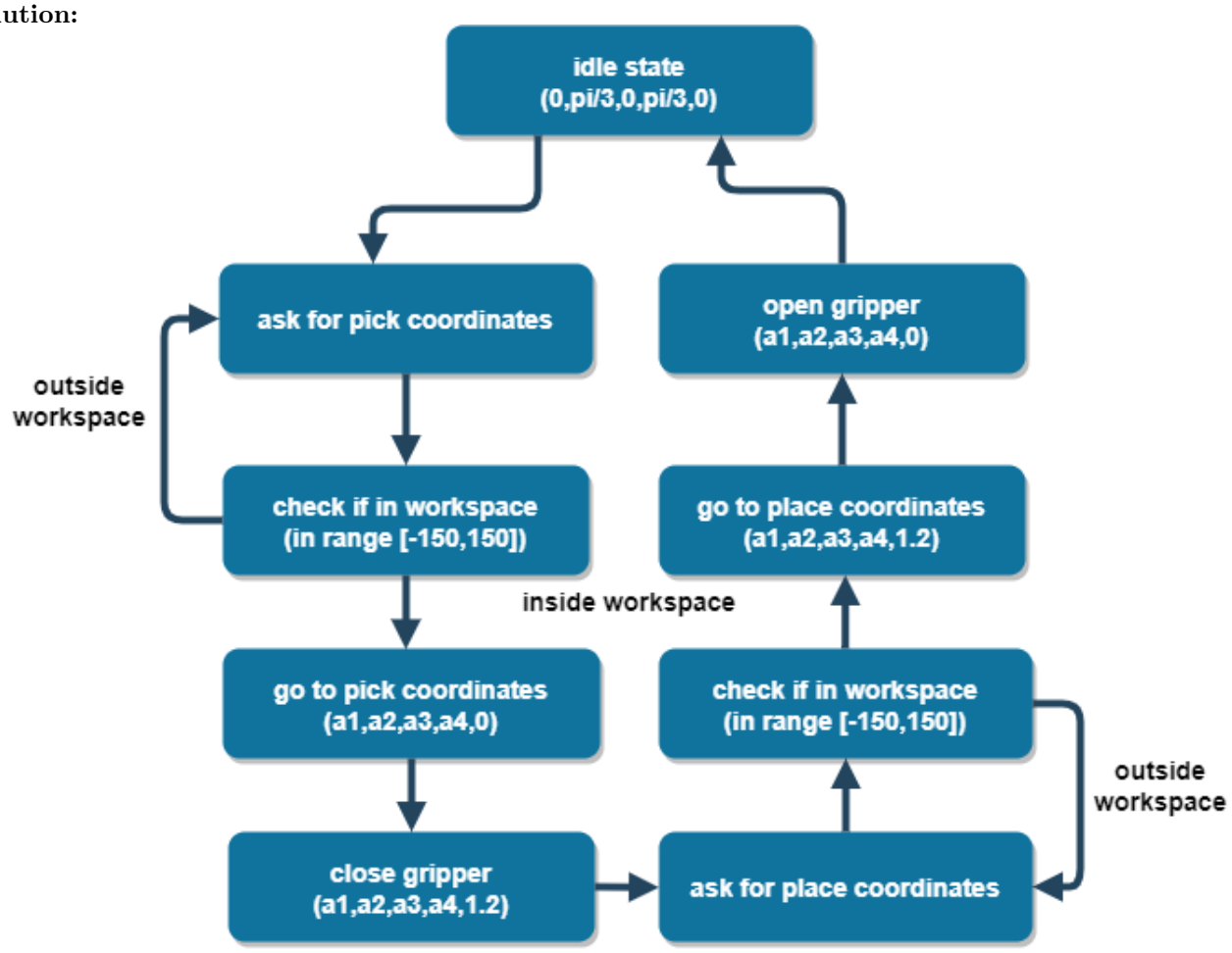
Robotics Lab 06

Syeda Manahil Wasti, Syed Mujtaba Hassan, Aatiqa Khalid

March 27, 2023

1. Draw a state-transition diagram of an FSM corresponding to the following scenario:
   - System is in idle state till it receives pick location, (x1, y1, z1, ϕ1) and place location, (x2, y2, z2, ϕ2).
   - Geometry of the object to be picked and placed, including its orientation, is known before hand.
   - The locations can be assumed to lie in the interior of the manipulator's workspace, and the object is in an orientation so that it can be picked.
   - System should verify the final placement location, before determining that the task has concluded.
   - Smooth motion and accurate placement is desirable.

Solution:



2. Implement the system described by the previous FSM in MATLAB for Phantom X Pincher and the cube object. In addition to your implementation code, submit an explanation of your strategy, a video of your best execution, and identify and comment on points of improvement.

Solution:
Implementation code:

```
b = Arbotix('port', 'COM9', 'nservos', 5);
b.setpos([0,pi/3,0,pi/3,0], [65, 65, 65, 65, 65]);
f = input('arm is in idle state. enter 1 if you want to pick an object and 0 if
    you want to remain in idle state: ');
if (f == 1)
    disp('enter pick coordinates: ');
    a1 = input('enter first joint angle = ');
    a2 = input('enter second joint angle = ');
    a3 = input('enter third joint angle = ');
    a4 = input('enter fourth joint angle = ');
    if (((a1 > (-5*pi)/6) && (a2 > (-5*pi)/6) && (a3 > (-5*pi)/6) && (a4 > (-5*pi
        )/6)) && ((a1 < (5*pi)/6) && (a2 < (5*pi)/6) && (a3 < (5*pi)/6) && (a4 <
        (5*pi)/6)))
        b.setpos([0,pi/3,0,pi/3,0], [65, 65, 65, 65, 65]);
        pause(2)
        b.setpos([a1,a2,a3,a4,0], [65, 65, 65, 65, 65]);
        pause(2)
        b.setpos([a1,a2,a3,a4,1.2], [65, 65, 65, 65, 65]);
        pause(2)
        b.setpos([0,pi/3,0,pi/3,1.2], [65, 65, 65, 65, 65]);
        pause(2)
        disp('enter place coordinates: ')
        a5 = input('enter fifth joint angle = ');
        a6 = input('enter sixth joint angle = ');
        a7 = input('enter seventh joint angle = ');
        a8 = input('enter eighth joint angle = ');
        if (((a5 > (-5*pi)/6) && (a6 > (-5*pi)/6) && (a7 > (-5*pi)/6) && (a8 >
            (-5*pi)/6)) && ((a5 < (5*pi)/6) && (a6 < (5*pi)/6) && (a7 < (5*pi)/6)
            && (a8 < (5*pi)/6)))
            b.setpos([a5,a6,pi/3,a8,1.2], [65, 65, 65, 65, 65]);
            pause(2)
            b.setpos([a5,a6,a7,a8,1.2], [65, 65, 65, 65, 65]);
            pause(2)
            b.setpos([a5,a6,a7,a8,0], [65, 65, 65, 65, 65]);
            pause(2)
            b.setpos([a5,a6,pi/3,a8,0], [65, 65, 65, 65, 65])
            pause(2)
            b.setpos([0,pi/3,0,pi/3,0], [65, 65, 65, 65, 65]);
        else
            a5 = input('enter fifth joint angle = ');
            a6 = input('enter sixth joint angle = ');
            a7 = input('enter seventh joint angle = ');
            a8 = input('enter eighth joint angle = ');
            b.setpos([a5,a6,pi/3,a8,1.2], [65, 65, 65, 65, 65]);
            pause(2)
            b.setpos([a5,a6,a7,a8,1.2], [65, 65, 65, 65, 65]);
            pause(2)
            b.setpos([a5,a6,a7,a8,0], [65, 65, 65, 65, 65]);
            pause(2)
```

```
            b.setpos([a5,a6,pi/3,a8,0], [65, 65, 65, 65, 65])
            pause(2)
            b.setpos([0,pi/3,0,pi/3,0], [65, 65, 65, 65, 65]);
        end
    else
        a1 = input('enter first joint angle = ');
        a2 = input('enter second joint angle = ');
        a3 = input('enter third joint angle = ');
        a4 = input('enter fourth joint angle = ');
        b.setpos([0,pi/3,0,pi/3,0], [65, 65, 65, 65, 65]);
        pause(2)
        b.setpos([a1,a2,a3,a4,0], [65, 65, 65, 65, 65]);
        pause(2)
        b.setpos([a1,a2,a3,a4,1.2], [65, 65, 65, 65, 65]);
        pause(2)
        b.setpos([0,pi/3,0,pi/3,1.2], [65, 65, 65, 65, 65]);
        pause(2)
        disp('enter place coordinates: ')
        a5 = input('enter fifth joint angle = ');
        a6 = input('enter sixth joint angle = ');
        a7 = input('enter seventh joint angle = ');
        a8 = input('enter eighth joint angle = ');
        if (((a5 > (-5*pi)/6) && (a6 > (-5*pi)/6) && (a7 > (-5*pi)/6) && (a8 >
           (-5*pi)/6)) && ((a5 < (5*pi)/6) && (a6 < (5*pi)/6) && (a7 < (5*pi)/6)
           && (a8 < (5*pi)/6)))
            b.setpos([a5,a6,pi/3,a8,1.2], [65, 65, 65, 65, 65]);
            pause(2)
            b.setpos([a5,a6,a7,a8,1.2], [65, 65, 65, 65, 65]);
            pause(2)
            b.setpos([a5,a6,a7,a8,0], [65, 65, 65, 65, 65]);
            pause(2)
            b.setpos([a5,a6,pi/3,a8,0], [65, 65, 65, 65, 65])
            pause(2)
            b.setpos([0,pi/3,0,pi/3,0], [65, 65, 65, 65, 65]);
        else
            a5 = input('enter fifth joint angle = ');
            a6 = input('enter sixth joint angle = ');
            a7 = input('enter seventh joint angle = ');
            a8 = input('enter eighth joint angle = ');
            b.setpos([a5,a6,pi/3,a8,1.2], [65, 65, 65, 65, 65]);
            pause(2)
            b.setpos([a5,a6,a7,a8,1.2], [65, 65, 65, 65, 65]);
            pause(2)
            b.setpos([a5,a6,a7,a8,0], [65, 65, 65, 65, 65]);
            pause(2)
            b.setpos([a5,a6,pi/3,a8,0], [65, 65, 65, 65, 65])
            pause(2)
            b.setpos([0,pi/3,0,pi/3,0], [65, 65, 65, 65, 65]);
        end
    end
else
    b.setpos([0,pi/3,0,pi/3,0], [65, 65, 65, 65, 65]);
end
```

Explanation of strategy:
Our program starts by bringing the arm to the idle position which is at joint angles (0, pi/3, 0, pi/3, 0) and has gripper open. After that we are asked if we want to pick and place an object or remain at idle position. Input '1' is sent for executing the pick and place code and '0' is sent as input to remain at idle state.

The pick and place code is executed by first asking us for pick coordinates i.e., joint angles for pick position. Those are then checked to see if they lie in the workspace. If yes, then arm moves to that location and closes gripper. After firmly holding the object in place with the gripper, arm moves back to idle position. If the pick coordinates are not in workspace then user is asked for input again.

After that, we are asked to input the place coordinates and then those are checked for being in the workspace of the arm. If yes, then the arm moves to that location and opens gripper. Then, rather then directly going to the idle state after opening gripper, the arm is first raised to a certain height above the place coordinates and then returns to idle state in the end. This ensures that the block is no displaced from the place location as arm moves abruptly to the idle state. Also, if the place coordinates are not in workspace then user is asked for input again.

For our execution we used the following coordinates (joint angles):
pick position: [0,pi/3,1.27409,0]
place position: [pi/2,pi/3,1.27409,0]

Video of best execution:
https://youtu.be/ZY2zAZKbfW8

Points of improvement:

- We are currently taking direct input of joint angles. The code can be modified to take cartesian coordinates and then those can be converted to joint angles for a better execution.
- We have hard-coded the gripper value for the object we are picking i.e., the block. However, for different objects, gripper value should be variable.
- We have used 65 as the speed whereas it can also be taken as input from user so the arm can move at variable speeds.

3. Use the DH parameters and homogeneous transformation, 0T4, obtained in the previous lab to find the Jacobian for the manipulator in the lab.

- For convenience, a MATLAB function createA(theta, d, a, alpha) is available on canvas to easily create homogeneous transformations in symbolic form.
- Define your joint variables θ_i as functions of time, so that you can differentiate them.
  syms theta_1(t) theta_2(t) theta_3(t) theta_4(t)
  A1 = createA(theta_1, 'd_1', 0, -pi/2)
- The homogeneous transformation you'll obtain will be a 4×4 matrix function of t. To extract a particular entry of this matrix, you'll have to first evaluate it at a value of t and save it in an intermediate variable. For example, if B is a matrix symbolic function and you want to find matrix entry (1, 2), then use:
  tempVar = B(t);
  entry = tempVar(1, 2);
- You can find derivative of a symbolic expression using the MATLAB function diff. For example, diff(f, x) computes $\frac{\partial f}{\partial x}$.
- Chain rule will frequently yield simplified expressions.

Solution:

```
syms('theta1','theta2','theta3','theta4')
a1 = 0;
a2 = 10;
a3 = 10;
a4 = 10;
d1 = 13;
d2 = 0;
d3 = 0;
d4 = 0;
alpha1 = pi/2;
alpha2 = 0;
alpha3 = 0;
alpha4 = 0;

A1 = createA(theta1,d1,a1,alpha1);
A2 = createA(theta2,d2,a2,alpha2);
A3 = createA(theta3,d3,a3,alpha3);
A4 = createA(theta4,d4,a4,alpha4);

T_01 = A1;
T_02 = A1*A2;
T_03 = A1*A2*A3;
T_04 = A1*A2*A3*A4;

o4 = T_04(1:3,4);

Jv = [diff(o4, theta1) diff(o4, theta2) diff(o4, theta3) diff(o4, theta4)];

Jw = [[0;0;1], T_01(1:3,3), T_02(1:3,3), T_03(1:3,3)];

J = [Jv;Jw]
```

J =

$$\begin{pmatrix} 10\cos(\theta_4)\,\sigma_{13} - 10\cos(\theta_2)\sin(\theta_1) + 10\sin(\theta_4)\,\sigma_{14} + 10\sin(\theta_1)\sin(\theta_2)\sin(\theta_3) - 10\cos(\theta_2)\cos(\theta_3)\sin(\theta_1) & \sigma_{11} - \sigma_{12} - 10\cos(\theta_1)\sin(\theta_2) - \sigma_6 - \sigma_5 & \sigma_{11} - \sigma_{12} - \sigma_6 - \sigma_5 & \sigma_{11} - \sigma_{12} \\ 10\cos(\theta_1)\cos(\theta_2) - 10\cos(\theta_4)\,\sigma_{15} - 10\sin(\theta_4)\,\sigma_{16} - 10\cos(\theta_1)\sin(\theta_2)\sin(\theta_3) + 10\cos(\theta_1)\cos(\theta_2)\cos(\theta_3) & \sigma_9 - \sigma_{10} - 10\sin(\theta_1)\sin(\theta_2) - \sigma_4 - \sigma_3 & \sigma_9 - \sigma_{10} - \sigma_4 - \sigma_3 & \sigma_9 - \sigma_{10} \\ 0 & 10\cos(\theta_2) + \sigma_8 - \sigma_7 + \sigma_2 - \sigma_1 & \sigma_8 - \sigma_7 + \sigma_2 - \sigma_1 & \sigma_2 - \sigma_1 \\ 0 & \sin(\theta_1) & \sin(\theta_1) & \sin(\theta_1) \\ 0 & -\cos(\theta_1) & -\cos(\theta_1) & -\cos(\theta_1) \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

where

$$\sigma_1 = 10\sin(\theta_4)\,(\cos(\theta_2)\sin(\theta_3) + \cos(\theta_3)\sin(\theta_2))$$

$$\sigma_2 = 10\cos(\theta_4)\ (\cos(\theta_2)\cos(\theta_3) - \sin(\theta_2)\sin(\theta_3))$$

$$\sigma_3 = 10\cos(\theta_3)\sin(\theta_1)\sin(\theta_2)$$

$$\sigma_4 = 10\cos(\theta_2)\sin(\theta_1)\sin(\theta_3)$$

$$\sigma_5 = 10\cos(\theta_1)\cos(\theta_3)\sin(\theta_2)$$

$$\sigma_6 = 10\cos(\theta_1)\cos(\theta_2)\sin(\theta_3)$$

$$\sigma_7 = 10\sin(\theta_2)\sin(\theta_3)$$

$$\sigma_8 = 10\cos(\theta_2)\cos(\theta_3)$$

$$\sigma_9 = 10\sin(\theta_4)\,\sigma_{13}$$

$$\sigma_{10} = 10\cos(\theta_4)\,\sigma_{14}$$

$$\sigma_{11} = 10\sin(\theta_4)\,\sigma_{15}$$

$$\sigma_{12} = 10\cos(\theta_4)\,\sigma_{16}$$

$$\sigma_{13} = \sin(\theta_1)\sin(\theta_2)\sin(\theta_3) - \cos(\theta_2)\cos(\theta_3)\sin(\theta_1)$$

$$\sigma_{14} = \cos(\theta_2)\sin(\theta_1)\sin(\theta_3) + \cos(\theta_3)\sin(\theta_1)\sin(\theta_2)$$

$$\sigma_{15} = \cos(\theta_1)\sin(\theta_2)\sin(\theta_3) - \cos(\theta_1)\cos(\theta_2)\cos(\theta_3)$$

$$\sigma_{16} = \cos(\theta_1)\cos(\theta_2)\sin(\theta_3) + \cos(\theta_1)\cos(\theta_3)\sin(\theta_2)$$