Introduction to Robotics (IE-410)

Assignment – 1

Group Members:
Mihir Moolchandani (202201088)
Nisharg Modi (202201346)
Kkavy Dave (202201421)
Ayush Chaudhari (202201517)

Assignment 1 – Inverse Kinematics

Aim: This experiment deals with Inverse Kinematics for robotic arm manipulation. Implement Inverse Kinematics using C++ in Arduino UNO and analysis robotic arm movements

Material Used: Arduino Uno, Arduino Compatible Shield, USB Cable, Robotic Arm, Power Adapter.

Calculation Done:

Inverse Kinematics :-

$a = x - l_3 \cdot \cos(\gamma)$

$b = y - l_3 \cdot \sin(\gamma)$

$c = \sqrt{a^2 + b^2}$

$(l_1 + l_2) > c$

{Reachable Length}

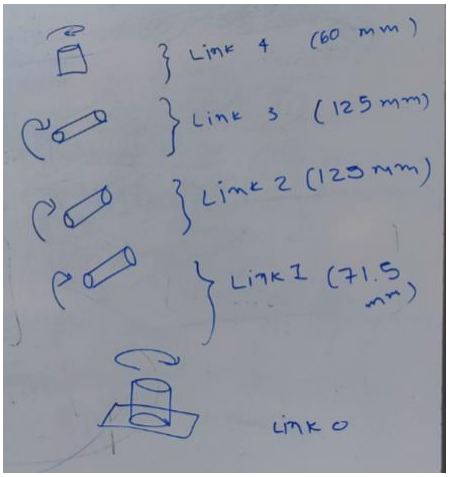
$\alpha = \cos^{-1}\left(\frac{l_1^2 + c^2 - l_2^2}{2 \cdot l_1 \cdot c}\right) \leftarrow \theta_6$

$\beta = \cos^{-1}\left(\frac{l_1^2 + l_2^2 - c^2}{2 l_1 l_2}\right) \leftarrow$

$\theta_1 = \tan^{-1}(b/a) - \alpha$

$\theta_2 = \pi - \beta$

$\theta_3 = \gamma - \theta_1 - \theta_2$; where $\gamma = \theta_1 + \theta_2 + \theta_3$



Code Used:

```
#include <Braccio.h>
#include <Servo.h>

Servo base;
Servo shoulder;
Servo elbow;
Servo wrist_rot;
Servo wrist_ver;
Servo gripper;

// Function prototype for inverse kinematics for 3 DOF
void inv_kinematics(float valuex, float valuey, float gamma);

// Link lengths (in cm)
const float Length1 = 12.5; // Shoulder - elbow
const float Length2 = 12.5; // Elbow - wrist
const float Length3 = 7.15; // Wrist to end-effector

void setup() {
  Serial.begin(9600);
  delay(1000);
  Braccio.begin();
}

void loop() {
  // Enter end-effector coordinates and orientation
  float valuex = -15, valuey = 15, gamma = 60;
  // Call inv_kinematics function and move the robotic arm
  inv_kinematics(valuex, valuey, gamma);
  delay(1000);
}

void inv_kinematics(float valuex, float valuey, float gamma) {
  // Calculate position of P3
  float valuea = valuex - (Length3 * cos(radians(gamma)));
  float valueb = valuey - (Length3 * sin(radians(gamma)));
  float valueC = sqrt(pow(valuea, 2) + pow(valueb, 2));

  // Check if position is within reachable workspace
  if ((L1 + L2) >= valueC) {
    // Calculate angles by using cosine rules at L1 and L2
    float alpha = degrees(acos((pow(Length1, 2) + pow(valueC, 2) - pow(Length2,
2)) / (2 * Length1 * valueC)));
```

```
    float beta = degrees(acos((pow(Length1, 2) + pow(Length2, 2) - pow(valueC,
2)) / (2 * Length1 * Length2)));

    // Joint angles for elbow-down configuration
    float theta1 = 180 - beta;
    float theta2 = degrees(atan2(valueb, valuea)) - alpha;
    float theta3 = gamma – (theta2 + theta1);
    // Set servo angles and move the Braccio arm
    Braccio.ServoMovement(20, 0, int(theta1), 90 + int(theta2), 90 + int(theta3),
0, 73);
  }
}
```

Method:

Use a USB cable to connect the Arduino UNO to the PC and upload the provided code by selecting 'Tools -> Board -> Arduino UNO' and choosing the correct serial port in 'Tools -> Port'.

This code is an Arduino sketch for controlling a robotic arm known as Braccio using inverse kinematics. The robotic arm has three degrees of freedom (DOF) controlled by servo motors at the base, shoulder, elbow, wrist rotation, wrist vertical, and gripper.

Libraries Included:

Braccio.h: This library provides functions to control the Braccio robotic arm.

Servo.h: This library allows control of servo motors.

inv_kinematics() Function:

This function calculates the servo angles required to position the end-effector of the robotic arm at the specified coordinates and orientation using inverse kinematics. It takes three parameters: valuex, valuey, and gamma, which represent the x and y coordinates of the end-effector and the orientation angle respectively.

First, it calculates the coordinates (a,b) by applying basic triangle laws and implementing them as shown in the calculations above. Then, it finds the values of alpha and beta by applying the basic cosine law, the implementation of which is shown in the picture shared above. From these values of alpha and beta, it further calculates the values of theta1, theta2, and theta3.

Inverse Kinematics Calculation:

The function first calculates the position of point P3 using trigonometry based on the input coordinates and orientation. It then checks if the desired position is within the reachable workspace of the robotic arm. If the position is reachable, it calculates the joint angles (theta1, theta2, and theta3) required for the elbow-down configuration of the arm.

Observation:

Here are some pictures what we observed for our robotic arm.

Robotic Arm analysis with the given valuex = -15, valuey = 15, gamma = 60

Robotic Arm analysis with the given valuex = 0, valuey = 31.5, gamma = 90

Result:

Robotic arm is able to set the values of theta1, theta2 and theta3 accordingly to the given value of (x,y) coordinate and angle of end effector gamma (orientation).

Youtube Link : https://youtu.be/JGDx4rtl6qw

GitHub Link : https://github.com/KkavyDave/Inverse-Kinematics