ME 598
Introduction to Robotics
Fall 2022

Lab 1: Robot Arm Kinematics

Group 8
10/24/22

Undergraduate:
“I pledge my honour that I have abided by the Stevens Honour System”

Graduate:
“I pledge that I have abided by the Graduate Student Code of Academic Integrity.”

The report has been prepared by:
1. Rakesh Gummapu
2. Shiv Bhatt
3. Sharvil Singh

Abstract

This lab used the programs and concepts of Forward and Inverse kinematics used in the previous lab to operate the DoBot bot. The sample code was tested against our code to check its accuracy. Moreover, the outputs of those codes were used to manipulate the DoBot to pick a ping pong ball and place it at a different location. And finally, the obstacle avoidance algorithm was tested by making the DoBot avoid another ping pong ball while performing the pick and place task.
Validation of the previous codes were performed by all three members in the lab along with pick and place and obstacle avoidance. We found that few parameters in our output were inaccurate and needed a recalculation. In pick and place activity, the home location was not accurate and we needed to code better to get more accuracy.

Introduction

We have a task to use the concept of Forward and Inverse kinematics on an actual robotic arm. This is succeeded by pick and place operation where multiple kinematics conversions take place to execute the path. This requires us to start from a home position. This is a position which is the most reliable and precise configuration. We need it because, regardless of where the arm stopped during the previous execution, we can send the arm to this position at the start to guarantee a correct output. The third and final experiment is to implement an obstacle avoidance algorithm. We have to perform the previous pick and place task but without colliding with another ping pong ball placed in its path.
To use our kinematics output, first need to test for its accuracy. This will be performed and tested against the sample code of DoBot. We will input same parameters in both codes and then compare the actual position (measured by a physical ruler on the DoBot platform), the calculated distance of the sample code and the calculated code of our code. This should come out to be similar. For pick and place, the DoBot should have reliable home position and complete the task. Obstacle avoidance algorithm should also do the said tasks accurately.

Results

Part A
This part tests our Forward Kinematics code accuracy. The readings were taken from the actual DoBot platform, the sample code and the user(our) code. The input parameters are in the form of θ_1, θ_2, θ_3, θ_4, θ_5 of the DoBot joints.

Reading 1
Input Parameters = (0,75,75,0,0)

	x-axis	***y-axis***	***z-axis***

Sample O/P	25	0	16.5
Our O/P	25	0	15.1

Reading 2
Input Parameters = (50, 25, -25,0,0)

	x-axis	***y-axis***	***z-axis***
Sample O/P	21.7	25.8	9.21
Our O/P	26.1	25.84	15.73
Calc O/P	18.9	23.3	13

Reading 3
Input Parameters = (50,75,75,0,0)

	x-axis	***y-axis***	***z-axis***
Sample O/P	16.1	19.1	16.5
Our O/P	16	19.1	6.9
Calc O/P	12.5	7.5	15

Part B
This part tests our Inverse Kinematics code accuracy. The readings were taken from the actual DoBot platform, the sample code and the user(our) code. The input parameters are in the form of x, y, and z co-ordinates of the DoBot end-effector. The method used was that we entered a random joint angle matrix and got a position matrix. This matrix was then used as an input in our Inverse Kinematics code and its output was matched with the input of the Sample Code.

Reading 1
Input Parameters = (27.4, -9.98, 6.71)

	$\boldsymbol{\theta_1}$	$\boldsymbol{\theta_2}$	$\boldsymbol{\theta_3}$	$\boldsymbol{\theta_4}$	$\boldsymbol{\theta_5}$
I/P in Sample Code	-20	50	-85	53	0
O/P in Our Code	-20.013	54.202	-83.131	28.929	0

Reading 2
Input Parameters = (21.6, -12.5, 16.5)

	$\boldsymbol{\theta_1}$	$\boldsymbol{\theta_2}$	$\boldsymbol{\theta_3}$	$\boldsymbol{\theta_4}$	$\boldsymbol{\theta_5}$
I/P in Sample Code	-30	75	-75	0	0
O/P in Our Code	-30.058	89.54	-91.33	1.780	0

Reading 3

Input Parameters = (21.5, -7.83, 11.3)

	θ_1	θ_2	θ_3	θ_4	θ_5
I/P in Sample Code	-20.0109	79.9582	-100.037	20.097	0
O/P in Our Code	-21.011	85.604	-106.32	20.718	0

Part C

This part implements the path planning and pick and place function. We selected 3 combinations of start and end points, calculated the input parameters and ran the code.

The DoBot would start at the home position, go to pick the ball, drop it off and return to home. We have taken intermediate configurations so as to avoid dragging the end-effector along the platform and colliding with an obstacle.

Reading 1

Input parameters:

- Home Location = (24.9, -0.868, 17)
- Pick Point = (13.4, -4.09, 3.6)
- Intermediate Point = (12.9, -4.1, 17)
- Place Point = (29.1, -1.02, 2.92)

Output Video link: https://www.youtube.com/watch?v=RvQjIjnOp1o

Reading 2

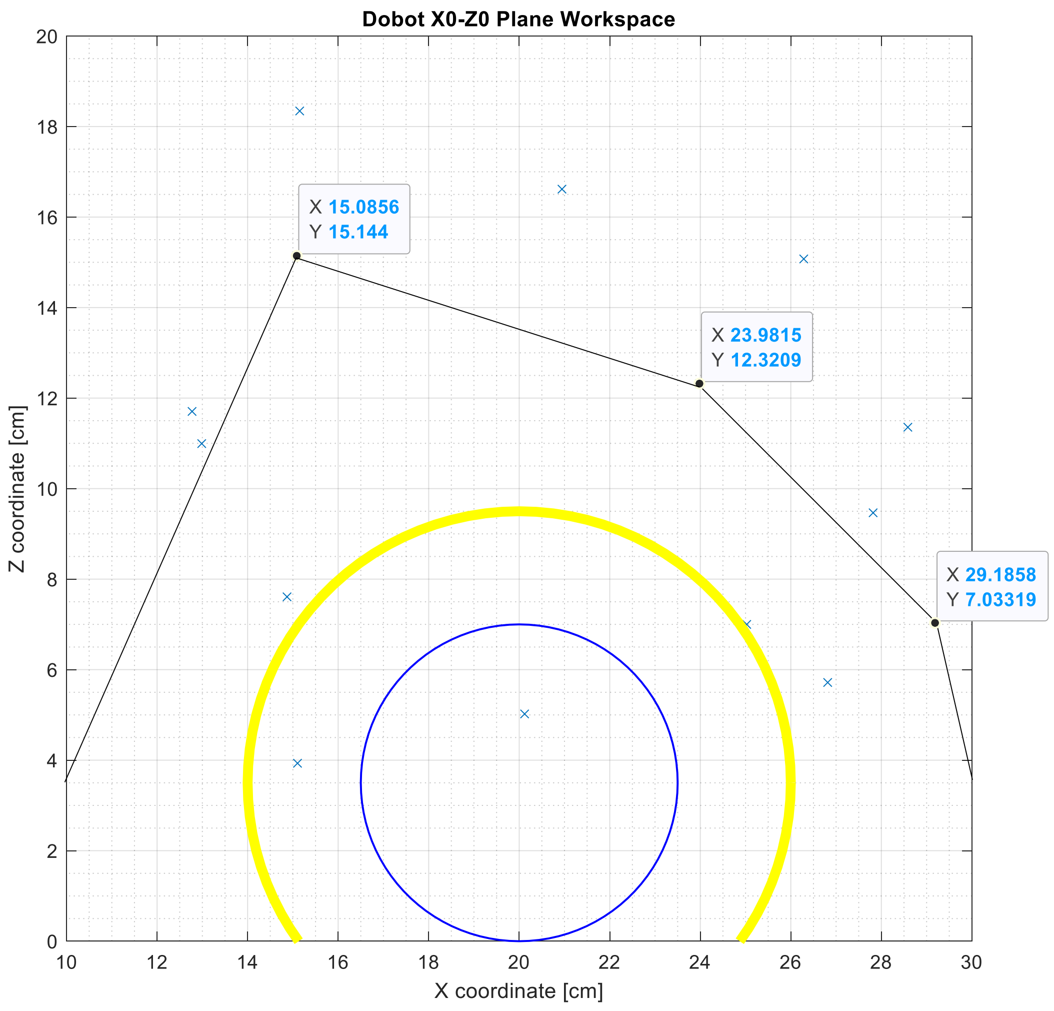
Input parameters:

- Home Location = (24.9, -0.868, 17)
- Pick Point = (24.7, -16.6, 3.5)
- Intermediate Point = (24.7, -16.6, 17)
- Place Point = (24.2, 8.82, 3.5)

Output Video link: https://www.youtube.com/watch?v=n85dAdO4VzQ

Part D

In this part, we generated a random graph using *rand_map.m* file provided in the lab. We added a function to generate circle representing the obstacle and another circle representing its circle of influence. After generating the graph, we manually selected points avoiding the obstacle and gave them as input in the code. Below is the graph output, way points for the end-effector and video link showing the result.



Input parameters:

- Home Location = (24.9, -0.868, 17)
- Pick Location = (18, -4, 2.5)
- Intermediate 1 = (15.08, 0 15.44)
- Intermediate 2 = (23.98, 0, 12.32)
- Intermediate 3 = (29.1858, 0, 7.03319)
- Place Location = (30, 0, 2.5)

Output video link:

Discussions

Part A and Part B

- We did see discrepancies in our results. This is occurring because of calculation error in our main code. It can have been caused by creating a wrong DH table.
- When we had different start positions, we got different output for the same input. We think this is because when the code is deciding the parameters for the path, a different start position might be causing minor calculation differences. These can be minimized by having the home position as an intermediate position.

Conclusion

We verified the accuracy of our kinematics code. We learnt how to use them to manipulate an actual robotic arm. We implemented obstacle avoidance algorithm along with nearest neighbour method to create a safe path for the arm.

Abstract

Code for Part C

```
%% Group_8 Dobotsimulation
% Rakesh; Shiv; Sharvil
% Last revision: Feb. 22, 2018
%
% Stevens Institute of Technology
% Mechanical Engineering Department
% ME 598, Introduciton to Robotics


%%

% Clean up the workspace etc. so that variables are clear, as is the
% instrument (COM) configuration
clear all;
close all;
clc;
instrreset

% Configure Dobot parameters for MATLAB communication
serport = 'COM8';
dobot = serial(serport, 'BaudRate' ,9600);

% Establish serial (USB) connection
fopen(dobot)

% Read current Dobot state
% Output the measured configuration and vacuum condition
[q_act, pump] = dobotReadDH(dobot)
```

```
%%

% Settings for simulator
multiplot   = 1;
fignummulti = 1;
fignumXZ    = 2;

figure(fignummulti)
clf

figure(fignumXZ)
clf
% Vacuum parameter: 0=OFF, 1=ON
% Home Position
desSuction = 0;
q_des = [-2; 76; -75; 0; 0]
dobotWriteDH(dobot, q_des, desSuction);
pause(3)

% Position #1
%bringing the dobot to pick the ball from the position_1
desSuction = 0;
q_des = [-34; 30; -71; 45; 0]
dobotWriteDH(dobot, q_des, desSuction);
 pause(3)

 % on the suction to pick and hold the ball
desSuction = 1;
q_des = [-34; 30; -71; 45; 0]
dobotWriteDH(dobot, q_des, desSuction);
pause(3)

%pick the ball from the position_1 and lift it up
desSuction = 1;
q_des = [-34; 76; -75; 0; 0]
dobotWriteDH(dobot, q_des, desSuction);
pause(3)

%come to the home position after picking the ball from position_1
desSuction = 1;
q_des = [-2; 76; -75; 0; 0]
dobotWriteDH(dobot, q_des, desSuction);
pause(3)

% from home position of the bobot place the ball at position_2
desSuction = 1;
q_des = [20; 22; -58; 0; 0]
dobotWriteDH(dobot, q_des, desSuction);
pause(3)

% release the ball after placing at the desired position_2
desSuction = 0;
q_des = [20; 22; -58; 0; 0]
dobotWriteDH(dobot, q_des, desSuction);
pause(3)

% bring the dobot to the home position after completeing the task
% Home Position
desSuction = 0;
```

```
q_des = [-2; 76; -75; 0; 0]
dobotWriteDH(dobot, q_des, desSuction);
pause(3)

% Simulate behavior
dobotPlotXZ(q_des,fignumXZ);
dobotPlot(q_des,fignummulti,multiplot);

% Make Dobot move to desired configuration + vacuum state
dobotWriteDH(dobot, q_des, desSuction);

% Pause to give Dobot time to reach commanded configuration
pause(3)

% Read current Dobot state
[q_act, pump] = dobotReadDH(dobot)

% Check configuration error
err_q = q_des - q_act

%%

% Disable Dobot object and release COM port (important)
fclose(dobot)
clear all;
instrreset
```