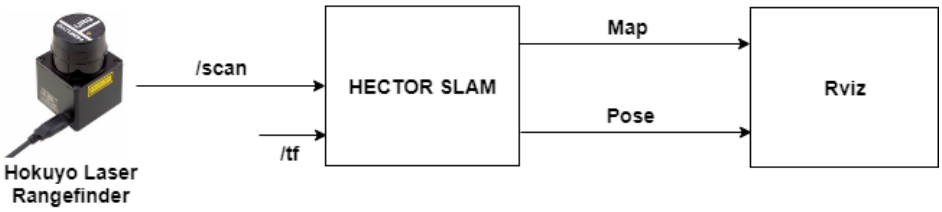


Figure 4.8: Hector SLAM Block Diagram

Simultaneous localization and mapping(SLAM) is a technique by which a mobile robot can build a map of an environment and at the same time localize

itself, that is know where it is located with respect to its environment. SLAM basically consists of two stages - Mapping and Localization. Figure 4.8 shows the block diagram of Hector SLAM[9] where the /scan data from the LiDAR is processed by the Hector SLAM node which outputs both position and map data that can be visualised in a software like Rviz. /tf is given to the Hector SLAM node to keep track of multiple coordinate frames over time or in short to achieve localization. tf keeps track of quintessential 3D coordinate frames over time like world frame, base frame, link frame etc. Figure 4.9 shows a binary occupancy grid representation of a scanned environment. The entire area is divided into a number of cells where each cell holds the probability P_{xy} of cell occupancy.

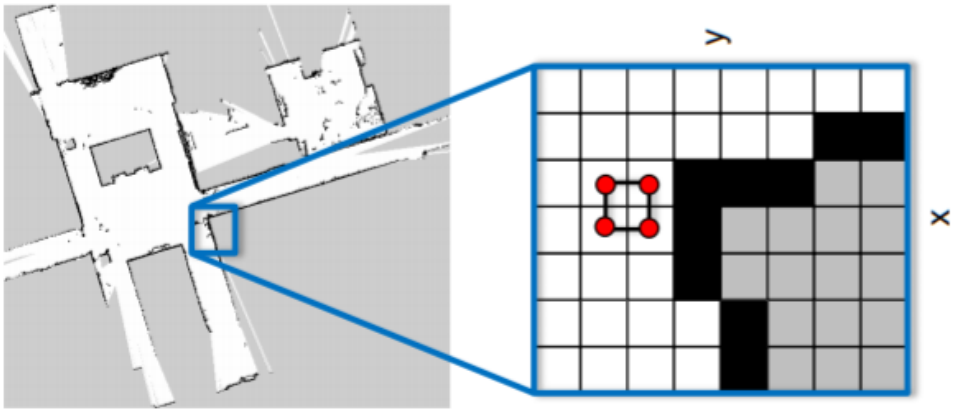


Figure 4.9: Map Representation

4.4.2 Probabilistic Road-map Algorithm

The PRM algorithm is basically a sampling based technique that takes random samples from the configuration space of the robot(set of all possible configurations of robot motions), tests them for whether they are in free space, and connects them to other nearby configurations if a straight line movement is possible without a collision.

The Probabilistic roadmap planner consists of two phases: a construction phase and a query phase. In the construction phase, a roadmap is built, approximating the motions that can be made in the environment. In the query phase,a query asks for a path between two free configurations (q_{init} and q_{goal} in fig 4.8) of the robot. To process the query, the method first attempts to find a path from the start (q_{init}) and goal (q_{goal}) configurations to two nodes of the roadmap (q' and q"). Next, an optimization search is done to find a sequence of edges connecting these

nodes in the roadmap. Concatenation of the successive path segments transforms the sequence found into a feasible path for the robot as shown in fig 4.8.

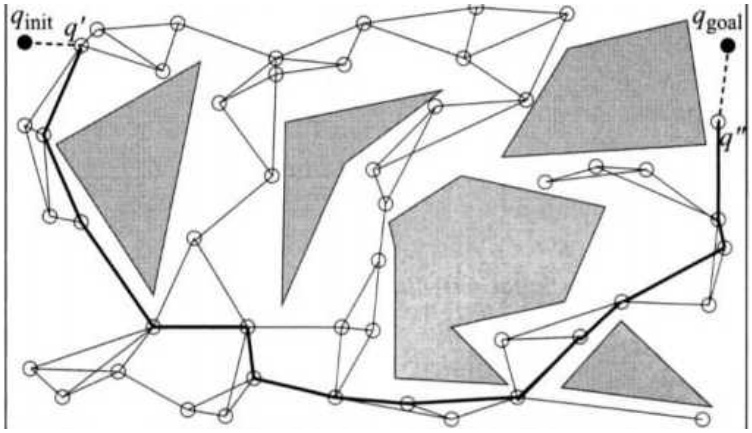


Figure 4.10: Solving a query using PRM Algorithm

The shortest path between start configuration and goal configuration obtained using Dijkstra's shortest path algorithm.

Dijkstra's algorithm.
This algorithm finds the shortest path between two nodes in a given network. The algorithm advances from a node i to an immediately successive node j using a labeling procedure.
The label for a node j is defined as:
$[u_j$,i] = $[u_i+d_{ij}$,i]
where u_i is the shortest distance from node 1 to node i, (d_{ij}) is the length of (i,j)th arc

The node label in this algorithm are of two types: temporary and permanent. A temporary label can be replaced with another temporary label, if shorter path to the same node is detected. The first step in the algorithm is to label the source node with the permanent label [0,-].

Next step is to compute the temporary labels $[u_j$,i] for each node j that can be reached from i, provided j is not permanently labeled. If node j is already labeled as $[u_j$,k] through another node k, and if $u_i+d_{ij} < u_j$, replace $[u_j$,k] with $[u_i+d_{ij}$,i]. If all the nodes are permanently labeled stop the iteration . The shortest path between starting node and desired destination obtained by starting at the desired node and backtracking through the permanently labeled nodes.

Chapter 5

Hardware Design

The fundamental step towards humanoid biped walking is a meticulous hardware design with specific requirements of the robot. Special care must be given for the selection of appropriate actuators as oversized motors increase the total weight of the robot, thereby deteriorating the walking performance.

A 407mm tall structure in standing posture. It has a total of 12 DOF, 6 for each leg. The robot is capable of walking with a maximum stride length of 5 cm.

The structural design and fabrication of the Humanoid was carried out as explained in the Structural Design and Kinematic Analysis of a Humanoid Robot [2].

The design of hardware pertaining to this particular project are as described below.

5.1 Actuators

Actuators used to actuate each joint here are electrical servomotors. Dynamixel AX-12A[11] Series from ROBOTIS Dynamixel range of smart actuators were used. The AX-12A robot servo has the ability to track its speed, temperature, shaft position, voltage, and load.
The control algorithm used to maintain shaft position on the AX-12 actuator can be adjusted individually for each servo, allowing you to control the speed and strength of the motor's response.
All of the sensor management and position control is handled by the servo's built-in microcontroller.

Figure 5.1: Dynamixel AX 12A

5.1.1 Specifications

The specifications of the AX 12A are as follows:

Item	Specification
Baud Rate	7843bps - 1Mbps
Resolution	0.29 degrees
Running Degree	0-300 , Endless Turn
Weight	54.6g
Dimensions (W x H x D)	32mm x 50mm x 40mm
Gear Ratio	254 : 1
Stall Torque	1.5 Nm (at 12V, 1.5A)
No Load Speed	59rpm (at 12V)
Operating Temperature	-5 to 70 C
Input Voltage	9.0 - 12.0V (Recommended : 11.1V)
Command Signal	Digital Packet
Protocol Type	Half Duplex USART (8bit, 1stop, No Parity)
Physical Connection	TTL Level Multi Drop Bus
ID	0 - 253
Feedback	Position, Temperature, Load, Input Voltage, etc
Material	Engineering Plastic

5.1.2 Drawings

The technical drawings of the AX 12A are included here.

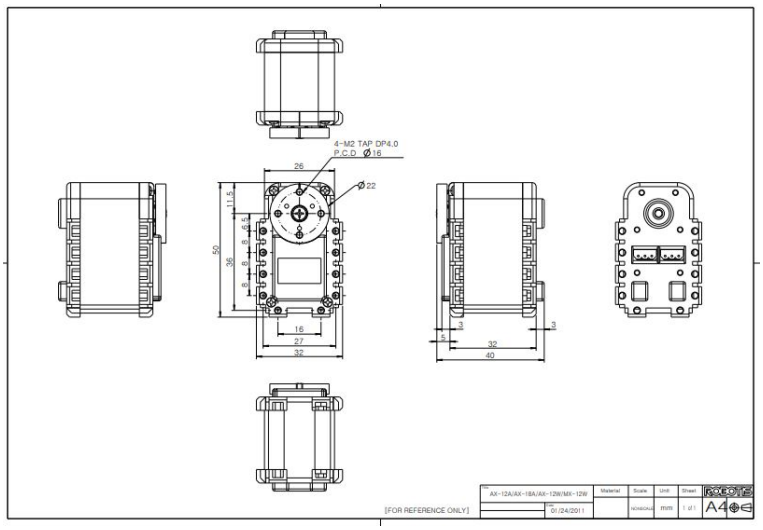


Figure 5.2: Technical drawings of AX 12A

5.1.3 Connection Diagram

The connection diagram of the AX 12A is included here.

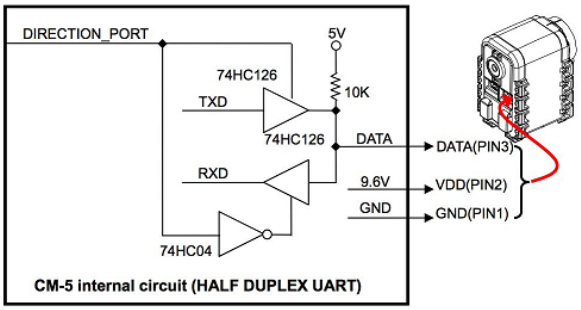


Figure 5.3: Connection diagram of AX 12A

5.2 Controller

An OpenCM 9.04[10] controller running on an ARM Cortex M3 processor was used as the robot controller. The controller is programmed using an OpenCM IDE in Embedded C Language.

Figure 5.4: OpenCM 9.04

5.2.1 Specifications

Item	Description
CPU	STM32F (ARM Cortex-M3)
Operation Voltage	5V - 16V
I/O	GPIO x 26
Timer	4 (16bit)
Analog Input(ADC)	10 (12bit)
Flash	128Kb
SRAM	20Kb
Clock	72Mhz
USB	1 (2.0 Full Speed) Micro B Type
USART	3
SPI	2
I2C(TWI)	2
Debug	JTAG, SWD
Dynamixel TTL BUS 3pin	4
Dimensions	27mm x 66.5mm

5.2.2 Block Diagram

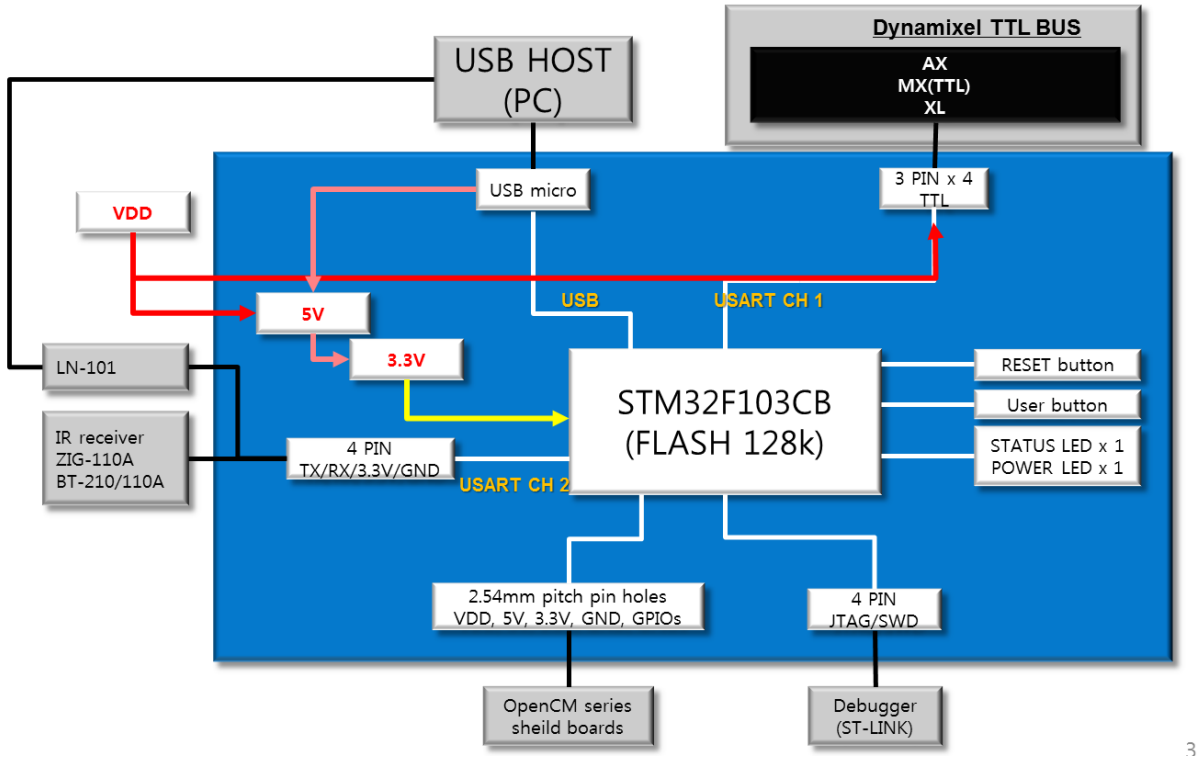


Figure 5.5: Architecture of OpenCM 9.04

5.3 Power Source

The power requirements of the robot is met in any one of the 2 ways mentioned here.
1. Portable Batteries 2. Switched Mode Power Supplies (SMPS)

5.3.1 Battery

The batteries used here are Lithium-Ion(Li-ion) or Lithium-Polymer(Li-po)batteries. Lithium batteries are characterized by their high value of energy density and numerous charge-discharge cycles.
A 11.1V, 20C, 3S, 3000mAh Lithium Polymer battery was used to power the robot.

Figure 5.6: Li-Poly Li-Ion Batteries

5.3.2 SMPS

A switched-mode power supply (SMPS) is an electronic circuit that converts power using switching devices that are turned on and off at high frequencies, and storage components such as inductors or capacitors to supply power when the switching device is in its non-conduction state.
A 13.8V, 15A SMPS was used to carry out the testing of the robot.

Figure 5.7: SMPS

5.4 Developed Prototype

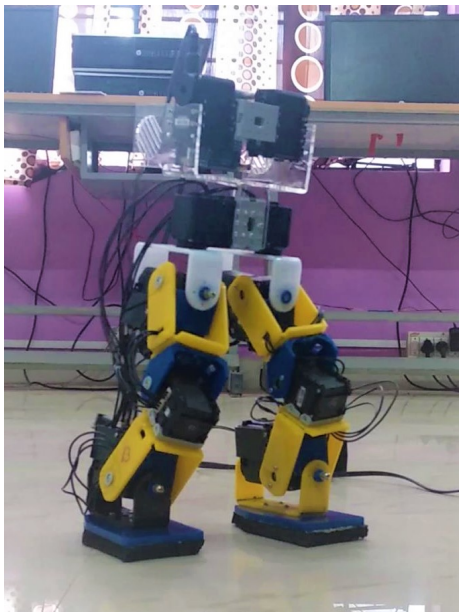


Figure 5.8: Developed prototype

A CAD model of the prototype with six degrees of freedom for each leg was developed in Autodesk Fusion 360. The dimensional accuracy and strength analysis for different parts were carried out. Then the individual parts were fabricated using acrylic laser cutting and 3D printing. The fabricated parts along with motors were assembled accordingly.The AX-12A actuators were connected as per the connection diagram shown in figure 5.3. Battery, SMPS and other control boards were mounted on the assembled model. Thus a prototype as shown in figure 5.8 was evolved.

Chapter 6

Software Design

6.1 Modelling of the Primitive System

As a preliminary analysis and testing phase, the kinematics calculations and walking simulation were done on Prof. Shuuji Kajita's bipedal model, which was imported as a function in MATLAB. The link diagram based on which all the calculations were performed is shown in Figure 4.1 in Chapter 4. The various computations performed and the generation of a walking pattern using the Linear Inverted Pendulum approach are explained in the subsequent sections.

6.1.1 Center of Mass

Computation of center of mass of a link

```
function com = calcCoM()
global uLINK

M = TotalMass(1);
MC = calcMC(1);
com = MC / M;
```

The above code computes the body's center of mass using equation (4.14) by dividing the sum of moments by the total mass M.

6.1.2 Forward Kinematics

Computation of position and rotation matrix of a link

```
function ForwardKinematics(j)
global uLINK

if j == 0 return; end
if j ~= 1
    mom = uLINK(j).mother;
    uLINK(j).p = uLINK(mom).R * uLINK(j).b + uLINK(mom).p;
    uLINK(j).R = uLINK(mom).R * Rodrigues(uLINK(j).a, uLINK(j).q);
end
ForwardKinematics(uLINK(j).sister);
ForwardKinematics(uLINK(j).child);
```

The above code calculates the absolute position and attitude of each link using equations (4.4) and (4.5).

6.1.3 Forward Velocity

Computation of link speed and angular velocity

```
function ForwardVelocity(j)
global uLINK

if j == 0 return; end
if j ~= 1
    mom = uLINK(j).mother;
    disp(uLINK(j).name);
    uLINK(j).v = uLINK(mom).v + cross(uLINK(mom).w, uLINK(mom).R *
        ↪ uLINK(j).b);
    uLINK(j).w = uLINK(mom).w + uLINK(mom).R * (uLINK(j).a * uLINK(j).
        ↪ dq);
    disp(uLINK(j).w);
    end
ForwardVelocity(uLINK(j).sister);
ForwardVelocity(uLINK(j).child);
```

The above code calculates the linear and angular velocity of each link using equations (4.6) and (4.7).

6.1.4 Linear Momentum

Calculation of robot's linear momentum

```
function P = calcP(j)
global uLINK

if j == 0
    P = [0 0 0];
else
    c1 = uLINK(j).R * uLINK(j).c;
    P = uLINK(j).m * (uLINK(j).v + cross(uLINK(j).w, c1) );
    P = P + calcP(uLINK(j).sister) + calcP(uLINK(j).child);
end
```

The above code calculates the robot's linear momentum using equation (4.15).

6.1.5 Angular momentum

Calculation of robot's angular momentum

```
function L = calcL(j)
global uLINK

if j == 0
    L = 0;
else
    c1 = uLINK(j).R * uLINK(j).c;
    c = uLINK(j).p + c1;
    P = uLINK(j).m * (uLINK(j).v + cross(uLINK(j).w , c1));
    L = cross(c, P) + uLINK(j).R * uLINK(j).I * uLINK(j).R' * uLINK(j).w;
    L = L + calcL(uLINK(j).sister) + calcL(uLINK(j).child);
end
```

The above code calculates the robot's angular momentum using equation (4.17).

6.1.6 Inverse Kinematics

IK to find q and $\dot{q}$ using PID controller

```
function InverseKinematicsAll(to, Target)
global uLINK
lambda = 0.9;
```

```
ForwardKinematics(1);
idx = FindRoute(to);
kp = 1;
ki = 0.1;
kd = 0;
for n = 1:10
  J = CalcJacobian(idx);
  err = CalcVWerr(Target, uLINK(to))
  pid = kp*err + ki*error + kd*error;
  if norm(pid) < 1E-6 break, end;
  dq = lambda * (J \ err);
  MoveJoints(idx, dq);
  ForwardKinematics(1);
end
```

The above code uses a PID controller in order to minimise the error between reference and actual position, velocity and angular velocity of each link and returns the joint angle and velocity as output.

6.1.7 ZMP

Calculation of ZMP

```
function [px,py] = calcZMP(c,dP,dL,pz)
global M G

px = (M*G*c(1) + pz * dP(1) - dL(2))/(M*G + dP(3));
py = (M*G*c(2) + pz * dP(2) + dL(1))/(M*G + dP(3));
```

The above code calculates the X and Y components of the Zero Moment Point using equations (4.21) and (4.22).

6.1.8 Linear Inverted Pendulum Model

Generation of instantaneous position and velocity in body frame using Inverted Pendulum dynamics

```
function [px,vx] = LIPM(t,org,x0,Tc)

px = (x0-org) * cosh(t/Tc) + org;
vx = (x0-org)/Tc * sinh(t/Tc);
```

The above code uses the dynamics equation (4.32) to solve for the instanteous position and velocity of the body.
All the above codes were integrated together in MATLAB and a successful walking simulation was implemented which is discussed in Chapter 7 under Software results. The set of instantaneous angles and angular velocities of each link obtained by solving Inverse Kinematics (as shown in Figure 6.6) for one walking cycle were stored in an array and was used for generating a look-up table which was directly fed to the robot. Details of the look-up table are discussed in Section 6.3.

6.2 Modelling of the Actual System

Simulation of the actual robot was performed by importing the CAD model into Simscape Multibody and controlling it using Simulink.

6.2.1 MATLAB and Simscape Multibody

MATLAB is a multi-paradigm numerical computing environment and proprietary programming language developed by MathWorks. MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, C, Java, Fortran and Python.
Simscape $Multibody^{TM}$ (formerly Sim$Mechanics^{TM}$) provides a multibody simulation environment for 3D mechanical systems, such as robots, vehicle suspensions, construction equipment, and aircraft landing gear. We can model multibody systems using blocks representing bodies, joints, constraints, force elements, and sensors. CAD assemblies, including masses, inertia, joints, constraints, and 3D geometry, can be imported into the model. An automatically generated 3D animation helps in visualizing the system dynamics. It also provides some tools to calculate Kinematics and Dynamics.

6.2.2 Simulink Block Diagram

Figure 6.1 shows the outer block diagram where the walking controller block gives the joint angles to the walking robot based on the trajectory and the sensor output which is fed back. The walking robot block is where the CAD model is imported and mechanical modelling is done.

Figure 6.2 shows the Walking Controller which mainly consists of ZMP controller and a Step Planning Logic which plans the next position based on the input trajectory. Based on this position, the Inverse Kinematics block calculates the joint angles which is passed onto the Walking Robot block.

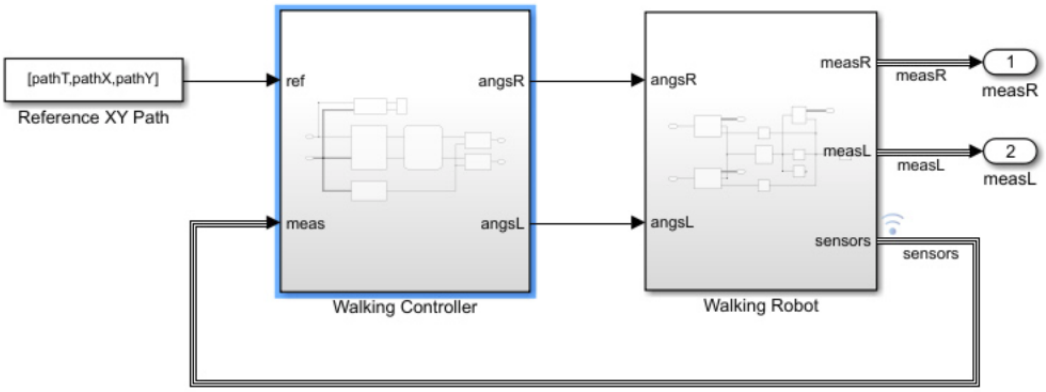


Figure 6.1: Closed loop control of Humanoid

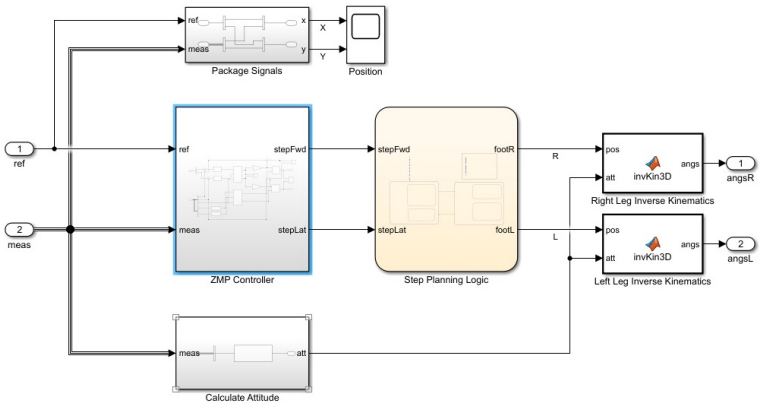


Figure 6.2: Walking Controller

Figure 6.3 shows the Simulink Block diagram for controlling the position of the robot. Based on the reference trajectory and the sensor measurements feedback, an inverted pendulum gain block converts it into instantaneous positions by using the LIPM dynamics equation (4.32) in Chapter 4. The PID controller and filter is used to minimise the error and make the system a closed loop one.

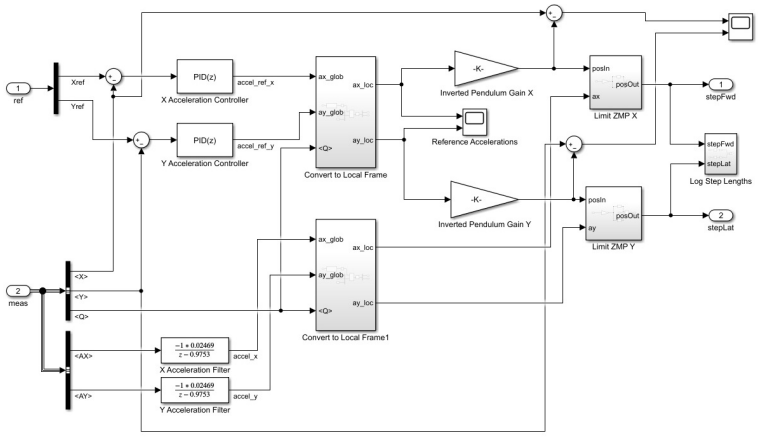


Figure 6.3: Position Controller

6.3 Controller Software

The programming for controlling the hardware was done in OpenCM IDE. The OpenCM IDE is a cross-platform, open-source IDE designed for programming the open-source embedded controller, the OpenCM9.04.
Based on the calculations and simulations done in Section 6.1, a lookup table was generated which is shown.
The code is essentially doing the task of taking the motors from one angle value to another, as obtained from the lookup table. The lookup table(LUT) of each leg for one of the gait cycles is described below.

RHip Roll	RHip Yaw	RHip Pitch	RKnee	RFoot Pitch	RFoot Roll
-3.22 to -14.36	-45.12	-49.8 to -56.25	-79.1 to -92.29	39.26 to 45.7	-3.22 to -10.25
-12.89 to -1.46	-45.12	-62.7 to -60.06	-86.72 to -77.34	36.62 to 29.88	-9.67 to -1.46
0.29 to -5.57	-45.12	-61.52 to -51.86	-77.93 to -71.78	31.93 to 35.45	0.29 to 6.15
-4.69 to -1.46	-45.12	-51.27	-72.07 to -77.34	36.33 to 46.58	5.86 to -1.46
-3.22 to -13.77	-45.12	-43.65 to -62.4	-75.59 to -92.58	47.75 to 46	-3.22 to -9.67
-13.18 to -1.46	-45.12	-65.92 to -62.99	-90.53 to -77.34	40.14 to 29.88	-9.38 to -1.46

Table 6.1: Lookup table for right leg

LHip Loll	LHip Yaw	LHip Pitch	LKnee	LFoot Pitch	LFoot Roll
-3.22 to -14.36	-45.12	-49.8 to -56.25	-79.1 to -92.29	39.26 to 45.7	-3.22 to -10.25
-12.89 to -1.46	-45.12	-62.7 to -60.06	-86.72 to -77.34	36.62 to 29.88	-9.67 to -1.46
0.29 to -5.57	-45.12	-61.52 to -51.86	-77.93 to -71.78	31.93 to 35.45	0.29 to 6.15
-4.69 to -1.46	-45.12	-51.27	-72.07 to -77.34	36.33 to 46.58	5.86 to -1.46
-3.22 to -13.77	-45.12	-43.65 to -62.4	-75.59 to -92.58	47.75 to 46	-3.22 to -9.67
-13.18 to -1.46	-45.12	-65.92 to -62.99	-90.53 to -77.34	40.14 to 29.88	-9.38 to -1.46

Table 6.2: Lookup table for left leg

Firmware

```
#define DXL_BUS_SERIAL1 1 //Dynamixel on Serial1(USART1)

/* Control table defines */
#define GOAL_POSITION 30
int arr1[16][12] = LUT1;
int arr2[32][12] = LUT2;

/* Dynamixel ID defines */

#define RHipY 24
#define RHipR 22
#define RHipP 23
#define RKnee 21
#define RFootP 13
#define RFootR 7

#define LHipY 4
#define LHipR 5
#define LHipP 2
#define LKnee 6
#define LFootP 9
#define LFootR 1


int tf(int angle)
{
  return map(angle,-180,180,0,1024);
```

```
}


 Dynamixel Dxl(DXL_BUS_SERIAL1);

void setup()
{
      // Dynamixel 2.0 Baudrate -> 1Mbps
      Dxl.begin(3);
      Dxl.jointMode(RHipY); //position mode
      Dxl.jointMode(RHipR);
      Dxl.jointMode(RHipP);
      Dxl.jointMode(RKnee);
      Dxl.jointMode(RFootP);
      Dxl.jointMode(RFootR);

      Dxl.jointMode(LHipY); //position mode
      Dxl.jointMode(LHipR);
      Dxl.jointMode(LHipP);
      Dxl.jointMode(LKnee);
      Dxl.jointMode(LFootP);
      Dxl.jointMode(LFootR);

     delay(1000);//wait for servo to move


     Dxl.writeWord(RHipR,GOAL_POSITION,tf(-3.22));
     Dxl.writeWord(RHipY,GOAL_POSITION,tf(-45.12));
     Dxl.writeWord(RHipP,GOAL_POSITION,tf(-49.8));
     Dxl.writeWord(RKnee,GOAL_POSITION,tf(-79.1));
     Dxl.writeWord(RFootP,GOAL_POSITION,tf(39.26));
     Dxl.writeWord(RFootR,GOAL_POSITION,tf(-3.22));

     Dxl.writeWord(LHipR,GOAL_POSITION,tf(-0.6));
     Dxl.writeWord(LHipY,GOAL_POSITION,tf(48));
     Dxl.writeWord(LHipP,GOAL_POSITION,tf(49.51));
     Dxl.writeWord(LKnee,GOAL_POSITION,tf(74.71));
     Dxl.writeWord(LFootP,GOAL_POSITION,tf(-39.55));
     Dxl.writeWord(LFootR,GOAL_POSITION,tf(-0.59));

     delay(1000);//wait for servo to move
```

```
    for(int i=0;i<16;i++)
    {
        Dxl.writeWord(RHipR,GOAL_POSITION,tf(arr1[i][0]));
        Dxl.writeWord(RHipY,GOAL_POSITION,tf(arr1[i][1]));
        Dxl.writeWord(RHipP,GOAL_POSITION,tf(arr1[i][2]));
        Dxl.writeWord(RKnee,GOAL_POSITION,tf(arr1[i][3]));
        Dxl.writeWord(RFootP,GOAL_POSITION,tf(arr1[i][4]));
        Dxl.writeWord(RFootR,GOAL_POSITION,tf(arr1[i][5]));
        Dxl.writeWord(LHipR,GOAL_POSITION,tf(arr1[i][6]));
        Dxl.writeWord(LHipY,GOAL_POSITION,tf(arr1[i][7]));
        Dxl.writeWord(LHipP,GOAL_POSITION,tf(arr1[i][8]));
        Dxl.writeWord(LKnee,GOAL_POSITION,tf(arr1[i][9]));
        Dxl.writeWord(LFootP,GOAL_POSITION,tf(arr1[i][10]));
        Dxl.writeWord(LFootR,GOAL_POSITION,tf(arr1[i][11]));
        delay(78);
    }
}

void loop()
{
    for(int i=0;i<32;i++)
    {
        Dxl.writeWord(RHipR,GOAL_POSITION,tf(arr2[i][0]));
        Dxl.writeWord(RHipY,GOAL_POSITION,tf(arr2[i][1]));
        Dxl.writeWord(RHipP,GOAL_POSITION,tf(arr2[i][2]));
        Dxl.writeWord(RKnee,GOAL_POSITION,tf(arr2[i][3]));
        Dxl.writeWord(RFootP,GOAL_POSITION,tf(arr2[i][4]));
        Dxl.writeWord(RFootR,GOAL_POSITION,tf(arr2[i][5]));
        Dxl.writeWord(LHipR,GOAL_POSITION,tf(arr2[i][6]));
        Dxl.writeWord(LHipY,GOAL_POSITION,tf(arr2[i][7]));
        Dxl.writeWord(LHipP,GOAL_POSITION,tf(arr2[i][8]));
        Dxl.writeWord(LKnee,GOAL_POSITION,tf(arr2[i][9]));
        Dxl.writeWord(LFootP,GOAL_POSITION,tf(arr2[i][10]));
        Dxl.writeWord(LFootR,GOAL_POSITION,tf(arr2[i][11]));
        delay(78);
    }

}
```

6.4 Mapping in ROS

Mapping of an indoor environment is done in Robot Operating System(ROS) using a Hokuyo URG-04LX-UG01 Scanning Laser Rangefinder. The ROS process flow graph is shown in Figure 6.4.

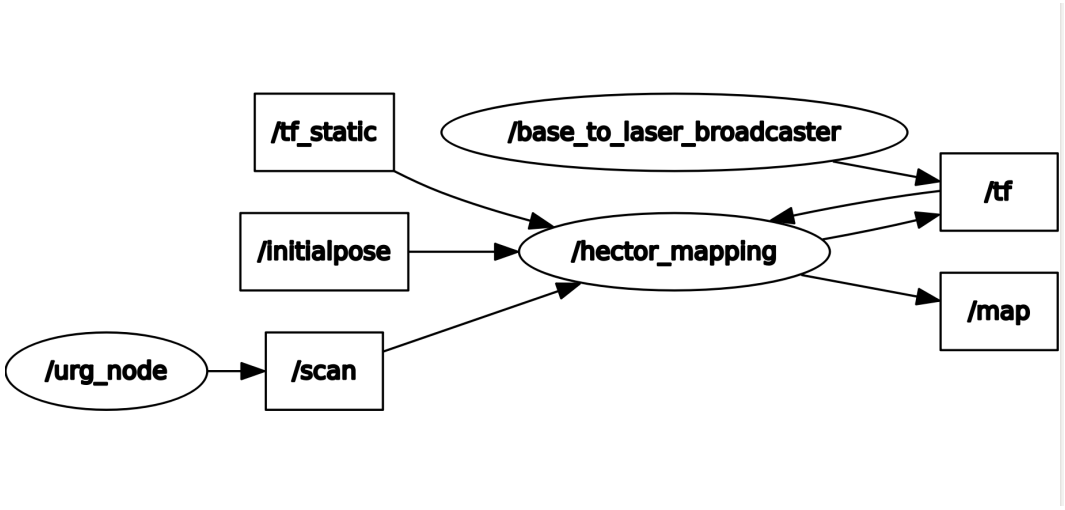


Figure 6.4: Mapping process flow graph

1. `urg_node` - It is the driver node for Hokuyo LiDAR in ROS through which the raw laser scan data is passed on to the `hector_mapping` node via the /scan topic.
2. `hector_mapping` - This node is a SLAM approach that provides 2D pose estimates at the scan rate of laser rangefinders (10 Hz for Hokuyo URG-04LX-UG01).
3. `base_to_laser_broadcaster` - This node uses the /tf topic to transform the coordinates from the base frame to the laser frame so that the data from the LiDAR can be processed by the `hector_mapping` node.

The terminal commands used in Linux for the mapping process are listed below:

1. roscore
2. rosrun `urg_node urg_node _serial_port:=/dev/ttyACM0`
3. `roslaunch hector_mapping mapping_default.launch`
4. rosrun rviz rviz

6.5 Path Planning in MATLAB

Convertion of PGM file to MAT file

```
image = imread('cellarmap.pgm');
imageBW = image < 100;
imshow(imageBW)
map = robotics.BinaryOccupancyGrid(imageBW);
show(map)
filename = 'test.mat';
save(filename)
```

The above code is used to convert a PGM file, which is obtained as an output from ROS to a MAT file. A PGM file is basically a grayscale image encoded with one or two bytes (8 or 16 bits) per pixel. A MAT file is a file in the binary data container format that the MATLAB program uses. "robotics.BinaryOccupancyGrid(imageBW)" creates a grid from the values in matrix imageBW. Each cell in the grid is assigned a value, either 0 or 1.

PRM Algorithm to find shortest path

```
load test.mat;
prmComplex = robotics.PRM(map,150);
show(map)
figure
startLocation = [50 300];
endLocation = [800 300];
path = findpath(prmComplex, startLocation, endLocation);
show(prmComplex)
```

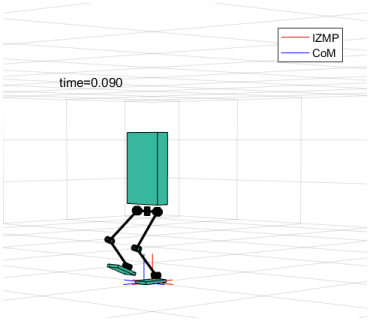
In the above code, robotics.PRM class randomly generates nodes and creates connections between them based on the parameters like start location, end location, connecting distance etc. The findpath() function takes in the node inputs, start and end locations and outputs the shortest path connecting the start and end configurations avoiding obstacles using Dijkstra's algorithm.

Chapter 7

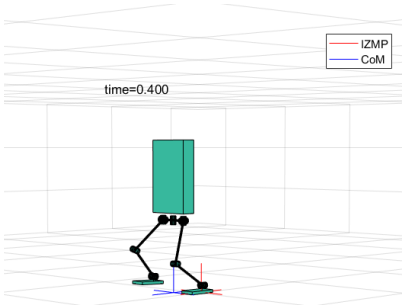
Results and Discussions

7.1 Simulation of the Primitive Model

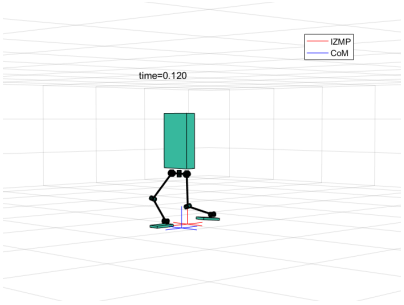
Figure 7.1 shows MATLAB simulation results obtained after kinematic analysis.



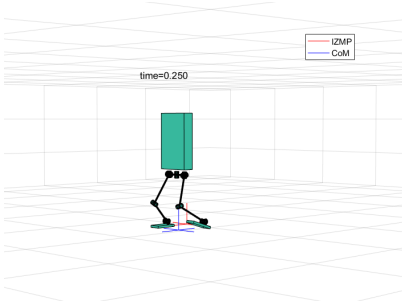
(a) Position one



(b) Position two



(c) Position three



(d) Position four

Figure 7.1: Simulation of the Primitive Model in MATLAB

The plot of left leg joint angles for one walking cycle is shown in Figure 7.2

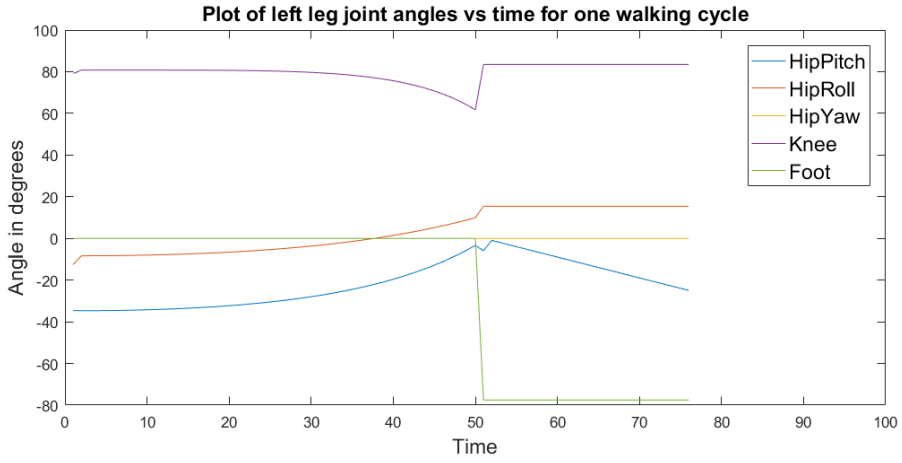


Figure 7.2: Plot of Left Leg joint angles vs time for one walking cycle

The plot of position and velocity of the body's center of mass is shown in Figure 7.3.

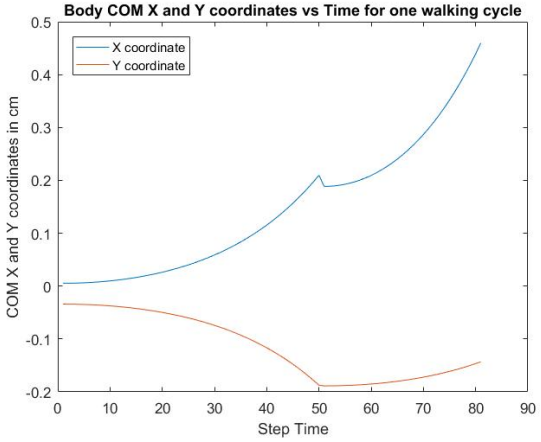
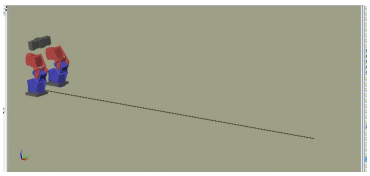
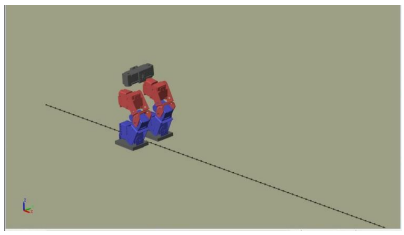


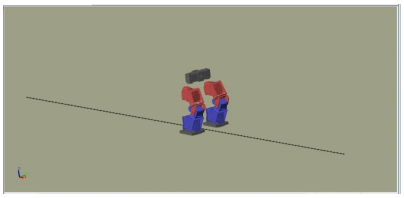
Figure 7.3: Plot of COM position and velocity vs time for one walking cycle



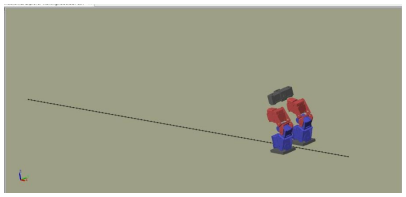
(a) Position one



(b) Position two



(c) Position three

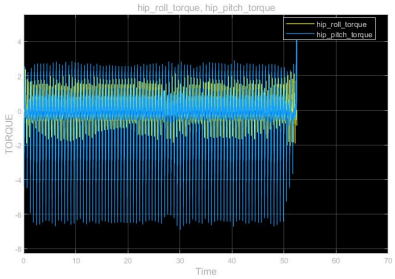


(d) Position four

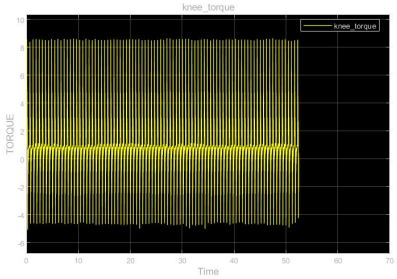
Figure 7.4: SIMSCAPE Multibody Simulation

7.2 Simulation of the Actual Model

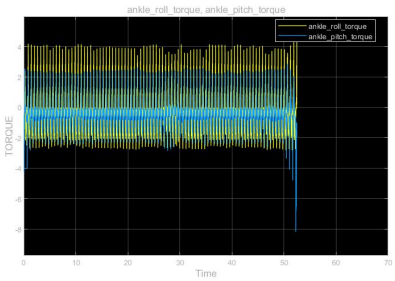
Dynamic simulation done in Simscape Mutlibody resulted in the motion of a model with the same dimensions as our prototype. Different walking postures in Simscape Multibody Simulation is shown in figure 7.4. Figure 7.4(a) shows the model about to follow a straight line trajectory. Figure 7.4(b),(c),(d) depicts snapshots taken sequentially every 1.0[sec] of the model following the straight line trajectory.
The torques at each joint of both legs for one walk cycle is obtained from the simulation. Figure 7.5 shows the plots of joint torques of right leg. Figure 7.5(a) represents the plot of torque at hip joints(roll and pitch), (b) represents the plot of torque at knee joint and (c) shows the plot of torque at ankle joints(roll and pitch).Figure 7.5(d) represents the plot of all the joint torques of right leg taken together.
The plot of ZMP(zero moment point)is shown in Figure 7.6. This is obtained from the position controller shown in figure 6.3. The sensor measured x values and the output from the gain component is given as input to the scope block of position controller to obtain ZMP-X shown in figure 7.6.
A straight line is given as the input trajectory during simulation. The plot of reference trajectory is shown in figure 7.7.



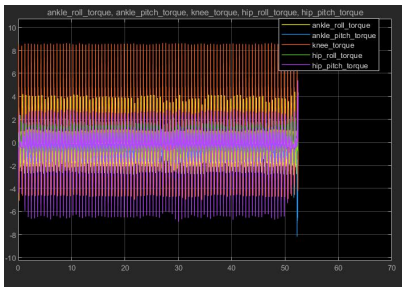
(a) hip torques



(b) knee torque



(c) Ankle torques



(d) right leg torques

Figure 7.5: Plots of joint torques of right leg

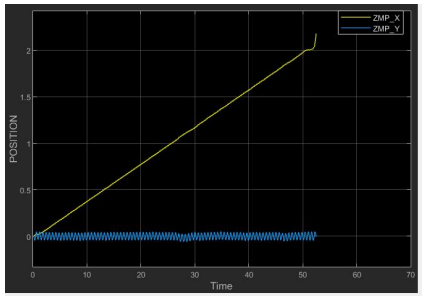


Figure 7.6: The plot of X and Y-components of ZMP

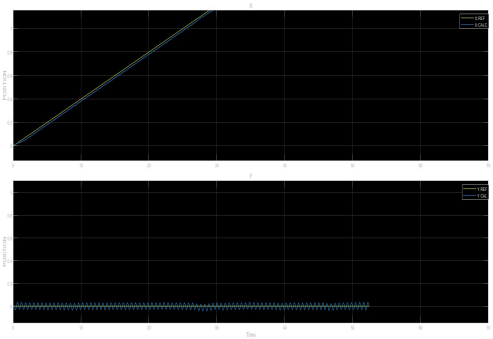


Figure 7.7: Reference Trajectory

7.3 Developed Prototype

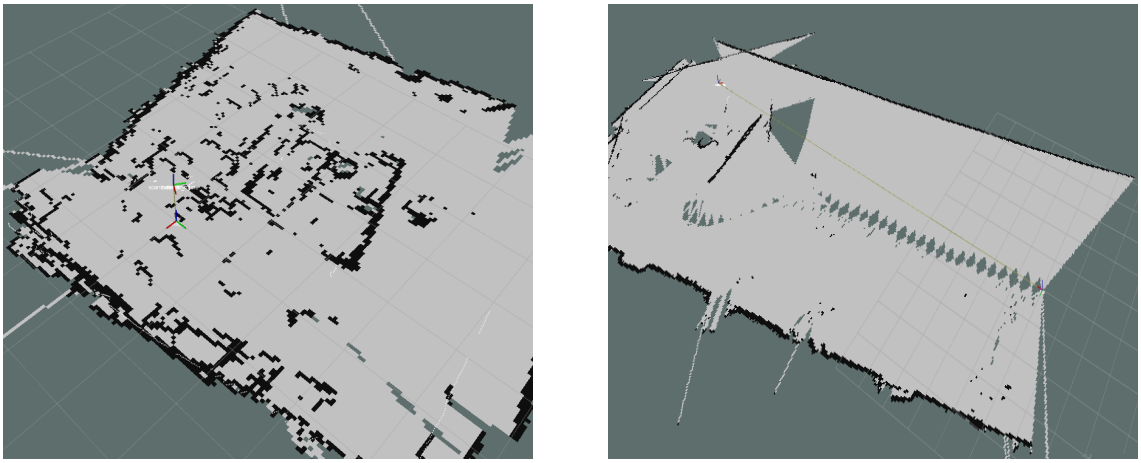
The overall system was designed and implemented structurally. The developed hardware was tested by conducting experiments on a smooth surface. Bipedal locomotion with stability was successfully implemented in the developed prototype. The walking pattern was similar to that obtained in the simulation. Figure 7.8 represents

Figure 7.8: Walking Postures of the Robot

the different postures of the developed robot during one walk cycle. In the initial position both the legs of the robot are in stance position. Then as the walking action starts right foot is lifted a little enabling slight forward motion as depicted in the first picture of figure 7.8. In the second posture both the legs are in stance position but right leg is slightly ahead, while in the third posture right leg is held stationary while left leg is lifted. In the last posture in the cycle both the legs of the robot are in stance position after covering a displacement.

7.4 Map Generation and Path Planning

Motion planning was accomplished by mapping the nearby area of the prototype using LiDAR and a binary occupancy grid based map was developed. The map obtained was given as input to the probabilistic road-map algorithm which produced an optimized collision free path for the robot to reach the desired destination. Several experiments were performed to test the effectiveness of the algorithms. Fig 7.7 shows the mapping of some environments obtained using LiDAR scans in discrete time intervals.



(a) Map of a room

(b) Map of a corridor

Figure 7.9: 2D Maps obtained from LiDAR

Figure 7.10 shows the actual room which was considered for mapping and testing of the shortest path. The binary occupancy grid representation and probabilistic road-map developed in MATLAB is shown in Figure 7.11. It was found that as the number of nodes on the roadmap increases, the efficiency of the path increases, thus giving more feasible paths. Also it is clearly seen that the actual shortest path between the two points in the room matches with the calculated one. Thus the results obtained are satisfactory.

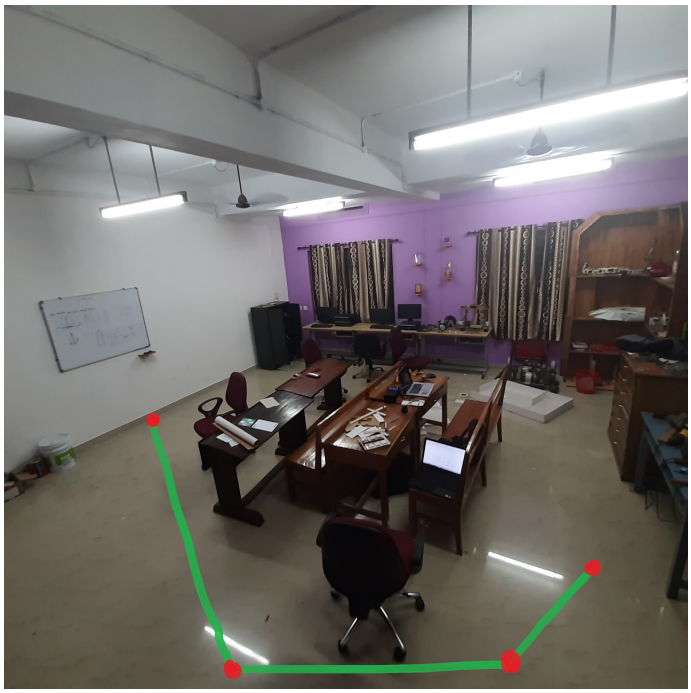


Figure 7.10: Mapping of a room

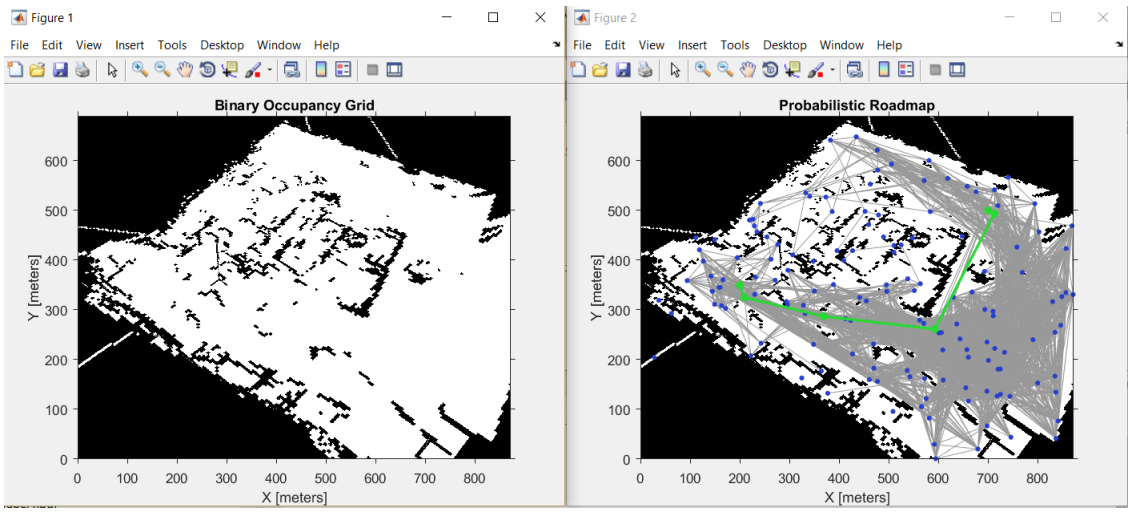


Figure 7.11: Binary occupancy grid and PRM

Chapter 8

Conclusion and Future Scope

Humanoid robots are envisioned as the ideal but probably most complex service robots. Indoor navigation for the visually impaired is a common problem that is addressed by many means conventionally. But with disruption in technology , engineers came up with much more viable and attractive methods like using RFID, Camera based image processing, WiFi etc. But the practicality of these methods are often questioned with respect to their cost, implementation and ease of access. The proposed prototype detailed in this report presents the design and control of a Humanoid with natural walking gait and human interaction capabilities. Kinematic Analysis was performed on a primitive model, as explained in Section 6.1, for generating the lookup table for hardware control. The system kinematics and dynamics equations used for the same are explained in Chapter 4. Using the actuators and controller as explained in Chapter 5, a successful walking gait was implemented in the prototype. The hardware was tested on a smooth surface and it was found to implement bipedal locomotion with stability.

The prototype presented in this report is a scaled down model and is tested only on flat surfaces. Future works include scaling up the model which requires an enhanced controller and actuator like Dynamixel MX 106T having higher torque specifications. Implementation of walking on inclined and rough terrains along with climbing up the stairs has to be achieved. The robot should be made capable to track the shortest path planned, avoiding obstacles, using the algorithm presented in the report, by a proper step planning logic.

There will be a drastic increase in number of humanoid robots over the next decade. According to the Boston Consulting Group, by 2025, robots will perform 25 percentage of all labour tasks. Wide applications of humanoid robots are possible in airport manufacturing facilities, medical industry, military operations at hazardous environments etc. But still humanoid robots are limited in human locomotion, cognition, artificial intelligence and emotions. As technology improves, humanoid robots could be used in newer methods chasing greater potentials.

References

[1] Kajita, S., Hirukawa, H., Harada, K. and Yokoi, K., 2014. Introduction to humanoid robotics (Vol. 101, p. 2014). Springer Berlin Heidelberg.

[2] Zarrugh, M. Y., and C. W. Radcliffe. "Computer generation of human gait kinematics." Journal of biomechanics 12.2 (1979): 99-111.

[3] Hirai, Kazuo, et al. "The development of Honda humanoid robot." Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146). Vol. 2. IEEE, 1998.

[4] Collins, Steven H., Martijn Wisse, and Andy Ruina. "A three-dimensional passive-dynamic walking robot with two legs and knees." The International Journal of Robotics Research 20.7 (2001): 607-615.

[5] Park, Ill-Woo, et al. "Mechanical design of the humanoid robot platform, HUBO." Advanced Robotics 21.11 (2007): 1305-1322.

[6] Mann, Moshe P., Lior Damti, Gideon Tirosh, and David Zarrouk. "Minimally actuated serial robot." Robotica 36, no. 3 (2018): 408-426.

[7] Pratt, Jerry, et al. "Capture point: A step toward humanoid push recovery." 2006 6th IEEE-RAS international conference on humanoid robots. IEEE, 2006.

[8] Kajita, Shuuji, et al. "The 3D Linear Inverted Pendulum Mode: A simple modeling for a biped walking pattern generation." Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180). Vol. 1. IEEE, 2001.

[9] Hector-slam, http://wiki.ros.org/hector-slam

[10] OpenCM9.04, http://support.robotis.com/en/product/controller/opencm9.04.htm

[11] Dynamixel AX-12A Robot Actuator, https://www.trossenrobotics.com/dynamixel-ax-12-robot-actuator.aspx