UNIVERSITY OF ENGINEERING AND TECHNOLOGY

LAHORE

FACULTY OF MECHATRONICS ENGINEERING

ADVANCE ROBOT PROGRAMMING REPORT

HUMANOID ROBOT SIMULATION WITH MATLAB

Student:

Muhammad Mudassir Shakeel

2017-MC-294

Contents

CHAPTER 1: INTRODUCTION

This project is about programming a humanoid robot with 6 degrees of freedom (DOF) in the arm, 6 DOF in the leg, and 2 DOF in the neck, utilizing the MATLAB platform. The study encompasses various aspects including kinematics (forward and inverse), gait generation, velocity control, path planning, and trajectory generation. Throughout the project, different algorithms and techniques are implemented and evaluated to achieve precise and efficient movements of the robot. The outcomes of this research contribute to the progress of humanoid robot programming and support the development of intelligent and capable humanoid robots for diverse applications.

1.1 Aim of Research

The primary objective of this project is to examine the movement of human joints and understand their correlation with the body during motion. From this understanding, the project aims to derive displacement matrices that can be used to simulate human movement in MATLAB.

1.2 Research Methods

This project is conducted through the following methods:

- Researching academic equations for forward kinematics and inverse kinematics.
- Performing simulations using MATLAB software.

CHAPTER 2: THEORETICAL BASIS AND DESIGN CONCEPTS

2.1. Humanoid robot

Humanoid robots are advanced machines designed to resemble and mimic human characteristics and movements. They are built upon a strong theoretical foundation that combines principles from various fields such as robotics, artificial intelligence (AI), biomechanics, and cognitive science. Understanding the theoretical basis of humanoid robots is crucial for comprehending their capabilities and potential applications.

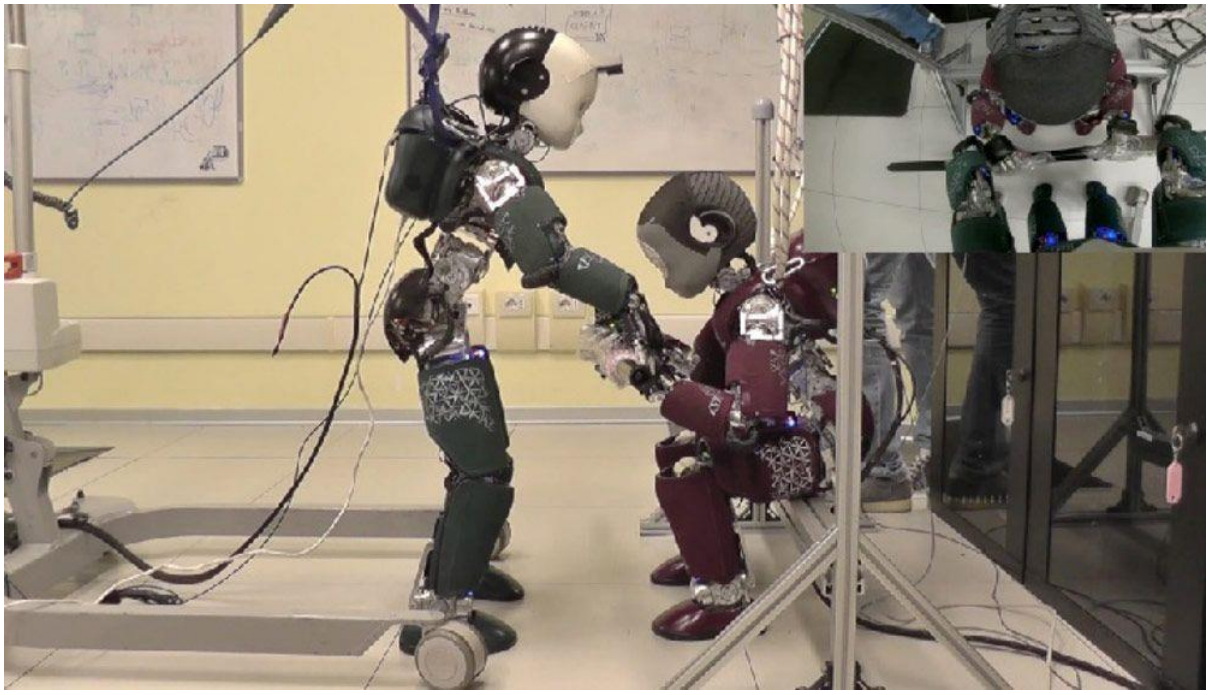


Figure 1: ICub Humanoid Robot

2.2. Theoretical Basis of Humanoid Robots for Simulink and MATLAB

Humanoid robots are complex machines designed to resemble and replicate human movements and behavior. Simulink, a simulation and modeling environment in MATLAB, provides a powerful platform for studying and simulating the movement and traits of humanoid robots. Understanding the theoretical basis of humanoid robots within the Simulink and MATLAB framework is crucial for comprehending their simulation and control capabilities.

a/ Robot Modeling and Simulation

Modeling and simulation play a fundamental role in studying and understanding humanoid robot behavior. In Simulink and MATLAB, you can create kinematic and motion models to represent the structure and movement of humanoid robots. These models enable you to prototype algorithms, simulate robot motion, and test various scenarios. By accurately modeling the kinematics and dynamics of the robot, you can simulate and analyze the robot's behavior in different environments.

b/ Importing Humanoid Robot Models

Simulink and MATLAB provide capabilities to import humanoid robot models from URDF files. URDF (Unified Robot Description Format) is a standard file format used to describe the structure and properties of robots. By importing URDF files into Simulink, you can visualize the robot's parts and create a simulation environment for studying its movement. These imported models can be used to analyze the behavior of the robot's joints, simulate various trajectories, and implement control algorithms.

c/ Kinematics and Motion Control

Kinematics is a crucial aspect of humanoid robot simulation. It involves the study of robot motion without considering the forces or torques involved. Simulink and MATLAB offer tools to model the kinematics of humanoid robots, including forward and inverse kinematics. Forward kinematics determines the position and orientation of robot end-effectors based on joint angles, while inverse kinematics calculates the required joint angles to achieve a desired end-effector position. By accurately modeling the kinematics, you can simulate and control the robot's movement in different scenarios.

d/ Trajectory Planning and Control

Trajectory planning and control are essential for humanoid robot movement. In Simulink and MATLAB, you can develop algorithms and controllers to generate smooth and efficient trajectories for the robot's joints. These trajectories can be planned to achieve tasks such as walking, reaching, or grasping objects. By implementing trajectory planning and control algorithms, you can simulate and optimize the robot's movement to perform desired actions.

e/ Sensor Integration and Perception

Humanoid robots often incorporate various sensors for perception and interaction with the environment. Simulink and MATLAB provide functionalities to integrate sensor models and simulate their interactions with the robot. This includes computer vision models for visual perception, inertial sensors for balance and orientation estimation, and force sensors for interaction with the surroundings. By integrating sensors into the simulation, you can study the impact of perception on the robot's movement and behavior.

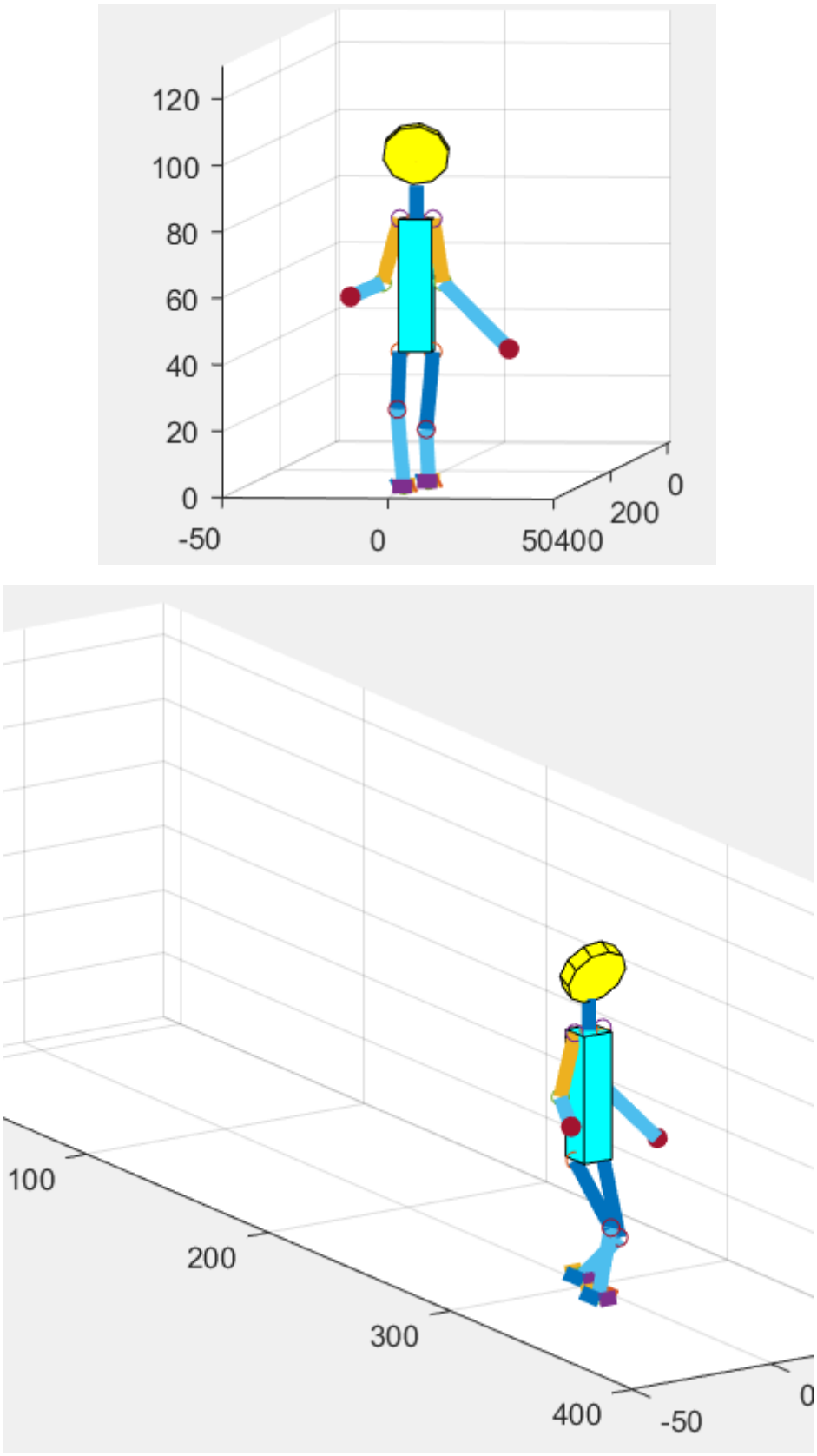


Figure 2 Model Design

The humanoid robot also has a 2-DoF neck that allows it to tilt and spin the head. The robot needs the neck joints' critical degrees of freedom in order to modify its line of sight, focus

on certain things or people, and participate in interactive behaviors with better situational awareness. The robot can do activities that involve head orientation and create efficient visual communication by simulating the human neck's movement capabilities. This increases the robot's overall adaptability and human-like presence.

The 6-DoF arms, 6-DoF legs, and 2-DoF neck work together to provide the humanoid robot an impressive amount of anthropomorphic capability. This design strategy broadens the robot's potential uses in the industrial, clinical, educational, and entertainment sectors by enabling it to carry out a variety of activities, interact with its surroundings, and display natural and coordinated motions similar to those of a person.

CHAPTER 3: SIMULATION MODEL, CONTROL ALGORITHMS

3.1 Forward kinematics and Inverse kinematics for motion

In robotics literature, forward kinematics is commonly known as the task in which the position and orientation of the end-effector is to be determined by giving the configurations for the active joints of the robot . This project, I use Denavit–Hartenberg (DH) method to find position of the end point of arms, legs, and neck. This section is also concerned with finding the solution to the inverse kinematics problem, which consists of determining the joint variables in terms of the end effector position and orientation. It is commonly known in the literature that for open kinematic chains, the determination of closed-form equations for the inverse kinematics represents a greater challenge than the forward kinematics.

```
function R = eulerXYZ(a1, a2, a3)
% Convert XYZ Euler angles to rotation matrix

R1 = [
1, 0, 0;
0, cos(a1), -sin(a1);
0, sin(a1), cos(a1)];

R2 = [
cos(a2), 0, sin(a2);
0, 1, 0;
-sin(a2), 0, cos(a2)];

R3 = [
cos(a3), -sin(a3), 0;
sin(a3), cos(a3), 0;
0, 0, 1];

R = R1*R2*R3;

end
```

Figure 3 Function for Euler transform

```
n = input("Number of joints: ");
syms l t an d l1 l2 l3 l4 t1 t2 t3 t4;
T = eye(4);
for i = 1:n
    t(i) = input("theta:");
    l(i) = input("l:");
    an(i) = input("alpha:");
    d(i) = input("d:");
    A = [
        cos(t(i)) -sin(t(i))*cos(an(i)) sin(t(i))*sin(an(i)) l(i)*cos(t(i));
        sin(t(i)) cos(t(i))*cos(an(i)) cos(t(i))*sin(an(i)) l(i)*sin(t(i));
        0 sin(an(i)) cos(an(i)) d(i);
        0 0 0 1
        ]
    T = T*A;
end
R= simplify(T)
```

Figure 4 the formula for DH transform

3.2 Leg Control:

It is worth mentioning that in the early stages of examining the simulation platform, the robot was tested whether it can run the normal gait with and without toe joint. The advantages of the presence of the toe joint can be noticed where faster and larger step size have been performed by the robot in the toe-joint.

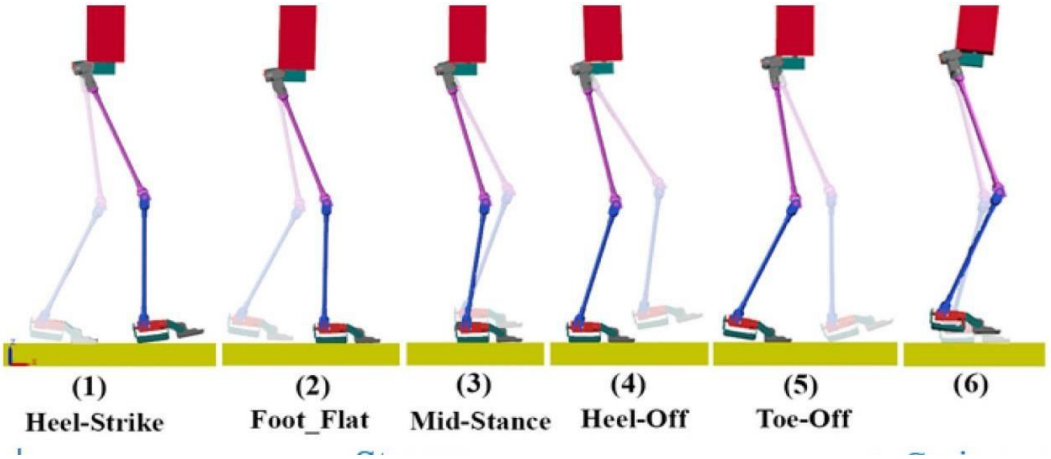
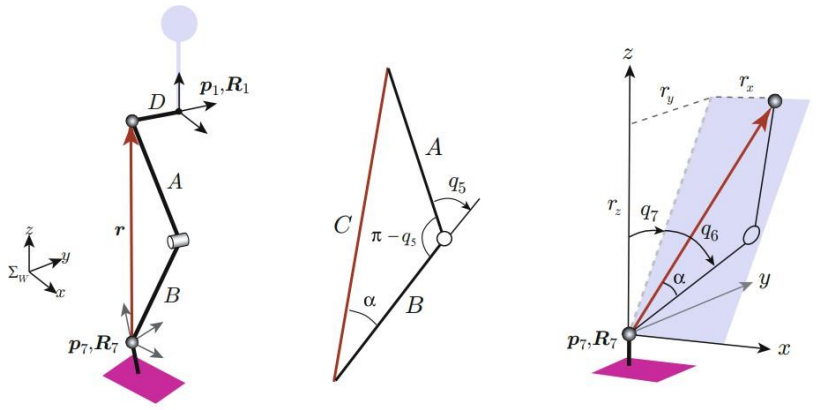


Figure 5 Leg movement planing

Spliting the movement period into 2 part,

Each part include 6 step in the upper figure, the difference in each parts is distinguished by which leg is the standing leg, which is the moving leg.



(a) Definition (b) Get the knee angle (c) Get the ankle angles

In the upper figure, we know (p_1, R_1) and (p_7, R_7). So, the position of the hip would be,

$$\boldsymbol{p}_2 = \boldsymbol{p}_1 + \boldsymbol{R}_1 \begin{bmatrix} 0 \\ D \\ 0 \end{bmatrix}.$$

Next, we calculate the position of the crotch viewed from the ankle coordinate space

$$\boldsymbol{r} = \boldsymbol{R}_7^T (\boldsymbol{p}_2 - \boldsymbol{p}_7) \equiv [r_x \; r_y \; r_z]^T.$$

From this we can calculate the distance between the ankle and the hip, which we will define as

$$C = \sqrt{r_x^2 + r_y^2 + r_z^2}.$$

After transforming and calculating, we can get some valuable equations. The angle of the knees will be,

$$q_5 = -\cos^{-1}\left(\frac{A^2 + B^2 - C^2}{2AB}\right) + \pi.$$

Next we will focus on the ankle local coordinates. As shown in Fig. 2.25(c), from vector r you can calculate the ankle roll and pitch angles. So,

$$q_7 = \text{atan2}(r_y, r_z)$$

$$q_6 = -\text{atan2}\left(r_x, \text{sign}(r_z)\sqrt{r_y^2 + r_z^2}\right) - \alpha.$$

$$q_2 = \text{atan2}(-R_{12}, R_{22})$$
$$q_3 = \text{atan2}(R_{32}, -R_{12}s_2 + R_{22}c_2)$$
$$q_4 = \text{atan2}(-R_{31}, R_{33}).$$

Another concerning parameter of leg movement is keeping center of body in the surface of the feet.
Which calculate using the function below

```
w=w+sqrt((x2-x1)^2+(y2-y1)^2+(z2-z1)^2);
p=p+((x2-x1)^2+(y2-y1)^2+(z2-z1)^2)/2;
yCenter = p/w;
```

So the movement must keep the y value in the range of the size of the standing feet.

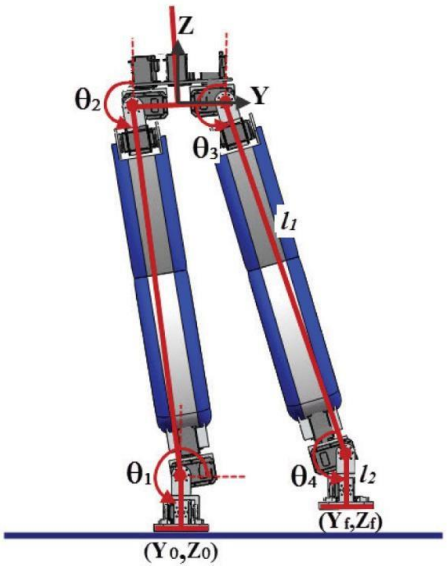


Figure 6 Frontal plane view of the humanoid robot

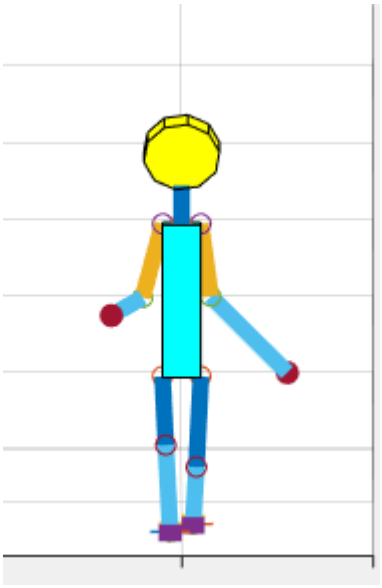


Figure 7 Demo for leg balancing mass center position

3.3 Head Control

The ideal is keep the head looking forward while moving. As the body is oscillate left and right while moving. The angle (a3) of this oscialate can be calculated depend on the trait of movement cotrolling.

Using Euler transform to rotate the angle as the formula:

```
R3 = [
cos(a3), -sin(a3), 0;
sin(a3), cos(a3), 0;
0, 0, 1];
```

Also the head can move around it's parents coordinate system (neck) using Euler transform for X,Y,Z rotation.

Position of neck:

- px = l*s1
- py = 0
- pz = l*c1

Inverse kinematics of neck motion

$\text{theta1} = \sin^{-1}\frac{px}{l1}$

$0° \leq theta2 \leq 180°$

3.4 Arm Control

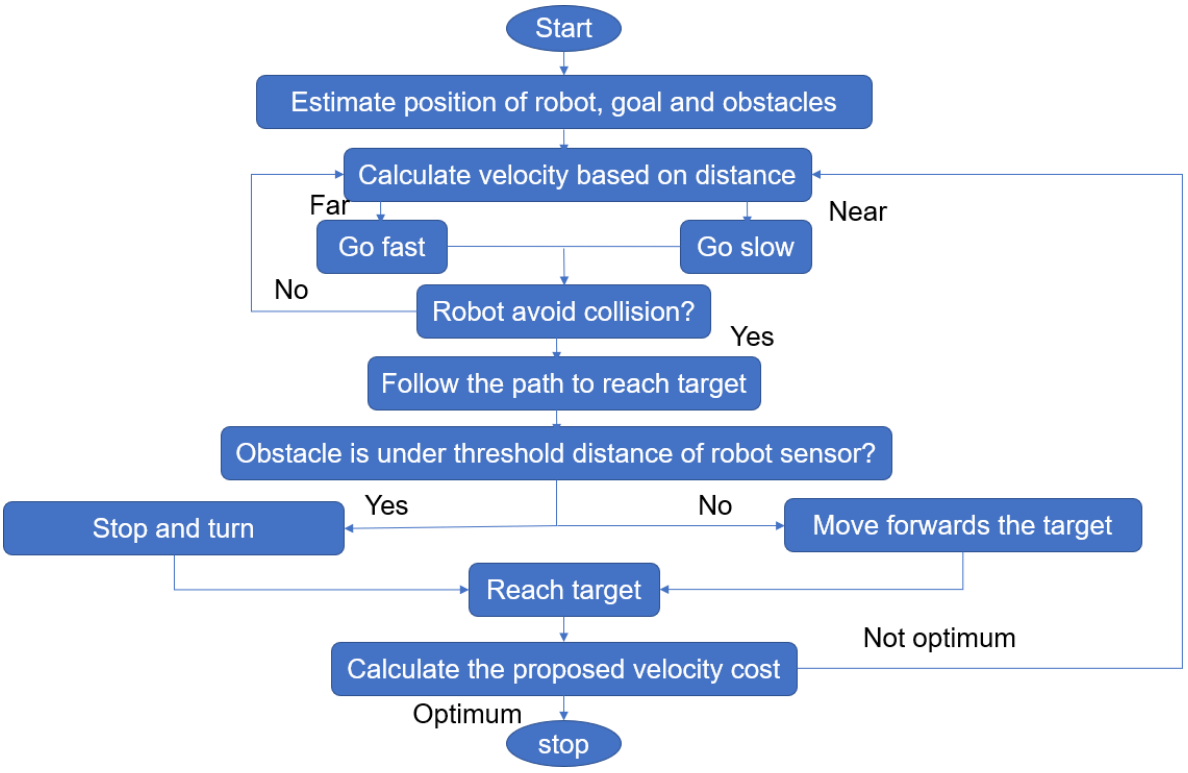
Inverse kinematics for arm motion

Right arm DH parameters				
Coord. Frame i	θ_i	α_i	a_i	d_i
1	$\theta_1 + \pi/2$	$\pi/2$	0	0
2	$\theta_2 - \pi/2$	$\pi/2$	0	0
3	$\theta_3 + \pi/2$	$-\pi/2$	0	$-l_{A2}$
4	θ_4	$\pi/2$	0	0
5	θ_5	$-\pi/2$	0	$-l_{A3}$
6	$\theta_6 + \pi/2$	0	l_{A4}	0

Applying DH transfrom to calculate the posiiton of each wrist of the robot Arm

3.5 Path planning

The dynamic window technique, which suggests an online collision avoidance strategy for mobile robots, is the newest player in the field of motion planning algorithms. The navigational strategy uses the mobile robot's dynamics to create the best avoidance plan while taking into account the limitations placed on the robotic mobility by various characteristics like velocity and acceleration. The algorithm creates a reasonable path for the robot to follow while taking into consideration the many environmental uncertainties, and it responds appropriately in the event of an unforeseen circumstance. The fundamental path of the robot's locomotion is laid out by geometric functions, and it is guided at corners by heuristic functions. To achieve the suggested goal, the heuristic functions follow the principle of "traveling faster in the right direction".



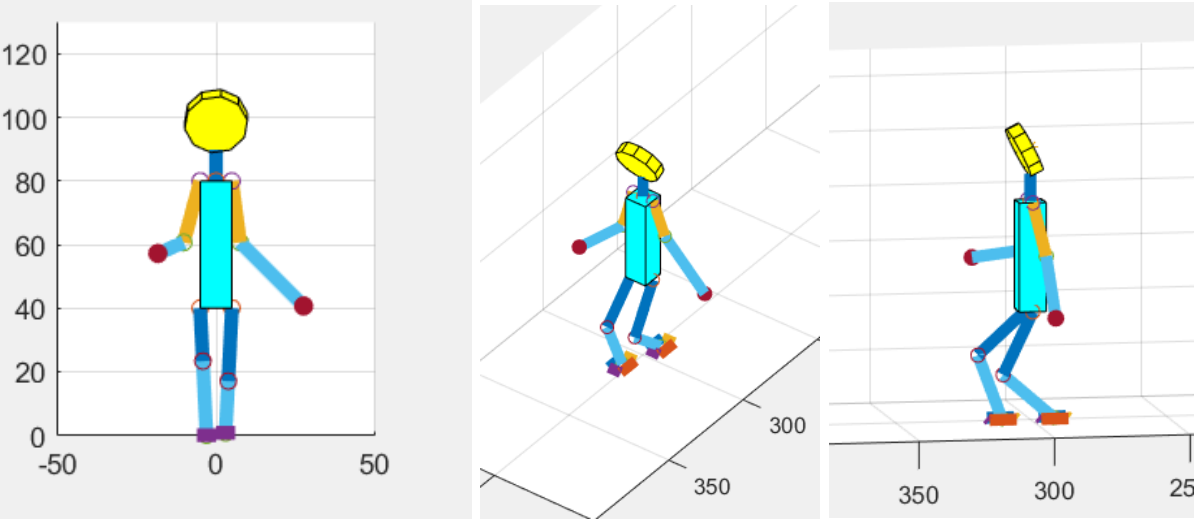
CHAPTER 4: EXPERIMENT

1. Environment setting

Hardware: All results are run in CPU Intel-12700H. Software: MATLAB R2021a

2. Result:

I am able to simulate a simple walking humanoid robot, which simulates the sneak movement.



Matlab code:

```
w=0;
p=0;
j=0;
t=0.5;
list=[];
xchp=-10;
xcht=10;
xtt=-10;
xtp=30;
dodaichan=22;
for i=0:1:300
    hold off

    if mod(i,40)<20
        %LEFT MOVEMENT
        %leg drawing
        xchp=xchp+2;
        zchp=-0.04*mod(i+1,40)^2+20/25*mod(i+1,40);
        X=sqrt((i-xchp)^2+(-2+j)^2+(40-zchp)^2);
        alpha1=acos(X/50)-atan2(i-xchp,X);
```

```
        beta1=acos(40/(X*cos(atan2(i-xchp,X))));
        [w,p]=banchan(w,p,xchp,-3,zchp);
        plot3(xchp,-3,zchp,'o');
        [w,p]=veline(w,p,xchp,-3,zchp,25*sin(alpha1)+i,-
5+j-25*cos(alpha1)*cos(beta1)/40*(-2+j),40-
25*cos(alpha1)*cos(beta1));
        plot3(25*sin(alpha1)+i,-5+j-
25*cos(alpha1)*cos(beta1)/40*(-2+j),40-
25*cos(alpha1)*cos(beta1),'o');
        [w,p]=veline(w,p,25*sin(alpha1)+i,-5+j-
25*cos(alpha1)*cos(beta1)/40*(-2+j),40-
25*cos(alpha1)*cos(beta1),i,-5+j,40);
        %hand drawing
        plot3(i,-5+j,40,'o');
        [w,p]=veline(w,p,i,-
5+j,80,i+20*sin(pi/12)*sin((mod(i,40)-20)*pi/60),-5+j-
20*sin(pi/12)*cos((mod(i,40)-20)*pi/60),80-
20*cos(pi/12));
        plot3(i,-5+j,80,'o');
        plot3(i+20*sin(pi/12)*sin((mod(i,20)-
20)*pi/60),-5+j-20*sin(pi/12)*cos((mod(i,20)-
20)*pi/60),80-20*cos(pi/12),'o');
        [w,p]=veline(w,p,i+20*sin(pi/12)*sin((mod(i,20)-
20)*pi/60),-5+j-20*sin(pi/12)*cos((mod(i,20)-
20)*pi/60),80-
20*cos(pi/12),i+20*sin(pi/12)*sin((mod(i,20)-
20)*pi/60)+20*sin(mod(i,20)/45*pi),-5+j-
20*sin(pi/12)*cos((mod(i,20)-20)*pi/60)-
20*cos(mod(i,20)/45*pi),80-20*cos(pi/12)-
20*cos(mod(i,20)/45*pi));
        scatter3(i+20*sin(pi/12)*sin((mod(i,20)-
20)*pi/60)+20*sin(mod(i,20)/45*pi),-5+j-
20*sin(pi/12)*cos((mod(i,20)-20)*pi/60)-
20*cos(mod(i,20)/45*pi),80-20*cos(pi/12)-
20*cos(mod(i,20)/45*pi),50,'filled');
        %RIGHT MOVEMENT
        Y=sqrt((i-xcht)^2+(-2-j)^2+(40)^2);
        alpha2=acos(Y/50)-atan2(i-xcht,Y);
        beta2=acos(40/(Y*cos(atan2(i-xcht,Y))));
        [w,p]=banchan(w,p,xcht,3,0);
        plot3(xcht,3,0,'o');
        [w,p]=veline(w,p,xcht,3,0,25*sin(alpha2)+i,5+j-
25*cos(alpha2)*cos(beta2)/40*(2+j),40-
25*cos(alpha2)*cos(beta2));
        plot3(25*sin(alpha2)+i,5+j-
25*cos(alpha2)*cos(beta2)/40*(2+j),40-
25*cos(alpha2)*cos(beta2),'o');
```

```
        [w,p]=veline(w,p,25*sin(alpha2)+i,5+j-
25*cos(alpha2)*cos(beta2)/40*(2+j),40-
25*cos(alpha2)*cos(beta2),i,5+j,40);
        plot3(i,5+j,40,'o');
        [w,p]=veline(w,p,i,5+j,80,i+20*sin(pi/12)*sin(-
mod(i,20)*pi/60),5+j+20*sin(pi/12)*cos(-
(mod(i,20))*pi/60),80-20*cos(pi/12));
        plot3(i,5+j,80,'o');
        plot3(i+20*sin(pi/12)*sin(-
mod(i,20)*pi/60),5+j+20*sin(pi/12)*cos(-
(mod(i,20))*pi/60),80-20*cos(pi/12),'o');
        [w,p]=veline(w,p,i+20*sin(pi/12)*sin(-
mod(i,20)*pi/60),5+j+20*sin(pi/12)*cos(-
(mod(i,20))*pi/60),80-
20*cos(pi/12),i+20*sin(pi/12)*sin(-
mod(i,20)*pi/60)+20*sin(20-
mod(i,20)/45*pi),5+j+20*sin(pi/12)*cos(-
mod(i,20)*pi/60)+20*cos(20-mod(i,20)/45*pi),80-
20*cos(pi/12)-20*cos((20-mod(i,20))/45*pi));
        scatter3(i+20*sin(pi/12)*sin(-
mod(i,20)*pi/60)+20*sin(20-
mod(i,20)/45*pi),5+j+20*sin(pi/12)*cos(-
mod(i,20)*pi/60)+20*cos(20-mod(i,20)/45*pi),80-
20*cos(pi/12)-20*cos((20-mod(i,20))/45*pi),50,'filled');
    end
    if mod(i,40)>=20
        %left movement
        xcht=xcht+2;
        zcht=-0.04*(mod(i,40)-19)^2+20/25*(mod(i,40)-
19);
        X=sqrt((i-xcht)^2+(-2+j)^2+(40-zcht)^2);
        alpha1=acos(X/50)-atan2(i-xcht,X);
        beta1=acos(40/(X*cos(atan2(i-xcht,X))));
        [w,p]=banchan(w,p,xcht,3,zcht);
         plot3(xcht,3,zcht,'o');

[w,p]=veline(w,p,xcht,3,zcht,25*sin(alpha1)+i,5+j-
25*cos(alpha1)*cos(beta1)/40*(2+j),40-
25*cos(alpha1)*cos(beta1));
        plot3(25*sin(alpha1)+i,5+j-
25*cos(alpha1)*cos(beta1)/40*(2+j),40-
25*cos(alpha1)*cos(beta1),'o');
        [w,p]=veline(w,p,25*sin(alpha1)+i,5+j-
25*cos(alpha1)*cos(beta1)/40*(2+j),40-
25*cos(alpha1)*cos(beta1),i,5+j,40);
        plot3(i,5+j,40,'o');
```

```
[w,p]=veline(w,p,i,5+j,80,i+20*sin(pi/12)*sin((mod(i,20)
-20)*pi/60),5+j+20*sin(pi/12)*cos((mod(i,20)-
20)*pi/60),80-20*cos(pi/12));
        plot3(i,-5+j,80,'o');
        plot3(i+20*sin(pi/12)*sin((mod(i,20)-
20)*pi/60),5+j+20*sin(pi/12)*cos((mod(i,20)-
20)*pi/60),80-20*cos(pi/12),'o');
        [w,p]=veline(w,p,i+20*sin(pi/12)*sin((mod(i,20)-
20)*pi/60),5+j+20*sin(pi/12)*cos((mod(i,20)-
20)*pi/60),80-
20*cos(pi/12),i+20*sin(pi/12)*sin((mod(i,20)-
20)*pi/60)+20*sin(mod(i,20)/45*pi),5+j+20*sin(pi/12)*cos
((mod(i,20)-20)*pi/60)+20*cos((mod(i,20))/45*pi),80-
20*cos(pi/12)-20*cos((mod(i,20))/45*pi));
        scatter3(i+20*sin(pi/12)*sin((mod(i,20)-
20)*pi/60)+20*sin(mod(i,20)/45*pi),5+j+20*sin(pi/12)*cos
((mod(i,20)-20)*pi/60)+20*cos((mod(i,20)/45*pi)),80-
20*cos(pi/12)-20*cos((mod(i,20))/45*pi),50,'filled');
        %right movement
        Y=sqrt((i-xchp)^2+(-2-j)^2+(40)^2);
        alpha2=acos(Y/50)-atan2(i-xchp,Y);
        beta2=acos(40/(Y*cos(atan2(i-xchp,Y))));
        [w,p]=banchan(w,p,xchp,-3,0);
        plot3(xchp,-3,0,'o');
        [w,p]=veline(w,p,xchp,-3,0,25*sin(alpha2)+i,-
5+j-25*cos(alpha2)*cos(beta2)/40*(-2+j),40-
25*cos(alpha2)*cos(beta2));
        plot3(25*sin(alpha2)+i,-5+j-
25*cos(alpha2)*cos(beta2)/40*(-2+j),40-
25*cos(alpha2)*cos(beta2),'o');
        [w,p]=veline(w,p,25*sin(alpha2)+i,-5+j-
25*cos(alpha2)*cos(beta2)/40*(-2+j),40-
25*cos(alpha2)*cos(beta2),i,-5+j,40);
        plot3(i,-5+j,40,'o');
        [w,p]=veline(w,p,i,-5+j,80,i+20*sin(pi/12)*sin(-
mod(i,20)*pi/60),-5+j-20*sin(pi/12)*cos(-
(mod(i,20))*pi/60),80-20*cos(pi/12));
        plot3(i,5+j,80,'o');
        plot3(i+20*sin(pi/12)*sin(-mod(i,20)*pi/60),-
5+j-20*sin(pi/12)*cos(-(mod(i,20))*pi/60),80-
20*cos(pi/12),'o');
        [w,p]=veline(w,p,i+20*sin(pi/12)*sin(-
mod(i,20)*pi/60),-5+j-20*sin(pi/12)*cos(-
(mod(i,20))*pi/60),80-
20*cos(pi/12),i+20*sin(pi/12)*sin(-
mod(i,20)*pi/60)+20*sin(20-mod(i,20)/45*pi),-5+j-
```

```
20*sin(pi/12)*cos(-mod(i,20)*pi/60)-20*cos(20-
mod(i,20)/45*pi),80-20*cos(pi/12)-20*cos((20-
mod(i,20))/45*pi));
        scatter3(i+20*sin(pi/12)*sin(-
mod(i,20)*pi/60)+20*sin(20-mod(i,20)/45*pi),-5+j-
20*sin(pi/12)*cos(-mod(i,20)*pi/60)-20*cos(20-
mod(i,20)/45*pi),80-20*cos(pi/12)-20*cos((20-
mod(i,20))/45*pi),50,'filled');
    end
    [w,p]=Than(w,p,i,j);
    [w,p]=veline(w,p,0+i,j,80,0+i,0,sqrt(10^2-
j^2)+80); % neck
    [w,p]=vedau(w,p,i,j,-pi/60*(mod(i,20)-9)); %head
    plot3(0+i,j,80,'o');
    plot3(0+i,0,100,'*');
    j=j+t;
    if j>=3 || j<=-3
        t=-t;
    end
    axis vis3d equal;
    view(90,0);
    axis([-10 400 -50 50 0 130])
    pause(0.05)
end
```

```
function R = eulerXYZ(a1, a2, a3)
% Convert XYZ Euler angles to rotation matrix

R1 = [
1, 0, 0;
0, cos(a1), -sin(a1);
0, sin(a1), cos(a1)];

R2 = [
cos(a2), 0, sin(a2);
0, 1, 0;
-sin(a2), 0, cos(a2)];

R3 = [
cos(a3), -sin(a3), 0;
sin(a3), cos(a3), 0;
0, 0, 1];

R = R1*R2*R3;

end
```

```
function [w,p]=vedau(w,p,i,j,a1)
    r = [i; 0; sqrt(10^2-j^2)+90];
    R = eulerXYZ(0, 90,a1);
    Radius = 10;
    Height = 5;
    SideCount = 10;

    % Vertices
    vertices_0 = zeros(2*SideCount, 3);
    for i = 1:SideCount
        theta = 2*pi/SideCount*(i-1);
        vertices_0(i,:) = [Radius*cos(theta),
Radius*sin(theta), 0];
        vertices_0(SideCount+i,:) = [Radius*cos(theta),
Radius*sin(theta), Height];
    end

    vertices = r' + vertices_0*R';

    % Side faces
    sideFaces = zeros(SideCount, 4);
    for i = 1:(SideCount-1)
        sideFaces(i,:) = [i, i+1, SideCount+i+1,
SideCount+i];
    end
    sideFaces(SideCount,:) = [SideCount, 1, SideCount+1,
2*SideCount];
    % Bottom faces
    bottomFaces = [
        1:SideCount;
        (SideCount+1):2*SideCount];

    % Draw patches
    h_side = patch('Faces', sideFaces, 'Vertices',
vertices, 'FaceColor', 'y');
    h_bottom = patch('Faces', bottomFaces, 'Vertices',
vertices, 'FaceColor', 'y');

end
```

```
function [w,p]=veline(w,p,x1,y1,z1,x2,y2,z2)
    i=linspace(0,100,101);
    plot3(x1+i/100*(x2-x1),y1+i/100*(y2-
y1),z1+i/100*(z2-z1),'LineWidth',5);
    hold on
    grid on
    w=w+sqrt((x2-x1)^2+(y2-y1)^2+(z2-z1)^2);
```

```
    p=p+((x2-x1)^2+(y2-y1)^2+(z2-z1)^2)/2;
end
```

```
function [Ax,Ay,Az] = leg(l1,l2,Cx,Cy,Cz,Bx,By,Bz)
a0 = atan2((By-Cy),(Bx-Cx));
U = (Bx-Cx)/cos(a0);
K = Bz-Cz;
gama = atan2(K*l2,U*l2);
t1 = 0.5*(U^2 + K^2 + l1^2 - l2^2);
m1 = sqrt((U*l1)^2+(K*l2)^2);
a1PlusGama = asin(t1/m1);
if ((a1PlusGama<0)&&(t1>0))
    a1PlusGama = pi - a1PlusGama;
end
a1 = a1PlusGama - gama;

Ax = Cx + l1*sin(a1)*cos(a0);
Ay = Cy + l1*sin(a1)*sin(a0);
Az = Cz + l1*cos(a1);
end
```

```
function [w,p]=banchan(w,p,xOc,yOc,zOc)
     [w,p]=vedt(w,p,xOc-5,yOc-3,zOc,xOc+5,yOc-3,zOc);
     [w,p]=vedt(w,p,xOc-5,yOc+3,zOc,xOc+5,yOc+3,zOc);
     [w,p]=vedt(w,p,xOc-5,yOc-3,zOc,xOc-5,yOc+3,zOc);
     [w,p]=vedt(w,p,xOc+5,yOc-3,zOc,xOc+5,yOc+3,zOc);
end
```

```
function [w,p]=Than(w,p,i,j)
    r = [i-5; -5+j; 40];

    % Reference orientation
    R = eulerXYZ(0, 0, 0);

    % Side lengths
    Lx = 10;
    Ly = 10;
    Lz = 40;

    % Vertices
    vertices_0 = [
        0,   0,  0;  % #1
        Lx,  0,  0;  % #2
        0,  Ly,  0;  % #3
        0,   0, Lz;  % #4
        Lx, Ly,  0;  % #5
        0,  Ly, Lz;  % #6
```

```
        Lx,  0, Lz;  % #7
        Lx, Ly, Lz]; % #8

    vertices = r' + vertices_0*R';

    % Faces
    faces = [
        1, 2, 5, 3;  % #1
        1, 3, 6, 4;  % #2
        1, 4, 7, 2;  % #3
        4, 7, 8, 6;  % #4
        2, 5, 8, 7;  % #5
        3, 6, 8, 5]; % #6

    % Draw patch
    h = patch('Faces', faces, 'Vertices', vertices,
'FaceColor', "#00FFFF");

    hold on
    grid on
end
```

REFERENCE

1. C. Hernández-Santos, E. Rodriguez Leal, R. Soto, and J.L. Gordillo. Kinematics and Dynamics of a New 16 DOF Humanoid Biped Robot with Active Toe Joint (2017). https://journals.sagepub.com/doi/10.5772/52452

2. Train Humanoid Walker:

https://www.mathworks.com/help/sm/ug/humanoid_walker.html#responsive_offcanvas

3. Abhishek Kumar Kashyap, Dayal R. Parhi, Manoj Kumar Muni, Krishna Kant Pandey. A hybrid technique for path planning of humanoid robot.

https://www.sciencedirect.com/science/article/abs/pii/S1568494620305196?via%3Dihub