



PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)
100-ft Ring Road, Bengaluru – 560 085, Karnataka, India

Report on

Dynamic Gait Generation for a Humanoid Using Stereo Vision

Submitted by

Anjana MN (01FB16EEC040)
Ashwathi Subramanian (01FB16EEC051)
Athindra Bandi (01FB16EEC054)

Jan - May 2020

under the guidance of

Prof. M Rajasekar
Associate Professor
Department of ECE
PES University

FACULTY OF ENGINEERING
DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
PROGRAM B.TECH



CERTIFICATE

This is to certify that the Report entitled

Dynamic Gait Generation for a Humanoid Using Stereo Vision

is a bona fide work carried out by

Anjana M N (01FB16EEC040)
Ashwathi Subramanian (01FB16EEC051)
Athindra Bandi (01FB16EEC054)

In partial fulfillment for the completion of 8th semester course work in the Program of Study B.Tech in Electronics and Communication Engineering under rules and regulations of PES University, Bengaluru during the period Jan – May 2020. It is certified that all corrections/suggestions indicated for internal assessment have been incorporated in the report. The report has been approved as it satisfies the 8th semester academic requirements in respect of project work.

<i>Signature with date & Seal</i> <i>Prof. M Rajasekar</i> <i>Internal Guide</i>	<i>Signature with date & Seal</i> <i>Dr. Anuradha M</i> <i>Chairperson</i>	<i>Signature with date & Seal</i> <i>Dr. B K Keshavan</i> <i>Dean – Faculty of Engg. & Tech.</i>

Name and signature of the examiners:

- 1.
- 2.
- 3.



DECLARATION

I, _____, hereby declare that the report entitled, '*Dynamic Gait Generation for a Humanoid Using Stereo Vision*', is an original work done by me under the guidance of **Prof. M Rajasekar**, Associate Professor of ECE Department, PES University, and is being submitted in partial fulfillment of the requirements for completion of 8th Semester course work in the Program of Study B.Tech in Electronics and Communication Engineering.

PLACE: BENGALURU

DATE: DD MMM YY

NAME AND SIGNATURE OF THE CANDIDATE



ABSTRACT

A humanoid robot that is bipedal can be suitable to act as an assistive robot functioning in the human environment. However, the bipedal walk can be very complicated. Just walking on an even floor with absolutely zero inclination and evenness is not satisfactory for the applicability of a bipedal humanoid robot. In this report, we will be presenting a study of bipedal walk on irregular surfaces. The gaits of the existing humanoid models are made to adjust dynamically according to the irregularities in the path. The simulation of the same is carried out on a simulation environment, Gazebo. A 34-degrees-of-freedom (DOF) bipedal humanoid model is used in the Gazebo 3-D simulations. These simulations are carried out on an even floor and inclined surface like stairs of a fixed slope value. The humanoid model comes with an in-built stereo vision module that helps it detect the presence of obstacles around it and the slope value is computed using the simple co-ordinate geometry. This value is provided as an input the servos for the corresponding adjustment of the gaits, without any explicit programming involved



ACKNOWLEDGEMENT

I take this opportunity to express my profound gratitude and deep regards to my guide Prof. M Rajasekar, Associate Professor of ECE Department, PES University, for his exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessings, help and guidance given by him from time to time shall carry me a long way in the journey of life on which I am about to embark.

I would also like to express my gratitude to Mr. Prasanna V for his constant guidance and assistance during the course of this project.

I also take this opportunity to express a deep sense of gratitude to Dr. Anuradha M, Chairperson of the Department of ECE, PES University, for her unconditional support, valuable information and guidance, which helped me in completing this project work through its various stages.

I would also like to extend my gratitude to Dr. Surya Prasad J, Vice Chancellor, PES University for his encouragement and support.



CONTENTS

1. INTRODUCTION	9
1.1 Static vs Dynamic Walking	9
1.2 Humanoid Robots of the Future	12
1.3 Stereovision System for 3-D Perception	14
2. PROBLEM STATEMENT	16
2.1 Motion Planning in Humanoids	16
2.2 The Goal	17
3. LITERATURE SURVEY	18
3.1 Zero Moment Point (ZMP)	18
3.1.1 Support Polygon, Center of Mass	18
3.2 Inverse Kinematics	19
3.2.1 Inverse Kinematics approaches	20
3.3 3D Linear Inverted Pendulum Model	21
3.4 Gait Planning	26
3.5 Simulation Results	29
3.6 Stereo Vision	30
3.6.1 Local Stereo Matching Methods	35
3.6.2 Global Stereo Matching Methods	35
4. IMPLEMENTATION USING ROS	37
4.1 ROS Related Concepts	37
4.2 About HRP-4	39
4.4 Our Simulation	42
4.5 Hardware Implementation	43
5. RESULTS	45
6. CONCLUSION	46
APPENDIX	47



LIST OF FIGURES

Fig 1.1	Implementation of stereo vision on a humanoid
Fig 1.2	Pictures of ASIMO, HRP, Nao, Marty (Top Left- Top Right), Kondo KHR 3HV, Kondo KHR 3HV and Pepper (Bottom left-Bottom Right)
Fig 1.3	Illustration of Stereo Geometry
Fig 3.1	Image to explain CoM, ZMP and Support Polygon
Fig 3.2	Forward and Inverse Kinematics
Fig 3.3(a)	Approximation of a biped humanoid robot as 3D inverted pendulum
Fig 3.3(b)	The defined constraint plane for the 3D LIPM, using the kick force f , The CoM is constrained to the particular plane.
Fig 3.3(c)	Walking pattern based on 3D LIPM
Fig 3.3(d)	foot placements to determine step length and the step width
Fig 3.5(a),(b)(c)(d)	3d LIP Trajectory, Front, Side and Top View
Fig 3.6(a)	Stereo Vision Instrument
Fig 3.6(b)	Disparity/stereo matching
Fig 3.6(c)	Middlebury stereo dataset
Fig 4.2	HRP-4 humanoid
Fig 4.4	Screenshot from the simulation
Fig 4.5(a)	Block Diagram
Fig 4.5(b)	Dynamic Walking
Fig 5.1	Simulation of HRP-4



LIST OF TABLES

Table 3.4	Stances of Robot
Table 4.1	Specifications for HRP-4



1. INTRODUCTION

In this chapter, we give a fair introduction about the difference between static walking and dynamic walking in bipedal humanoids. Also, we will take you through the existing humanoids that were developed by various research groups to test the dynamic walking algorithms and look at the basics of the stereo vision function.

1.1 Static vs Dynamic Walking

Till date, numerous methods to control the gait of a humanoid have been created considering the condition that the strolling surface is completely even and leveled without any slant. Till now millions of bipedal humanoids have walked on properly planned level floors with stability. However, in reality, though a typical room floor appears to be level, it will have nearby and worldwide slants of around 2 degrees at least. It is very crucial to keep in mind that even a little slant in the floor can result in the biped humanoid falling and collapsing.

Numerous and intense research on biped walking robots have been performed since 1970. During that period, biped walking robots have changed into biped humanoid robots through clever and innovative improvements. Moreover, the bipedal humanoid robot become as one of the agents to investigate subjects within the brilliantly robot investigate society. Numerous analysts expect that the humanoid robot industry will be the pioneer industry pioneer of the 21st century and we in the long run enter a period of one robot in each domestic. The solid center on biped humanoid robots comes from a long-established interest for human-like robots. Apart from this, it is really fascinating for a robot to have a structure like a human. This can be needed for a human-robot society. Developing a bipedal humanoid platform is not a very tedious task. However, making this humanoid walk on various types of surfaces with stability has always given a challenge. This is because of a lack of an understanding and knowledge on how human beings walk stably.



Early biped strolling of robots included inactive strolling with a small strolling speed. The step time of the biped humanoid was around 10 seconds per step and the technique to adjust the controls was performed by utilizing the Center Of Gravity (COG). The expected point of COG on the ground continuously seems to be within the support polygon that is made by two feet. While performing inactive strolling, the robot can stop its strolling motion at any point of time without losing its balance and avoiding falling down. The disadvantage of inactive strolling is that the movement is as well moderate and wide for moving the COG. WABIAN-2 of Waseda College, ASIMO of HONDA, and HRP-3 of AIST are some of the famous bipedal humanoid robots.

Till this date, many bipedal humanoid robots have given a steady energetic gait on well-designed, leveled floors. However, strolling on the uneven and slanted floors is still too early to be achievable and requires a lot of study. Hence, researchers began to learn more about dynamic walking of bipedal humanoids. It is nothing but a fast walk with a speed of about 1 second per 1 step. Once we make sure the equilibriums are maintained often, dynamic walking will be much smoother and active even while making tiny body movements. But, if the inertia generated from the acceleration of the humanoid's body isn't controlled properly, a bipedal robot will easily collapse. Apart from this, a biped humanoid has an increasing chance of falling down from disturbances like an unexpected obstacle and can't stop the walking motion suddenly during dynamic walking. So for this reason, the concept of Zero Moment Point (ZMP) was introduced to make sure inertial forces are regulated.

So what is ZMP? In static state, a point on the ground is predicted to be the COG based on some calculations. This is the Zero Moment Point or the ZMP. In the dynamic state, the entire force of inertia which comprises of gravity and mass goes through this point. If the ZMP continuously falls within the supporting polygon made by the humanoid feet, it will never fall down. Billions of research groups use ZMP as a criterion for stable walking of bipedal dynamic walking and the robot is controlled so that the ZMP is maintained within the supporting polygon.



Dynamic walking on a non-leveled, rough floor by a bipedal humanoid robot is really difficult to implement and appreciate. This is because majority of bipedal humanoid robots use motors and gearing to perform hard position control of the joints. Hence the response times of the actuators and sensors are low as a result of the reduction gear and sensor noise. So, it's not realistic for the robot to live the ground conditions instantaneously. Also, it's not feasible for the robot to respond properly even if the ground conditions are measured instantaneously. The human ankle can instantly adapt to changing ground conditions on the other hand.

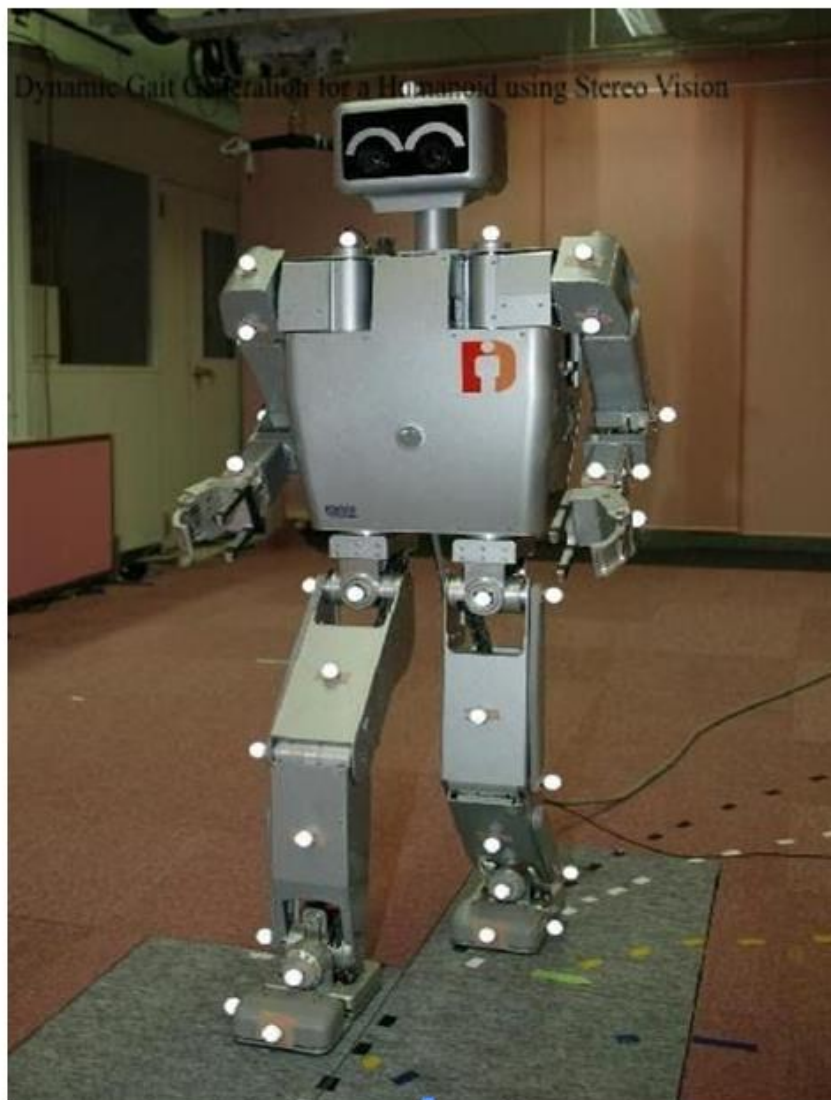


Figure 1.1: Implementation of stereo vision on a humanoid



1.2 Humanoid Robots of the Future

Dynamic walking, empathizing and grasping an object are some of the hardest problems in robotics that involve trying to replicate things that humans can do easily. The goal has always been to create a general purpose robot rather than a specialized one for specific industrial machines. Here are the six existing robots that point the way towards the humanoid robots of the future.

1. **ASIMO**: Asimo (Advanced Step in Innovative Mobility) was originally built for the Darpa Robotics Challenge. It was developed by Boston Dynamics. The main aim of the ASIMO humanoid is to balance and move around like a normal human. This is achieved by using AI. [7] Currently, it can be seen displayed in the Miraikan Museum, Tokyo, Japan.
2. **Pepper**: Pepper can serve as a daily companion and a customer assistant. Soft Bank developed it. Pepper makes an effort to identify and give an appropriate response to human emotions. Pepper definitely appears like a robot rather than a human. However, to provide a natural and more intuitive feeling, Pepper uses its body movements and tone of voice.
3. **HRP(Humanoid Robotics Platform)**: HRP robots are available in many versions. HRP-2 was designed and integrated by Kawada Industries, Inc. It is 154 cm tall and has a weight of 58 kg with 30 Degrees of Freedom. The HRP robot is widely used in educational institutions for teaching students the concepts that were used to make this humanoid. HRP-3 was developed as a research and development of platform for the New Energy and Industrial Technology Development Organization (NEDO). It has a better structure and system than HRP-2. It is 160 cm tall and has a weight of 68 kg with 42 Degrees of Freedom. The latest humanoid in HRP series is HRP-4. It is a combination of the main features of HRP-2 and HRP-3.
4. **Nao**: Soft Bank Robotics was instrumental in developing this humanoid. It is of the size of a kid and is 58 cm tall and has a weight of about 4 kg. This humanoid looks very fascinating and it has a brawny structure. Hence, it is also used as a platform to work on. It has 27 Degrees of Freedom. Thus, it is capable of performing yoga asanas. This humanoid has been selected to be



given to the RoboCup Standard Platform League contestants

5. **Marty**: This is a very tiny humanoid. ROBOTICAL came up with it. It has got 8 degrees of freedom and is 3D printable. It is also not very difficult to be built.
6. **KHR Kondo series**: This series of humanoids are also small in size. They are mainly used in research. KHR Kondo 2HV and KHR Kondo 3HV are 2 versions which are available. The only difference between the two is that the KHR Kondo 3HV has 2 extra degrees of freedom in its legs. Thus, it can turn with ease.

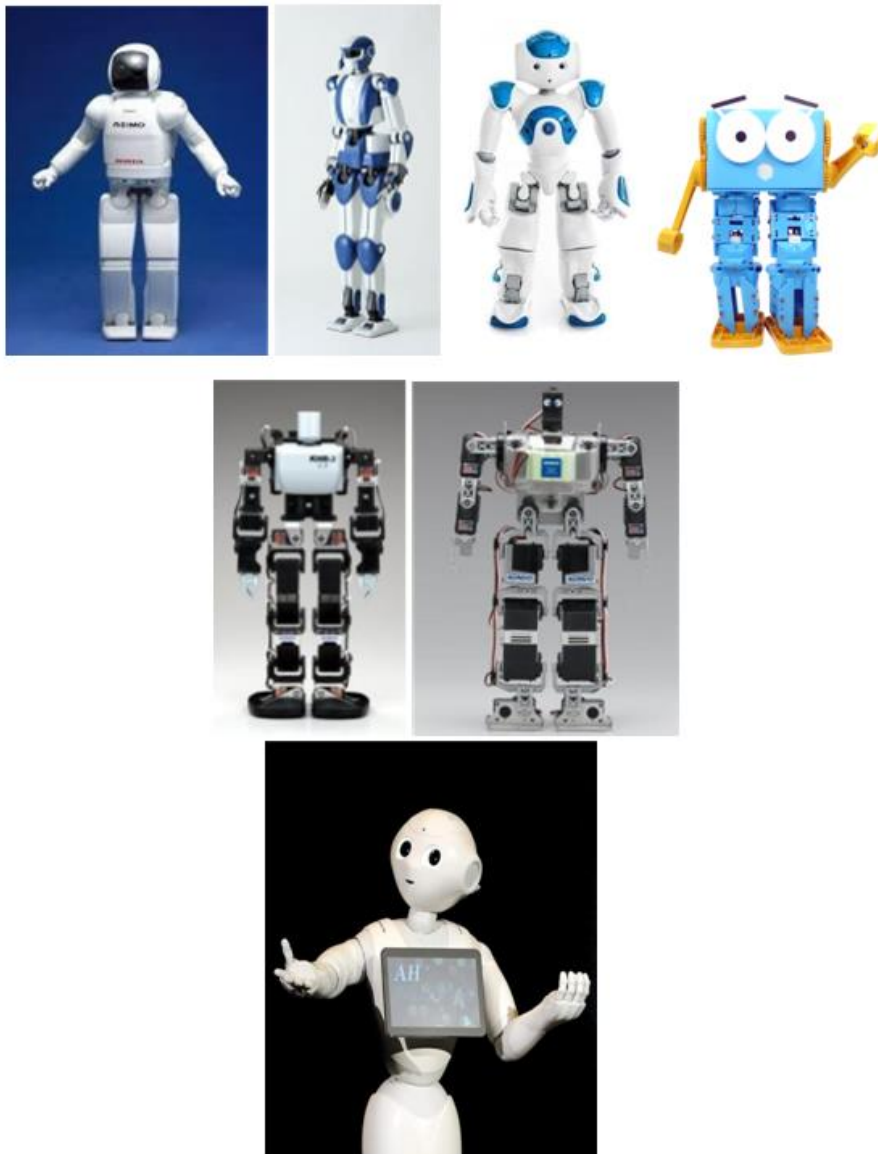


Figure 1.2(a)(b)(c)(d)(e)(f)(g): Pictures of ASIMO, HRP, Nao, Marty (Top Left- Top Right), Kondo KHR 3HV, Kondo KHR 3HV and Pepper (Bottom left-Bottom Right)



1.3 Stereovision System for 3-D Perception

In recent times, real-time camera monitoring from distant, isolated applications has had a lot of scope because of its capacity to strictly watch and monitor any place. But, there is a scarcity of realistic view because only 2D images are produced that don't contain any depth information and natural motion control.

The issue in stereo vision is based on the physical phenomena, that a 3-D object has different projections depending on the point of view. In this manner, it is conceivable to recreate the 3-D scene from at slightest two pictures of the same protest taken from two unmistakably, unique and distinct focuses[2]. Then comparing pixels from one picture are found within the other one and concurring to this data the difference outline is built. Afterward, the 3-D scene is recreated agreeing to the dissimilarity outline. The most complicated issue in stereo computation lies in plan of comparing focuses and points searching metrics and algorithms[2].

Stereo application takes the input as stereo sets of 2-D pictures and produces the recreated 3-D information by finding the comparing focuses. The comparing issue requires a particular looking and coordinating method, the strength of which decides quality and accuracy of reproduced 3-D information .Most stereo recreation strategies are based on the utilization of the pinhole camera demonstration and parallel geometry. In these methods, two points are selected from each of the images, left and right, by using any procedure for stereo matching. Then the depth of the image will be calculated by looking at how inconsistent the two points are from each other(disparity). Depth can also be found by the partition of the two pixel points.[2].

If (x_L, y_L) are the pixel coordinate in the left image and (x_R, y_R) are the pixel coordinates in the right images, then the 3D world coordinates (X, Y, Z) are computed as follows :

$$X = x_L b / d$$

$$Y = y_L b / d$$

$$Z = fb / d$$

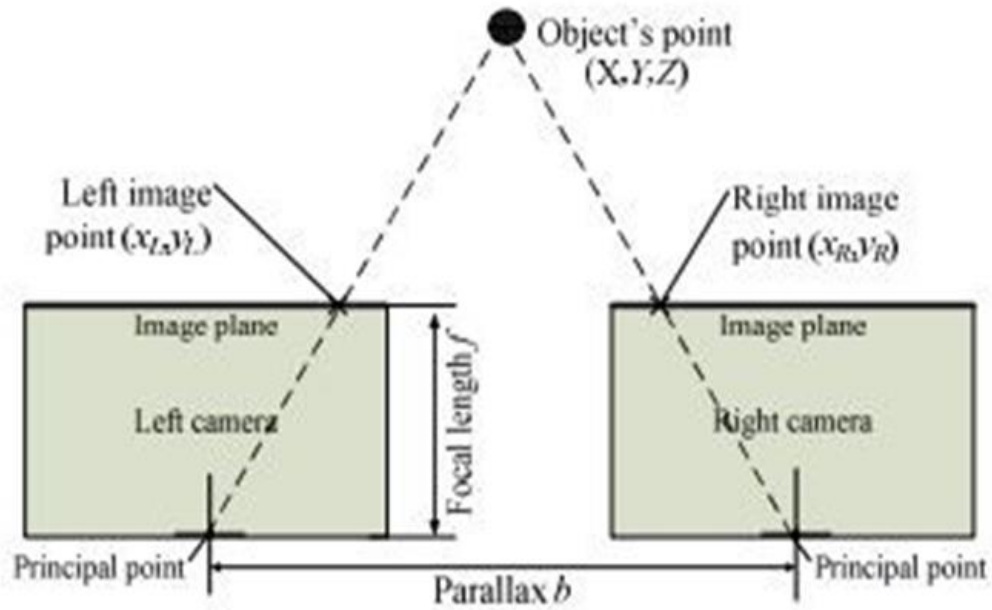


Figure 1.3 Illustration of Stereo Geometry



2. PROBLEM STATEMENT

The human walk may be a monotonous process of ‘tilt over’ or unsteady movements that might in some cases cause a sound person to fall down while walking over an uneven surface. The first time the walking process of humans came into the picture was during the First World War while creating replacement legs for impaired workers and soldiers. To mimic human-like walking trajectory in a humanoid, especially over the inclined and uneven planes, is something challenging and unique when it comes to implementing the cost-effective and dynamic solution.

2.1 Motion Planning in Humanoids

The two of the most important features of bipedal humanoid robots are the shape and movements that are very much human-like. The Bipedal humanoid robots consists of two human-like legs. These legs will help the humanoids walk with stability and they should be capability to be on various types of floors such as uneven floors or steps also. Hence, for this reason, numerous researchers throughout the world have studied on bipedal dynamic walking and have come up with many robotic platforms. Two main methods of using motion planners to generate dynamically balanced robotic motions.

- 1) In one of the methods, the humanoid movements are preplanned. The area of the space that needs to be explored can be decided based upon the speed of the humanoid and the position of its footprints.
- 2) In another method, a geometric route is decided. The humanoid is supposed to follow this geometric path. In the second step, the geometric path is estimated again.



2.2 The Goal

The goal of this project can be defined as,

“Dynamic Gait Generation for a Humanoid Robot using Stereo Vision”

The humanoid will be made to walk in an environment with a few obstacles like steps or cuboids. The stereo vision will come up with the computation of the fixed slope angle and the humanoid will adjust its gaits accordingly and walk on the steps. All of the motions are shown by proof using the ROS Gazebo simulation environment.



3. LITERATURE SURVEY

This venture came in reaction to realize dynamic gait for small sized humanoid robots. This process includes utilizing the stereo vision yield (the slant point) as the input to the as of now existing model, so that the pre-planned walks can be self-adjusted such that the humanoid can walk with stability. Zero Minute Point (ZMP) and support polygon play critical parts in guaranteeing the steadiness and stability[1][4].

3.1 Zero Moment Point (ZMP)

Achieving stability in locomotion is a challenging aspect in humanoid robotics. Zero Moment Point can be defined as a point that lies on the humanoid's foot area through which the resultant passes[1].

3.1.1 Support Polygon, Center of Mass

Support Polygon is an important concept related to ZMP. Support Polygon is the region formed by enclosing all the contact points of the robot and ground using some elastic cord. A point is called center of mass if it moves in the direction of the force without rotating.

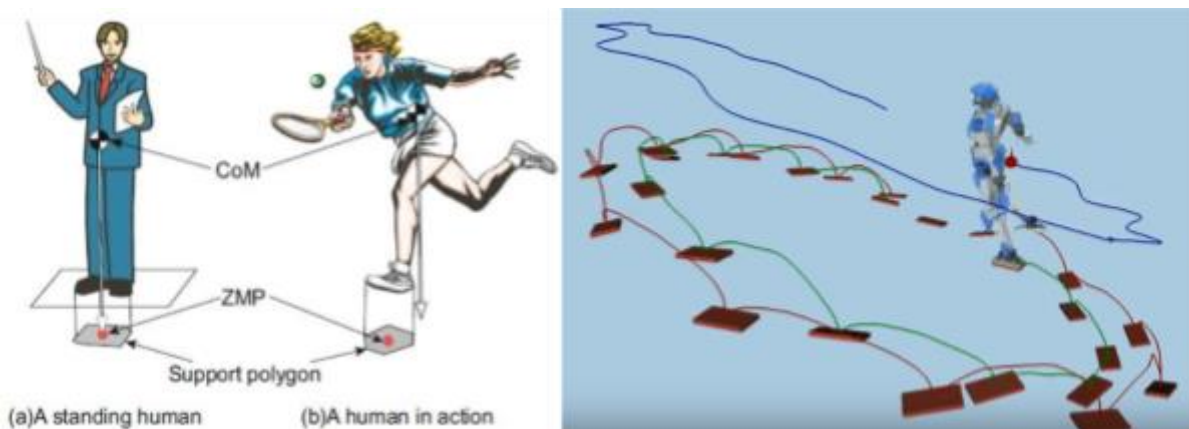


Fig 3.1 (a) : Image to explain CoM, ZMP and Support Polygon

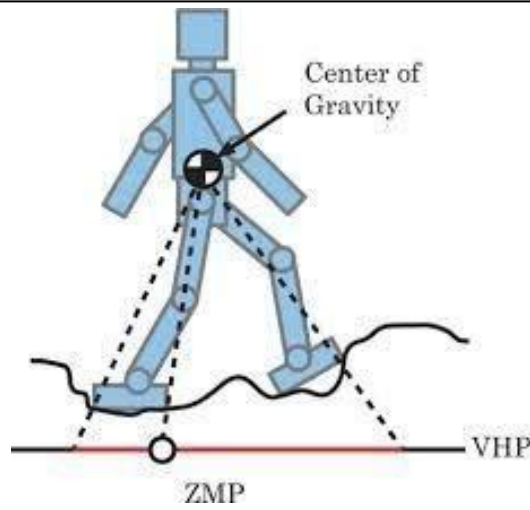


Fig 3.1 (b) : Image to explain CoM, ZMP and Support Polygon

3.2 Inverse Kinematics

Kinematics is the theory which analyses the interrelation between the joint angles and position and attitude of a link [4]. Kinematics lays a basic foundation for robotics. It includes mathematics which deals with rendering a dynamic object in 3D space. If the joint angles are known, and the final orientation of the robot tool tip is calculated, it's known as Forward Kinematics. If the required position is known and the joint angles need to be calculated, it's called Inverse Kinematics. Inverse kinematics is a field where the constraints define whether or not the equations are solvable. Hence, the approach in representation of links of a robot decides the effectiveness and solvability of the problem.

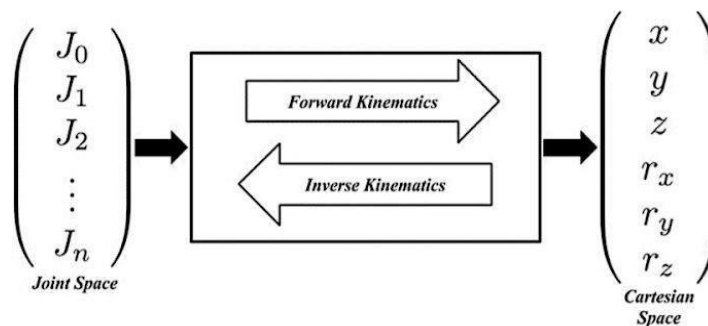


Figure 3.2: Description of Forward and Inverse Kinematics



3.2.1 Inverse Kinematics approaches

The different approaches to the inverse kinematics problem will depend on the construction of the mechanism. Not all approaches have solutions to all the mechanisms. The choice of approach depends on the way the robot links are joined and constructed[1].

1. **Pseudo Inverse of Jacobian**

This method involves using a pseudo inverse of a Jacobian matrix of the homogeneous transformation matrix. The homogeneous matrix is a series of multiplied rotation matrices and translation matrices in the order all the links are placed. It's based on the assumption that the end-effector reaches the goal position in incremental steps. The Jacobian is rarely square and hence the pseudo inverse issued.

2. **Denavit-Hartenberg Method**

The Denavit-Hartenberg method uses 4 key parameters to represent each link. Using these parameters the entire homogeneous transformation matrix can be calculated. From the transformation matrix, the Jacobian can be calculated and using pseudo inverse method, the joint angles can be determined.

3. **Geometric Approach**

Sometimes when the joint structure is simple, it is possible to find out the joint angles using a geometric approach using positions of each link and their constraints.



3.3 3D Linear Inverted Pendulum Model

3D linear Inverted Pendulum Model is an algorithm to generate a gait pattern that realizes the biped gait by producing a track. A gait pattern is nothing but a set of joint angles to achieve the required gait [3].

The 3D Inverted Pendulum Model consists of a few assumptions. They are listed down below:

1. The entire mass of the robot is concentrated at its center of mass.
2. It is assumed that the legs of the robot are mass-less. The tips of the leg contact the ground at single rotating joints.
3. The end of the joint is only capable of moving in a single constrained plane, which has to be parallel to the ground.

The important parameter in the 3D LIP model is R – supporting distance, G – gravity, f – kickforce.

The kick force f can be resolved into components f_x , f_y , f_z

$$f_x = (x/r)f \quad \text{..3.3.1}$$

$$f_y = (y/r)f \quad \text{..3.3.2}$$

$$f_z = (z/r)f \quad \text{..3.3.3}$$



The robot's leg is assumed to be extensible using the kick force which makes sure the CoM stays on the same plane. The only forces that indeed action CoM are the kick force and the gravitational force and, hence we get the motion equations as follows.

Introduction of a constraint plane defined as shown in figure 3.3. Where slopes $k_x k_y$ give slopes and z_c gives height of the constraint plane.

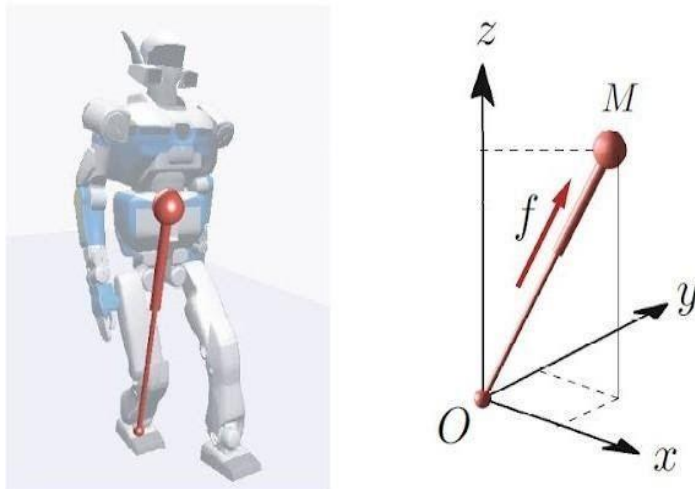


Figure 3.3(a) Approximation of a biped humanoid robot as 3D inverted pendulum

$$M\ddot{x} = \left(\frac{x}{r}\right)f \quad M\ddot{y} = \left(\frac{y}{r}\right)f \quad M\ddot{z} = \left(\frac{z}{r}\right)f - Mg$$

Acceleration of the CoM has to be orthogonal to normal vector of the constraint for CoM to move along the constraint plane. The CoM is restricted to move only in the defined plane, and to maintain it, enough kick force must be applied. The slope of the plane can be manipulated to make the robot climb inclined plane or a stairway

$$z = k_x x + k_y y + z_c$$



Solving equations and making proper substitution, we get

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

Using this kick force equation, we can easily tell that, the entire model depends on physical parameters of the robot and not on other parameters.

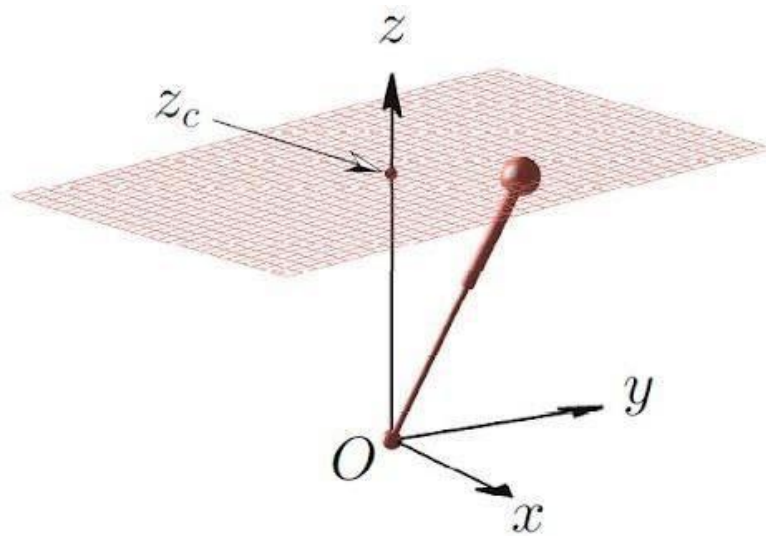


Figure 3.3(b): The defined constraint plane for the 3D LIPM, using the kick force f , the CoM is constrained to the particular plane.

Derivation of horizontal dynamic equations of the CoM is as follows

$$\ddot{x} = \frac{g}{z_c} x \quad \text{.. 3.3.9}$$

$$\ddot{y} = \frac{g}{z_c} y \quad \text{.. 3.3.10}$$



These equations give the acceleration required for the CoM in the plane. Hence we call it as 3D linear inverted pendulum as the acceleration is linear which can be inferred by the equation.

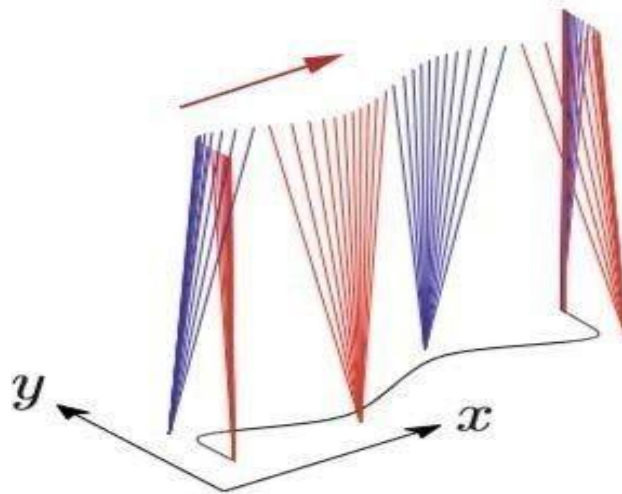


Figure 3.3(c) :Walking pattern based on 3D LIPM

The geometric representation of 3D linear Inverted Pendulum Model is a hyperbola as shown in the above figure. The overall motion of the CoM is given by the equation 3.2.11. In the figure, it's evident that the CoM trajectory is continuous but the foot placements cannot be.

$$\frac{g}{2z_c E_x} x^2 + \frac{g}{2z_c E_y} y^2 + 1 = 0 \quad \text{.. 3.3.11}$$

In the figure 3.7, the red line indicates the way left and right should interchangeably move to mimic a walking sequence. The algorithm basically shifts the CoM over either of the legs and makes the other leg swing to the next position. This shift in CoM and ZMP has to be carefully calculated using the actual robot's parameters.



In practical situations, we need to directly specify the foot placements which can be represented by using length of the step s_x and width of the step s_y . The foot placements can be represented (p_x, p_y) from which we can determine walk primitive of n^{th} step. The placement is given by the equation below.

$$\begin{bmatrix} \bar{x}^{(n)} \\ \bar{y}^{(n)} \end{bmatrix} = \begin{bmatrix} \frac{s_x^{(n+1)}}{2} \\ (-1)^n \frac{s_y^{(n+1)}}{2} \end{bmatrix}$$

..3.3.12

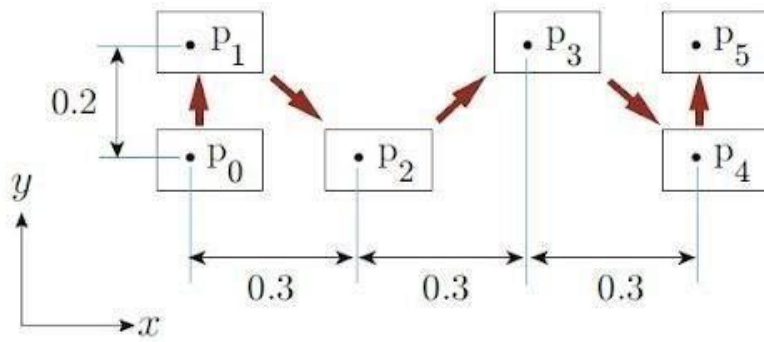


Figure 3.3 (d) :foot placements $p_0 \dots p_N$ from which determine step length and the step width

Since the placements of the humanoid foot are preplanned, we can obtain the position of the end effector directly. The trajectory of the center of mass can also be estimated by using the eqn. 3.3.12. Now since we know the position of the end effector and the CoM, the orientation of the rest of the links and can also be estimated easily. Now on applying the theory of inverse kinematics, we will be able to come up with the orientation of the joints throughout the complete span of the humanoid's locomotion. This will provide us the sequence of angles. The humanoid uses this sequence of angles in order to execute its gait.



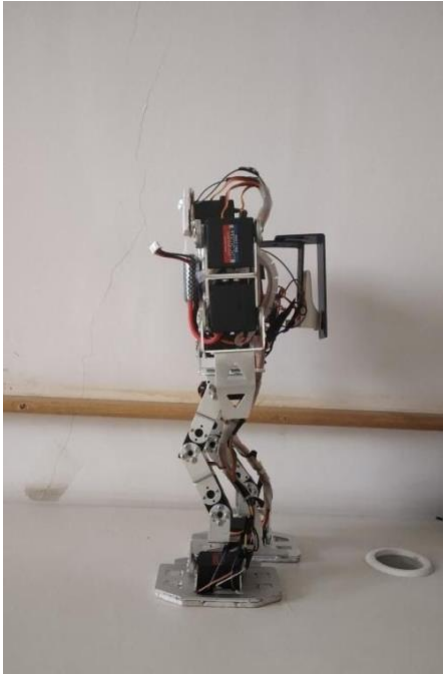

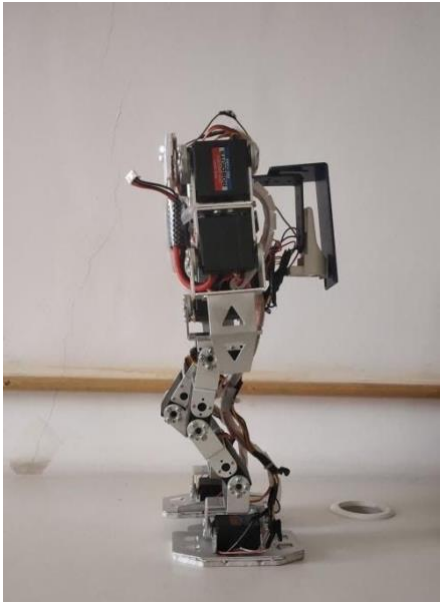
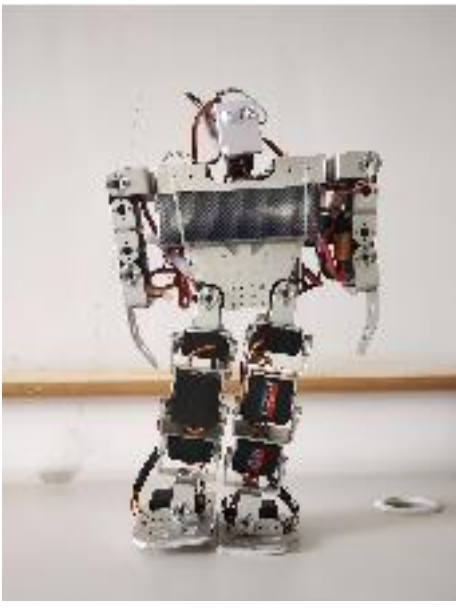
The 3D LIP model is stable walking algorithm in itself. This algorithm, unlike the various other algorithms, doesnot make use of the Zero Moment Point for better gaits.

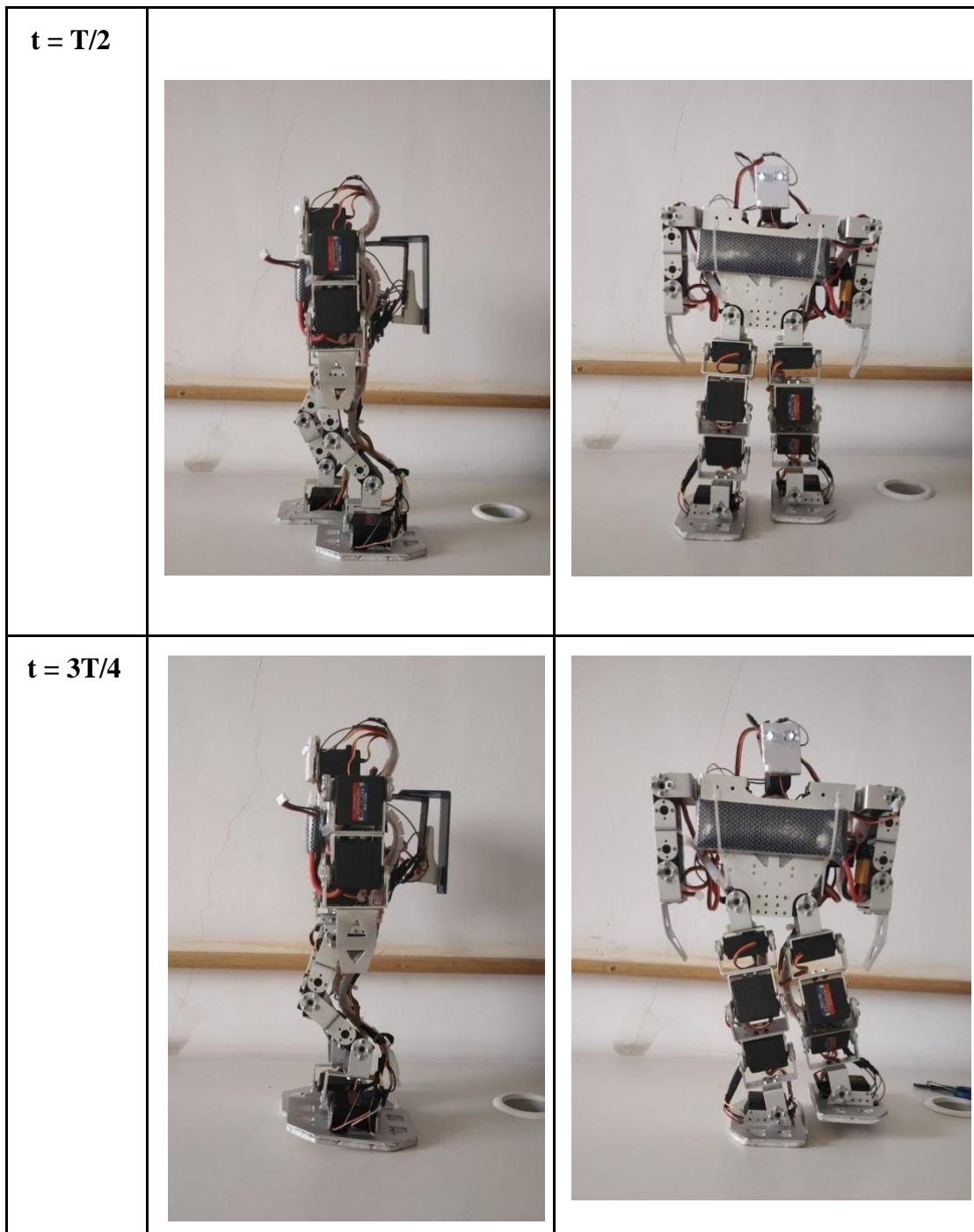
3.4 Gait Planning

The orders of the angles generated above were implemented on the humanoid. The result was successful execution of the gait of the humanoid. The gait was very much like to a human's gait. However, there were some problems with the humanoid's weight that resulted in slight inertia. However, this did not affect the gait.

The table below displays humanoid stances at important points during the gait span [1].



Time	Side View	Front View
$t = 0$		
$t = T/4$		



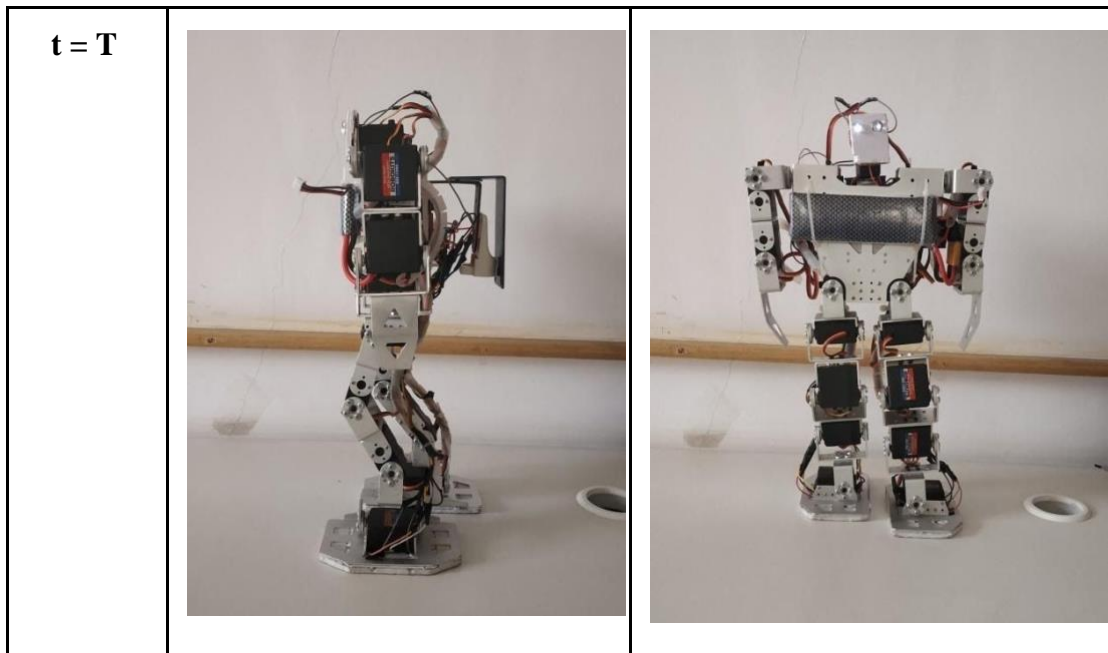


Table 3.4: Stances of robot at $t=0, T/4, T/2, 3T/4$ and T

3.5 Simulation Results

Initially, a python simulation was run on the 3D LIP model. The matplotlib package was used to plot the trajectories. The figures below display the simulated plots obtained using python. The figure 3.5 displays the gait that was produced by using the 3D LIP model. The blue color in the support phase indicates the right leg. Green indicates the left leg green whereas, right leg in swing phase is indicated by red and left leg by yellow [1]. The

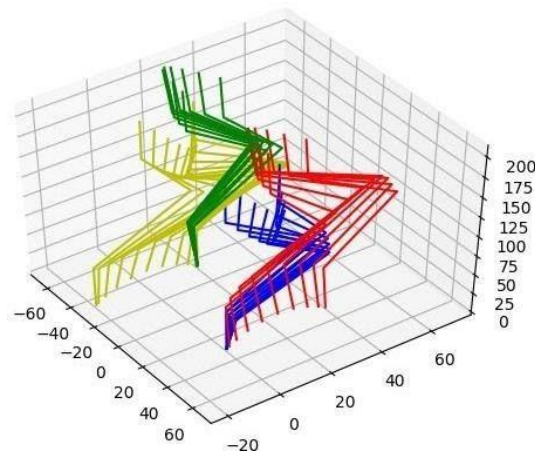


Figure 3.5(a): 3D LIP trajectory



figure 3.5 is the entire gait in 3D view. The graphs displayed below provide the simulated results of the gaits from various views. The next figure 5.1(d) indicates the plot of every link angle that right leg has over the entire time period T .

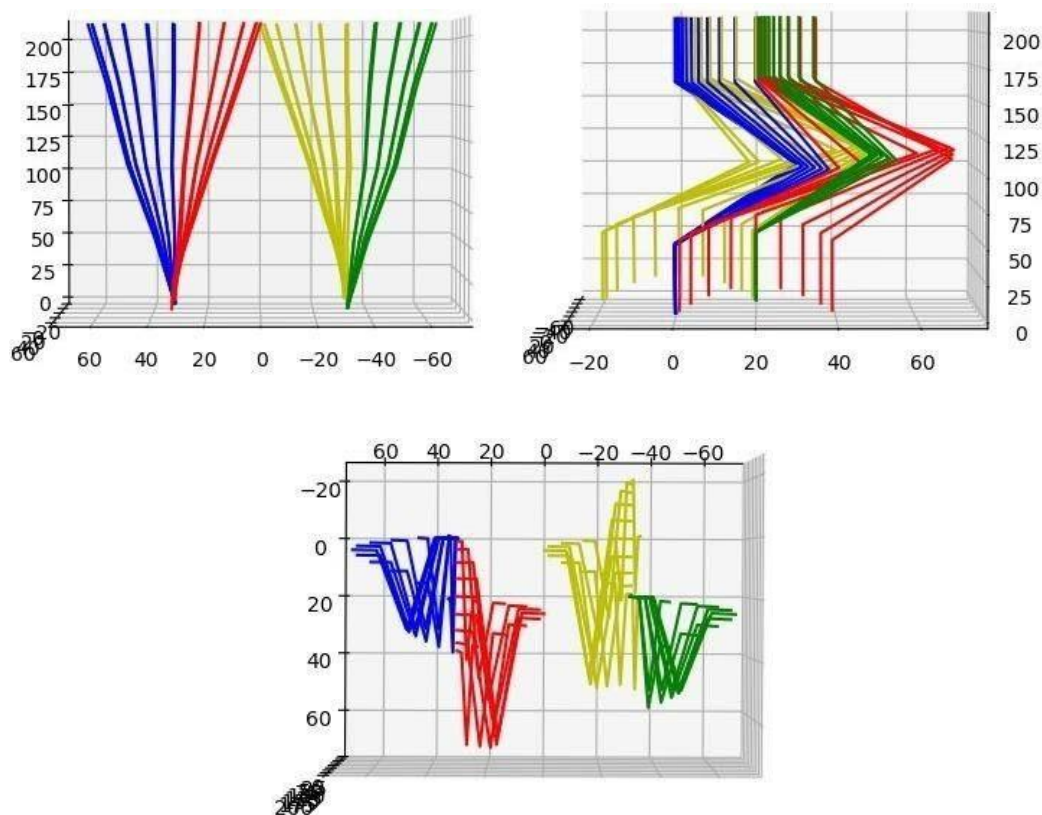


Figure 3.5 (b),(c),(d): 3D LIP front, side and top view

3.6 Stereo Vision

One of the most popular computer vision approaches is the stereo vision framework. The concept is to make use of parallax error to our benefit. The same view of an image is recorded from two different areas, Then, by finding the intensity of parallax error, the depth is estimated. Though this approach is a very conventional it has proved to be of great help in numerous applications. This field has given an idea to a millions of mathematicians and analysts to plan new and interesting computations for the exact produce of the stereo frameworks. This framework plays a huge role in robotics. The framework computes the depth of the object and also gives a better understanding of the image being viewed. This chapter gives a complete overview of the stereo system framework. In this chapter, we will



observe that the depth can be estimated in a better way to yield productive outputs in contrast to the approaches that are being used today, especially if they are made to work with other recognition methods.

Stereo vision is an artificial visual system that was developed by understanding the working of the human eyes. We know the fact that there exists some parallax when we view the same object from two different angles. This parallax helps us to compute the distance between the object and camera. [2]. Parallax error is inversely proportional to the distance. This simplifies it to a small equation. However, computing the disparity between the pixels in the image frames or the parallax error is a serious and important task to manage. Depth can be estimated only if there are any fields that are overlapping between the two. This is described in Figure 3.6. In contrast to a monocular view, a multi-view system is a much more trustworthy and powerful structure for estimating the depth of the objects in the image.

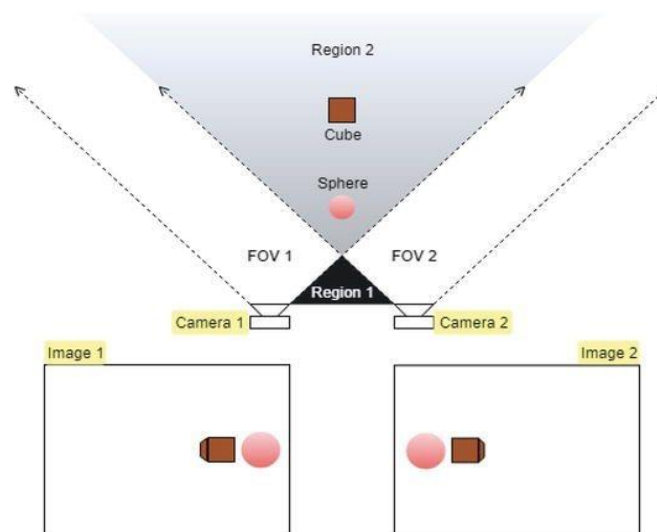


Figure 3.6 (a) : Stereo Vision Instrument

The instrument was initially introduced to us in 1838 by Charles Whitestone to see all of the relief observations. Charles called it as the stereoscope. Some creative and intuitive researchers afterward utilized this concept to create their adaptations of stereoscopes. It indeed led to the foundation of the London Stereoscopic Company in 1854. During the earlier times, depth was estimated from various angles in order to avoid the cosmic objects that were present far away from us. In addition to this fact, the depth is directly related to the baseline. Here, baseline is nothing but the distance between the two cameras. So to get a denser depth we should also keep the two cameras as far away as possible thereby increasing the length of the



baseline. Hence, for finding the depth of an object in the galaxy, the baseline would be the diameter of the orbit of the Earth. And the data was traced with the Earth being on either side of the Sun. This approach is called as the trigonometric or the stellar parallax method.

Considering other applications, mechanical, robotic and the control systems applications request a bounty of stereo vision frameworks for the depth estimations of the nearby objects. Be it humanoids, robots for pressing the clothes or picking up the fallen utensils, or indeed autonomous vehicles, stereo vision frameworks illuminate numerous complexities and hence opportunities. Also we can reduce the cost by making our procedure independent of radars and other measuring instruments. But this can be done only if our applications are short-range and unidirectional.

The Overview of the Stereo Architecture

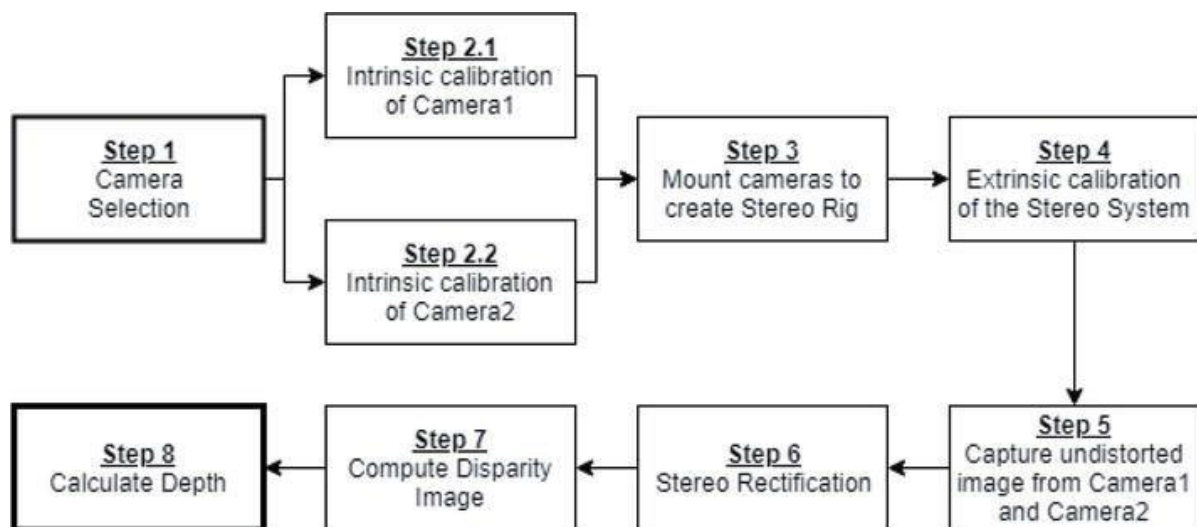


Figure3.6 (b) : Disparity/stereo matching



The architecture shown above provides us the complete explanation of the working of the stereo system. In the stereo system, the two cameras that are having like qualities or attributes are adjusted individually for their intrinsic calibration parameters

Then these cameras are mounted on a hard stereo rig. Then the two cameras are adjusted together like a single system rather than individually. This provides the extrinsic calibration parameters. Then images from the cameras are made accurate to remove the camera distortion effects.

The extrinsic calibration parameters allow us to know the rotation and translation of one camera in contrast to the other. This data helps us align the two images from the stereo system along the epipolar line. The two images can provide us the proper compute of the disparity(dissimilarity between the two images). Matching the pixels of the two images is a very tedious and complicated task. Hence, it is really complex to attain a real-time performance on the images. Once we are given the dissimilarity of each pixel in the two images, we will be able to estimate the depth for each pixel.

This crucial perspective of the entire procedure of estimating the depth by using a stereo, i.e., calculating the difference from the stereo picture match will be covered over here. If we consider the unprocessed picture combine from the stereo, then the whole image is the space where we can find the corresponding pixel. Even if we could simplify the look space a little, it will still not be comparable to searching for a single row of the image. In general, a powerful and strong framework ignores all the image's disfigurements, artifact, and obstacle cases. Hence it can also provide to us the estimated disparity for the pixels in the image pair. Researchers in the field of Analysts and Robotics brought many new and innovative opinions and strategies such as high-end camera units, sensors, the conventional calibration strategies and enhanced measures to compute disparity. These strategies help to evaluate the irregularities in the pixels of the image for a better experience of the stereo vision.

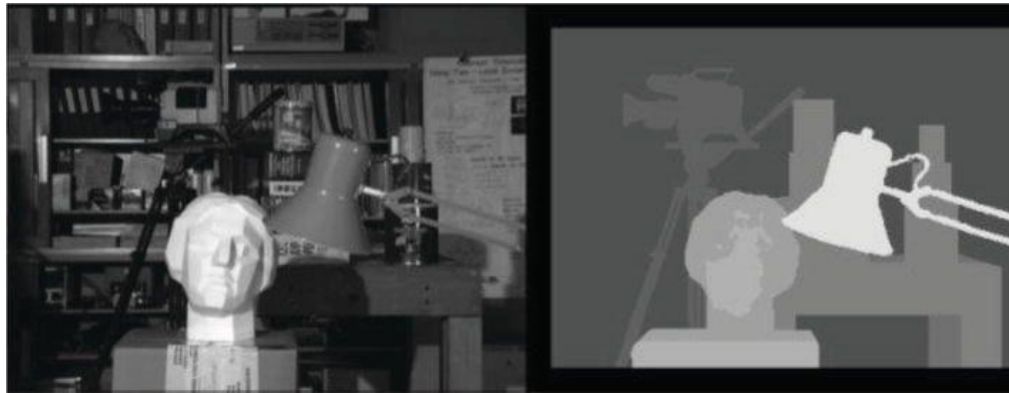


Figure 3.6 (c) : Middlebury stereo dataset. Scene (left), ground truth disparity (right)

The cost function is one of the most essential benchmark of the stereo matching algorithms. It is used to evaluate the similarity between the two images. Some of the significant cost functions are:

- (a) The sum of squared difference (SSD)
- (b) The sum of absolute difference (SAD)
- (c) Normalized cross-correlation (NCC)
- (d) Normalized pixel

Though these fetched capacities are ordinary options for measuring similarity, they are significantly influenced by a few components. These components include illumination contrasts and viewing angles. In order to reduce the effect these variables have on the produce, the pixel patches that are used for closeness check can be normalized some time recently by using SSD or SAD likelihood values. Some methods like censor and rank transform can help us to keep the computation self-sufficient i.e. without a need to use such variables and factors. These changes give out with the sensitivity toward outright concentrated and exceptions.

Inspite of taking care of all these issues, obtaining a dense disparity is a very cumbersome work. It takes a lot of time and effort to compute the disparity. So the next step would be to achieve dense disparity map by using lesser amount of calculations.[2].



3.6.1 Local Stereo Matching Methods

Local stereo matching methods look at the image like a small group of pixels around the chosen pixel that has been is considered, i.e. like a small patch of image. In general, this local approach requires a proper understanding of the image. However, it is extremely efficient and also has a less computation cost compared to global strategies making it more worthy. But the problem is, if there is no proper understanding of the entire image, this can result in incorrect difference maps since the neighborhood ambiguities of the locale like uniform-textured surfaces or blocked pixels cannot be avoided. This issue can be resolved till a certain level. This is possible by using few approaches mentioned below:

1. feature based strategies,
2. area based strategies
3. strategies based on angle optimization.

The post-processing steps have received significant consideration from numerous specialists since it helps to keep the method cheap. [2].

3.6.2 Global Stereo Matching Methods

On the other hand, the global strategies have always shown to be better than the local strategies considering the quantity of the yield. However, it requires extensive calculation. These calculations are harmless to the irregularities that are present local. But sometimes they can manage difficult places that would be hard to manage by using local strategies. Dynamic programming and closest neighbor strategies fall under this category. Global strategies are used once in a while due to their very high computational requests. Analysts for the most part slant toward the nearby stereo coordinating strategies because of its endless extend of conceivable applications through a real-time stereo output[2].

One of the most popular and easiest algorithms used for estimating disparity is block matching. In block matching, a block of pixels are compared with each other. These blocks of pixels are chosen since they are present around the pixel under study. This comparison between the two patches in order to find the disparity is done by making use of one or a number of cost functions that are not restricted to the ones mentioned above. SSD and SAD perform pretty well and hence are the first choices in many algorithms.



In summary, throughout this entire chapter, we pondered around the fundamentals with a little knowledge of the stereo vision framework and we also got to know how insufficiently disparity maps can be sound. We went through the main requirements and applications and also understood the working of the stereo system [2]. As it was already discussed previously, there are a couple of complicated challenges to be addressed while using the stereos for any application. The preparations and planning to these challenges require a lot of labor and it is also required to have a deep knowledge to arrive at the best plan that works for the chosen application. Many specialists have driven ready-made stereo vision frameworks with an appropriate sum of exactness and hence easing the work for analysts to set up a great stereo. These frameworks are capable of being used even for risky applications as well. These applications or use cases can be of personal use or extensive applications as well. Cases of a few of these items are ZED Stereo, Microsoft Kinect, Bumblebee, and numerous more. Today, various arrangements exist for increasing the speed of computing the depth of the images. Such faster calculations are possible only in two scenarios. It is possible when the stereo system makes use of custom-built hardware. Else it is also possible if different types of cameras are used such as the infrared cameras. The strategies that have been discussed till now can help us to use disparity maps with more motivation and confidence regardless of their density.



4. IMPLEMENTATION USING ROS

In this chapter, we will be telling what has been implemented till now in our simulation and how it can be implemented on hardware. We have used ROS Gazebo for simulating our robots.

4.1 ROS Related Concepts

Robot Operating System (ROS or ros) consists of a group of software frameworks for developing robot software. ROS is not an OS. However, it provides assistance for a mixed computer cluster like functionality implementation, low-level device control, package management, hardware abstraction and message-passing between processes. Running processes in ROS can be depicted in graph architecture. Processing happens in ROS nodes. ROS nodes can get and post information to and from the sensor. It also multiplexes data, messages, state, control, planning etc. Though low latency and reactivity are very vital for robot control, ROS is not RTOS (Real Time Operating System) on its own. But still combining ROS with real-time code is still practical. Given below are some of the important ROS concepts.

(a) ROS Nodes

ROS Nodes are nothing but processes that are used to execute computations. ROS software has been created in such a way that it can be ROS is designed to be transformable at a fine-grained scale. In general, there are many nodes in a robot control system. Each node will be used to control a particular task. For example, one node might control the gait of the humanoid, one might work to achieve the local maps, one might execute the task of localization and one might execute the task of path planning. Libraries like roscpp and rospy are used to initiate and handle ROS nodes.

-



(b) ROS Master

The lookup to the computation graph and registration is given by the ROS Master. This helps the ROS nodes to find each other, invoke services and exchange messages.

(c) Parameter Server

It is a part of the master. It enables data to be stored by a key in a central location.

(d) Messages

Messages help the ROS nodes to communicate with each other. A Message can be described as a data structure which comprises of typed fields. ROS Message supports standard types like integer, floating point, Boolean and arrays.

(e) Topics

The ROS messages are transported to the ROS nodes through the subscribe/publish semantics. A message is sent by a node by publishing the message to a particular topic. The topic helps to identify the substance of the message. A node that needs to know about a certain kind of data must subscribe to the appropriate topic. Then it gets the required information.

It is possible for many nodes to publish and subscribe for a single topic concurrently. Also a single node can publish and/or subscribe to multiple topics. Generally, publishers and subscribers are not aware of each others' existence. The plan is to keep the production and the consumption of data separate.

A bus helps in sending and receiving messages. A bus is very similar to a ROS topic. A node that wants to communicate must connect to the ROS bus. Then it will be able to send or receive messages. Every bus will be given a particular name.



(f) **Services**

The publish / subscribe model makes the communication among the ROS nodes very flexible. But for a distributed system to communicate efficiently, a one way system connecting many parts would not be sufficient and appropriate. A reply or a request can be sent or received by using the ROS services. In ROS, there are two kinds of structures for messages. One message structure for reply and one for request. The node that requires any kind of service will send a request message and waits for the reply. A node that is ready to offer any kind of service provides the service under a name. The programmer or the user accesses this interaction in the form of a remote procedure call. This job is again done by the ROS libraries.

(g) **Bags**

There is also a requirement for storing and playing data, messages etc. ROS bag only provides a format for this purpose. This format is needed since it can be very tedious to collect all the data and store them. However, this data would be needed to test and run algorithms.

4.2 About HRP-4

We carried out our simulation by importing HRP-4 robot in the ROS Gazebo environment. The most recent model in the HRP series is the HRP-4 humanoid. It is also one of the most advanced humanoid all thanks to the deep rooted partnership between Kawada industries. Kawada industries are led by Tadahiro Kawada, and Japan's National Institute of Advanced Industrial Science and Technology (AIST), headed by Tamotsu Nomakuch. HRP-2 was developed in 2002. HRP-3 then was developed in 2006. After this, HRP-4 was developed. The aim of this project was to make the humanoid more light weight and hence secure for human interactions. In this project, engineers from Kawada focused on the humanoid robot hardware while the AIST researchers



Figure4.2: HRP-4 Humanoid

developed the motion-control software. The goal was to make the new humanoid convenient and lighter making it safer for human interaction. But HRP-4 was created with aim to make it capable than HRP-2 and HRP-3 humanoids in terms of manipulating objects and navigating human environments. It was made sure that this objective was fulfilled. The final design was unveiled in September 2010.

HRP-4 is one of the world's most advanced humanoids, the result of a decade of R&D. It's designed to collaborate with humans and can perform remarkably natural, human-like movements.

**Table 4.1 Specifications for HRP-4**

Created by	Kawada Industries and AIST
Country	Japan
Year	2010
Type	Humanoids, Research
Features	Compact, lightweight design. Able to lift 0.5 kg (1.1 lb) with each arm. Equipped with low-power motors for improved safety.
Height	151 cm 59.4 in
Weight	39 kg 86 lb
Sensors	Cameras (in the head and arms), microphones.
Actuators	80-W motors and lower power motors
Computing	PC/104 Pentium M computer with Wi-Fi and speakers.
Software	Linux OS with RT-PreemptPatch and OpenRTM-aist middleware.
Degrees Of Freedom (DOF)	34 (Leg: 6 DoF x 2; Neck: 2 DoF; Chest: 2 DoF; Arm: 7 DoF x 2; Hand: 2 DoF x 2)
Materials	Plastic covers, aluminum alloy frame.
Official Name	HRP - 4

4.3 ROS Gazebo

Gazebo allows us to view the actions of the humanoid in a 3D scenario on your computer. It provides the robots, obstacles and many other objects that will be required to carry out the simulation. Gazebo uses a physical engine for illumination, gravity, inertia, etc. So the robot or the humanoid can be assessed and tested during risky tasks without any harm to the robot in the real world hence minimizing the damage. Apart from this, it takes much lesser time to run a instead of starting the



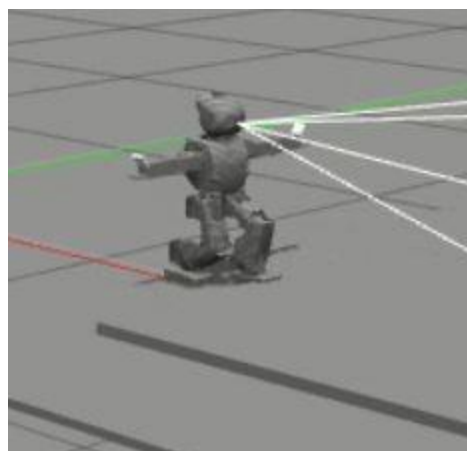
whole scenario on your robot in the real world.

Actually Gazebo was designed to address the problem of testing the algorithms on the robots. During a lot of times, it is crucial to evaluate your robot application, for various activities such as battery life, navigation, grasping, error handling and localization. Since simulating the robot was a priority, Gazebo was produced and then later improved.

4.4 Our Simulation

1. In ROS Gazebo, we have imported the rospy library that helps us to execute all the ROS related commands in Python.
2. There is a ROS node that is connected to all the leg joints. We know that ROS nodes connect the external environment to the humanoid and vice versa. Here the external environment is python. In the python code, the servo angles are being calculated. These servo angles are given as input to the humanoid with the help of ROS nodes.
3. There is a stereo vision camera sensor that has been attached to it. Hence, now the robot can perceive its surroundings. There is a cuboid that is kept in front of the humanoid. When the robot nears it, the robot calculates its slope and raises its leg to place it on the cuboid.
4. For calculating the slope, a simple equation $\tan^{-1}((y_2 - y_1)/(x_2 - x_1))$ has been used. Here x_1, y_1 are the coordinates of the lower rectangle and x_2, y_2 are the coordinates of the higher rectangle. The slope is calculated using this equation and the servo motors are adjusted accordingly.
5. So on perceiving the obstacles and while walking on unsmooth surfaces the robot plans its gaits accordingly and walks.

Figure 4.4: Screenshot from the simulation





4.5 Hardware Implementation

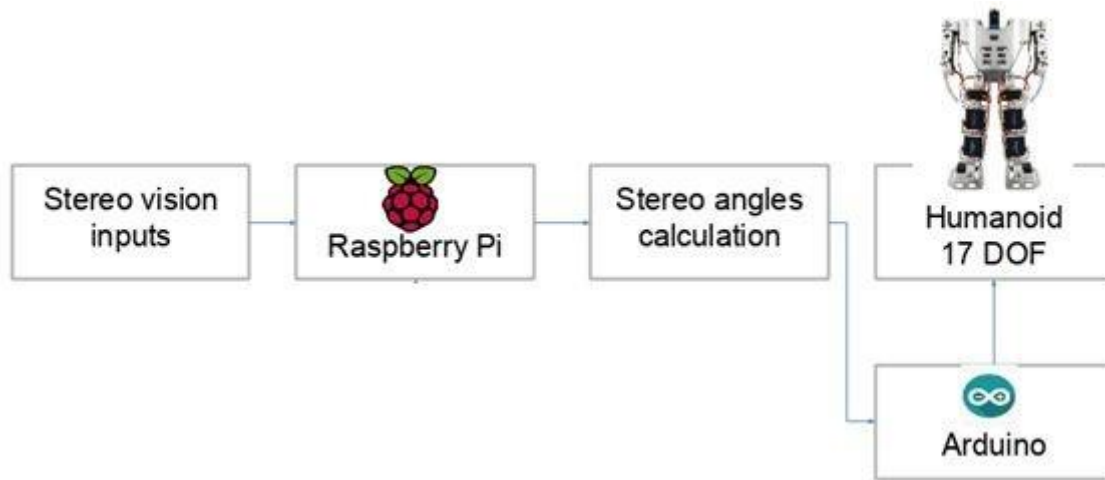


Figure 4.5(a): Block Diagram

1. The humanoid on which our logic will be implemented will have a stereo vision camera sensor attached to it. The camera provides the image data to the humanoid that is used for object identification, tracking and manipulation tasks. Hence this allows the humanoid to sense the obstacles around it and gets familiar with the environment. It computes its distance from the obstacles with respect to its own frame.
2. When the x, y coordinates of the obstacles are found then the slope are also calculated. This is given to the raspberry pi where the servo angles are calculated.
3. After the angles are calculated, they are given as inputs to the Arduino.
4. The Arduino in turn, controls the servos and the humanoid adjusts its gaits and starts moving with stability.

This is how the logic will be implemented on hardware.





5. RESULTS

The generated gaits were tested on the imported humanoid model in ROS Gazebo. The robot performed to match the expectations. The results obtained are described below. The main operations that were performed on the robot were, walking forward, turn left and right and also it was capable of walking on slopes and stairs.

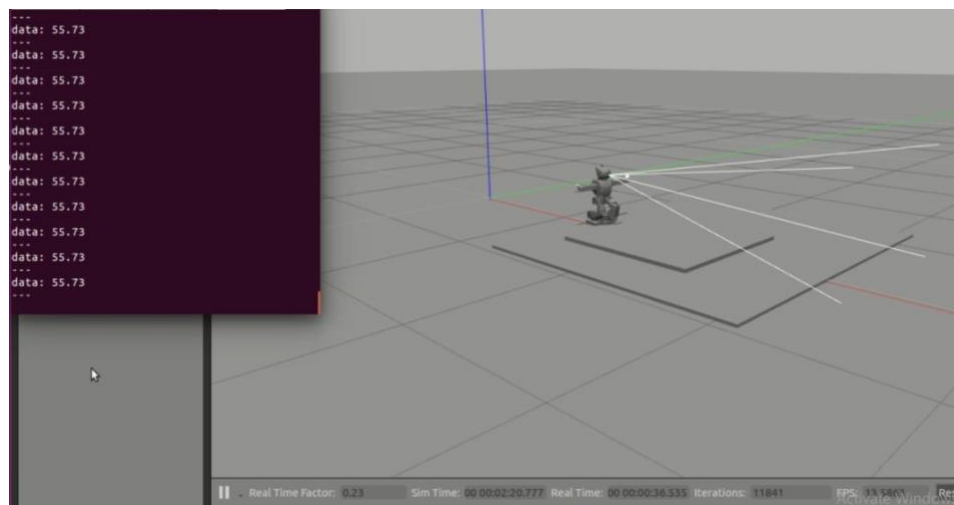


Figure 5.1: Simulation of HRP-4

Whenever the robot moves from a lower step to a higher step, the slope is calculated and is displayed on the terminal as shown in the image above, thus allowing the robot to walk on unsmooth surfaces.

The logic has been successfully implemented on ROS Gazebo. This can be further implemented on the hardware as well.



6. CONCLUSION

A noteworthy advance has been made towards steady mechanical and robotic bipedal walking, thus resulting in a big and interesting query about creating an independent route navigation methodologies custom-made particularly to humanoid robots. The capacity of the bipedal, humanoid robots to not only move around an obstacle but also to be able to walk over a few deterrents makes them apt for terrains and surfaces planned for the humans that frequently consist of a wide mix of objects and deterrents like furniture, entryways, stairs, and uneven ground. If the humanoid should be able to navigate and explore its environment with liberty, without any thought of falling down anywhere, it should be able to visualize the environment that it is present in from the information it receives from its sensors. Using this data, the route can be developed. So, in order to create the gait that will make the humanoid reach a destination, the robot's environment should be available to it in the form of global maps. This results in more productivity. In any case, on-body perception is solely restricted in extent and course. Some approaches have focused on retrieving the information about the local environment that the humanoid is currently in. This only enables a few strategies for navigation such as reactive obstacle avoidance to be implemented. Some other approaches have used off-body visual sensing in order to compute the global maps. However the latter approach sacrifices some level of liberty of the robot. Through this approach, barriers like parallax and occlusions can be found.

Further study will focus on addition of toes to robot's foot and make it counteract the disturbances and use of an on-board computer for real time generation of sequences to get more realistic walk on this humanoid, TONY. The scope of possibilities for future work on the humanoid is endless and there are multiple ways of improvement in the robot whether be it improving the automatic gait-generating algorithm to upgrading the hardware.



APPENDIX

1. Python code for slope calculation

```
#!/usr/bin/env python
```

```
import rospy
from std_msgs.msg import Float64
from gazebo_msgs.msg import ModelStates
import math
```

```
class Angle(object):
    def __init__(self):
        self.angle = 0
        rospy.Subscriber("/gazebo/model_states", ModelStates, self.angle_detect)
        self.pub = rospy.Publisher('angle_detected', Float64, queue_size=10)

    def angle_detect(self, data):
        if(data.pose[3].position.x > 1.14 and data.pose[3].position.x < 1.3):
            self.angle = math.atan(data.pose[3].position.y/data.pose[3].position.x) #angle value in
                                                                                      radians

        self.angle = math.degrees(self.angle)    # self.angle value is in degrees
        self.pub.publish(self.angle)
        elif(data.pose[3].position.x > 1.5 and data.pose[3].position.x < 2):
            self.angle = math.atan(data.pose[3].position.y/data.pose[3].position.x) # self.angle
value is in radians
        self.angle = math.degrees(self.angle)    # self.angle value is in degrees
        self.pub.publish(self.angle)
        else:
            self.angle = 0

    def spin(self):
        print(self.angle)
        while not rospy.is_shutdown:
            self.pub.publish(self.angle)
            rate.sleep()

def main():
    rospy.init_node("angle_detector")
    node = Angle()
    rospy.spin()

if __name__ == '__main__':
    main()
```

