Assignment6

April 26, 2020

[ ]:
```
#import required librires
import numpy as np
import math
from vpython import *
```

[ ]:
```
#define arms length
l1 = 15
l2 = 50
l3 = 60
l4 = 40
```

[ ]:
```
#find the location of each arm and join using FK
def drawRobot(theta2,theta3,theta4):
    theta2 = (theta2 / 180) * np.pi
    theta3 = (theta3 / 180) * np.pi
    theta4 = (theta4 / 180) * np.pi

    T01 = np.float32([[1, 0, 20],
                      [0, 1, 20],
                      [0, 0, 1]])

    T12 = np.float32([[math.cos(theta2), -math.sin(theta2), 0],
                      [math.sin(theta2), math.cos(theta2), l1],
                      [0, 0, 1]])

    T23 = np.float32([[math.cos(theta3), -math.sin(theta3), 0],
                      [math.sin(theta3), math.cos(theta3), l2],
                      [0, 0, 1]])

    T34 = np.float32([[math.cos(theta4), -math.sin(theta4), 0],
                      [math.sin(theta4), math.cos(theta4), l3],
                      [0, 0, 1]])

    T45 = np.float32([[1, 0, 0],
                      [0, 1, l4],
                      [0, 0, 1]])
```

```
    T02 = np.dot(T01, T12)
    T03 = np.dot(T02, T23)
    T04 = np.dot(T03, T34)
    T05 = np.dot(T04, T45)

    return T01,T02,T03,T04,T05
```

[ ]:
```
#initialize scene and robot at 0,0,0
T01,T02,T03,T04,T05 = drawRobot(0,0,0)

# scene.width = scene.height = 600
scene2 = canvas(title='Robot hand - Fahad Alsuliman',
     width=800, height=400,
     center=vector(100,100,0), background=color.blue)
```

[ ]:
```
#same function as above but return the end effector location only
def endEffectorLocation(theta2,theta3,theta4):
    theta2 = (theta2 / 180) * np.pi
    theta3 = (theta3 / 180) * np.pi
    theta4 = (theta4 / 180) * np.pi

    T01 = np.float32([[1, 0, 20],
                      [0, 1, 20],
                      [0, 0, 1]])

    T12 = np.float32([[math.cos(theta2), -math.sin(theta2), 0],
                      [math.sin(theta2), math.cos(theta2), l1],
                      [0, 0, 1]])

    T23 = np.float32([[math.cos(theta3), -math.sin(theta3), 0],
                      [math.sin(theta3), math.cos(theta3), l2],
                      [0, 0, 1]])

    T34 = np.float32([[math.cos(theta4), -math.sin(theta4), 0],
                      [math.sin(theta4), math.cos(theta4), l3],
                      [0, 0, 1]])

    T45 = np.float32([[1, 0, 0],
                      [0, 1, l4],
                      [0, 0, 1]])

    T02 = np.dot(T01, T12)
    T03 = np.dot(T02, T23)
    T04 = np.dot(T03, T34)
    T05 = np.dot(T04, T45)

    return T05[0,2],T05[1,2]
```

[ ]:
```
#draw the robot and obstecles using vPython
part1 = cylinder( pos=vec(T01[0,2],T01[1,2],0),␣
↪axis=vec(T02[0,2]-T01[0,2],T02[1,2]-T01[1,2],0))
part2 = cylinder( pos=vec(T02[0,2],T02[1,2],0),␣
↪axis=vec(T03[0,2]-T02[0,2],T03[1,2]-T02[1,2],0))
part3 = cylinder( pos=vec(T03[0,2],T03[1,2],0),␣
↪axis=vec(T04[0,2]-T03[0,2],T04[1,2]-T03[1,2],0))
part4 = cylinder( pos=vec(T04[0,2],T04[1,2],0),␣
↪axis=vec(T05[0,2]-T04[0,2],T05[1,2]-T04[1,2],0))

joint1 = sphere(pos=vector(T02[0,2],T02[1,2],0), radius=2, color=color.green)
joint2 = sphere(pos=vector(T03[0,2],T03[1,2],0), radius=2, color=color.green)
joint3 = sphere(pos=vector(T04[0,2],T04[1,2],0), radius=2, color=color.green)

end_effector = sphere(pos=vector(T05[0,2],T05[1,2],0), radius=2, color=color.
↪red)

Obst  = cylinder(pos=vector(80, 130, 0),
                 axis=vector(0, 0, 1), radius=7, color=color.yellow)

secObst = cylinder(pos=vector(120, 70, 0),
                 axis=vector(0, 0, 1), radius=7, color=color.yellow)

goalPoint = sphere(pos=vector(100, 120, 0), radius=4, color=color.green )
goal = vector(100, 120, 0)

i = 1
```

[ ]:
```
#the cost function which has six cases +1 -1 degree for each joint + field␣
↪value penelty
def cost(end_effector , goal,theta2,theta3,theta4,):
    d= 0.2

    costFirstAnglePlusX,costFirstAnglePlusY =␣
↪endEffectorLocation(theta2+d,theta3,theta4)
    distanceCost1 = sqrt(((costFirstAnglePlusX - goal.x) ** 2) +␣
↪((costFirstAnglePlusY - goal.y) ** 2))
    fieldcost = 0
    fieldcost = fieldcost +␣
↪fieldValueCalcuation(costFirstAnglePlusX,costFirstAnglePlusY)
    curcost1 = distanceCost1 + fieldcost

    costFristAngleMinX,costFristAngleMinY =␣
↪endEffectorLocation(theta2-d,theta3,theta4)
    distanceCost2 = sqrt(((costFristAngleMinX - goal.x) ** 2) +␣
↪((costFristAngleMinY - goal.y) ** 2))
```

```
    fieldcost = 0
    fieldcost = fieldcost + fieldValueCalcuation(costFristAngleMinX, costFristAngleMinY)
    curcost2 = distanceCost2 + fieldcost

    costSecondAnglePlusX,costSecondAnglePlusY = endEffectorLocation(theta2,theta3+d,theta4)
    distanceCost3 = sqrt(((costSecondAnglePlusX - goal.x) ** 2) + ((costSecondAnglePlusY - goal.y) ** 2))
    fieldcost = 0
    fieldcost = fieldcost + fieldValueCalcuation(costSecondAnglePlusX, costSecondAnglePlusY)
    curcost3 = distanceCost3 + fieldcost


    costSecondAngleMinX,costSecondAngleMinY = endEffectorLocation(theta2,theta3-d,theta4)
    distanceCost4 = sqrt(((costSecondAngleMinX - goal.x) ** 2) + ((costSecondAngleMinY - goal.y) ** 2))
    fieldcost = 0
    fieldcost = fieldcost + fieldValueCalcuation(costSecondAngleMinX, costSecondAngleMinY)
    curcost4 = distanceCost4 + fieldcost


    costThirdAnglePlusX,costThirddAnglePlusY = endEffectorLocation(theta2,theta3,theta4+ d)
    distanceCost5 = sqrt(((costThirdAnglePlusX - goal.x) ** 2) + ((costThirddAnglePlusY - goal.y) ** 2))
    fieldcost = 0
    fieldcost = fieldcost + fieldValueCalcuation(costThirdAnglePlusX, costThirddAnglePlusY)
    curcost5 = distanceCost5 + fieldcost

    CostThirdAngleMinX,CostThirdAngleMinY = endEffectorLocation(theta2,theta3,theta4 -d)
    distanceCost6 = sqrt(((CostThirdAngleMinX - goal.x) ** 2) + ((CostThirdAngleMinY - goal.y) ** 2))
    fieldcost = 0
    fieldcost = fieldcost + fieldValueCalcuation(CostThirdAngleMinX, CostThirdAngleMinY)
    curcost6 = distanceCost6 + fieldcost

    total_cost = [curcost1,curcost2,curcost3,curcost4,curcost5,curcost6]
    low = 120000
    seq = 0
```

```
    for i in range(0,5):
        if total_cost[i] < low :
            low = total_cost[i]
            seq = i

    if seq == 0 :
        newTheta2 = theta2 + d
        newTheta3 = theta3
        newTheta4 = theta4

    elif seq == 1:
        newTheta2 = theta2 - d
        newTheta3 = theta3
        newTheta4 = theta4

    elif seq == 2:
        newTheta2 = theta2
        newTheta3 = theta3 + d
        newTheta4 = theta4

    elif seq == 3:
        newTheta2 = theta2
        newTheta3 = theta3 - d
        newTheta4 = theta4

    elif seq == 4:
        newTheta2 = theta2
        newTheta3 = theta3
        newTheta4 = theta4 + d

    elif seq == 5:
        newTheta2 = theta2
        newTheta3 = theta3
        newTheta4 = theta4 - d


    return newTheta2,newTheta3,newTheta4
```

```
[ ]: #field value calcuation
def fieldValueCalcuation(positionX ,positionY):

    distance = sqrt(((positionX - Obst.pos.x) ** 2) + ((positionY - Obst.pos.y)
 ↪** 2))
    if(distance >  25 ):
        return 0
    elif(distance <= 25 and distance >15):
        fieldValue = math.log(25/distance)*30
```

```python
    elif(distance<= 15):
        fieldValue = math.log(15/distance)*30
    return fieldValue
```

[ ]:
```python
#initialize theta and start calculating next angles and draw
theta1 = 0
theta2 = 0
theta3 = 0
theta4 = 0
while ((end_effector.pos - goal).mag >3):
    rate(50)
    theta2,theta3,theta4 = cost(end_effector.pos,goal,theta2,theta3,theta4)
    T01,T02,T03,T04,T05 = drawRobot(theta2,theta3,theta4)
    part1.pos = vec(T01[0, 2], T01[1, 2], 0)
    part1.axis = vec(T02[0, 2] - T01[0, 2], T02[1, 2] - T01[1, 2], 0)
    part2.pos = vec(T02[0, 2], T02[1, 2], 0)
    part2.axis = vec(T03[0, 2] - T02[0, 2], T03[1, 2] - T02[1, 2], 0)
    part3.pos = vec(T03[0, 2], T03[1, 2], 0)
    part3.axis = vec(T04[0, 2] - T03[0, 2], T04[1, 2] - T03[1, 2], 0)
    part4.pos = vec(T04[0, 2], T04[1, 2], 0)
    part4.axis = vec(T05[0, 2] - T04[0, 2], T05[1, 2] - T04[1, 2], 0)
    joint1.pos  = vector(T02[0, 2], T02[1, 2], 0)
    joint2.pos = vector(T03[0, 2], T03[1, 2], 0)
    joint3.pos = vector(T04[0, 2], T04[1, 2], 0)
    end_effector.pos = vector(T05[0, 2], T05[1, 2], 0)
```