ECE 470: Lab 5
Inverse· Kinematics

Yutong Xie
NetID: yutongx6
TA: Ben Walt
Section: Wednesday 2PM

November 15, 2019

1 How to find six joint variables

1.1 theta1

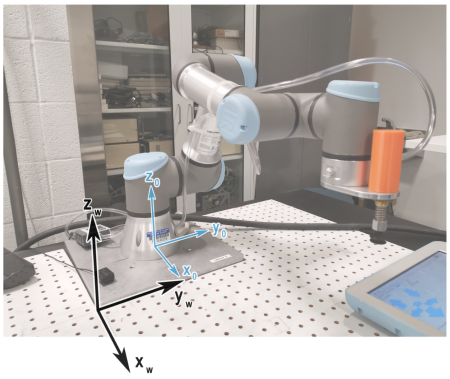

Figure 1: Correct location and orientation of the world frame.

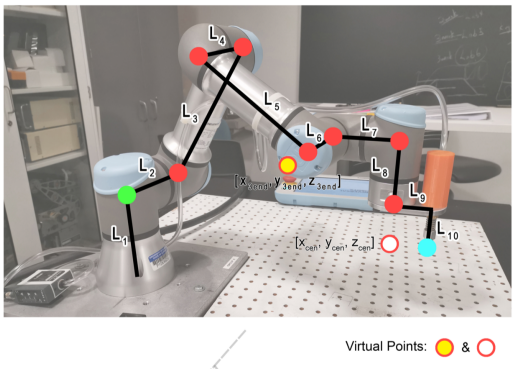

Figure 2: Dimensinos of UR3

We can calculate theta1 using thetaL - theta_1 as shown in Fig 3. To find the value of thetaL and theta11, we need xcen, ycen, and zcen, which can be derived from the coordinate of gripper shown in Fig 4. However, we only got the coordinate of gripper in world frame, and we need to transform it to base frame. As shown in Fig 1, we can get the following equation.

$$xgrip = xwgrip + 0.15$$
$$ygrip = ywgrip - 0.15$$
$$zgrip = zwgrip$$

When we get xgrip, ygrip, and zgrip, we can calculate the coordinate of center using following equations. l09 refers the length of L9 in Fig 2.

$$xcen = xgrip - l09 * cos(yaw)$$
$$ycen = ygrip - l09 * sin(yaw)$$
$$zcen = zgrip$$

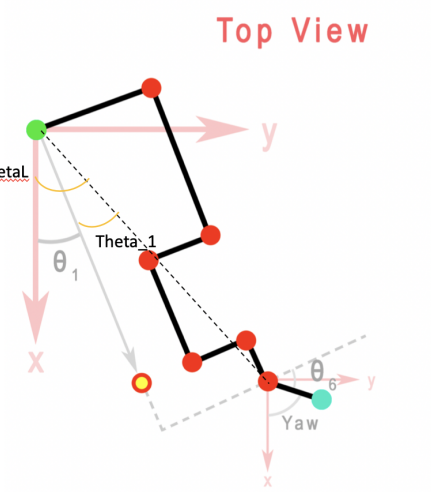

Figure 3: Top view of UR3

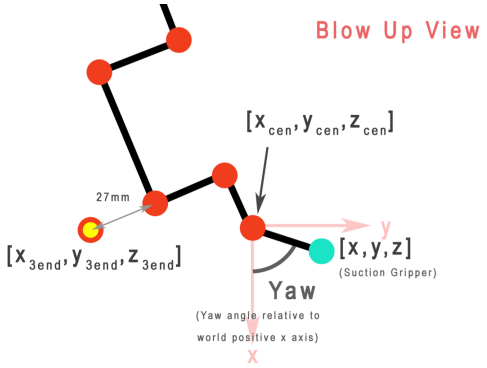

Figure 4: Blow up view of UR3

After we get xcen, ycen, and zcen, we can get the theta using the following equations.

$$thetaL = arctan2(ycen, xcen)$$
$$length = \sqrt{xcen^2 + ycen^2}$$
$$theta11 = arcsin((l02 - l04 + l06)/length)$$
$$theta = thetaL - theta11$$

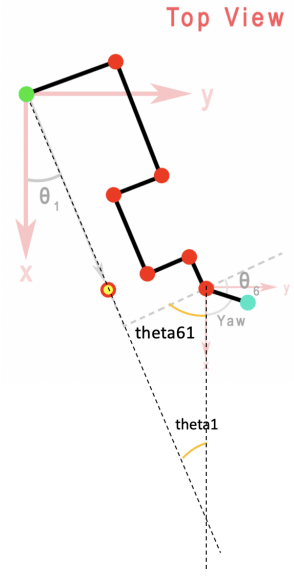

Figure 5: Geometry approach for theta6.

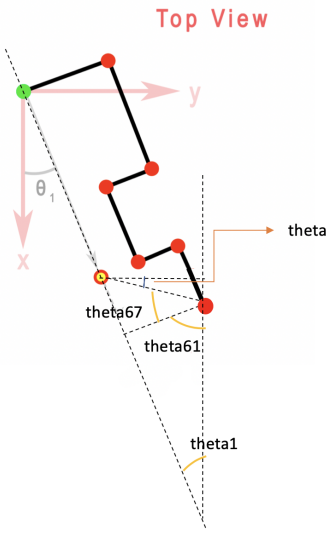

Figure 6: Geometry approach for 3end.

Fig 5 shows how we calculate the value of theta6. According to the theory for alternate interior angle, we can get one more theta1. Then, theta61 should be pi/2-theta1. Hence, we can calculate the value

of theta6.

$$\begin{aligned} theta6 &= \pi - theta61 - yaw \\ &= \pi - (\pi/2 - theta1) - yaw \\ &= \pi/2 + theta1 - yaw \end{aligned}$$

1.2 theta2, theta3, theta4

After calculating theta1 and theta6, we need to find the projected end point (x3end, y3end, z3end) for calculating theta2, theta3, and theta4. Fig 6 shows the geometry relationship of several angels. We need to find the value of theta for calculating the coordinate of point 3end. Using the theory for alternate interior angle, we get one more theta1 here, and theta61 is pi/2-theta1. Also, we can get theta67 according to the length of two right angle edges. Then, we can get the complementary angle of theta.

$$\begin{aligned} theta61 &= \pi/2 - theta1 \\ theta67 &= arctan2(l07, (l06 + 0.027)) \\ theta &= \pi/2 - (\pi - theta67 - theta61) \\ &= \pi/2 - (\pi - theta67 - \pi/2 + theta1) \\ &= theta67 - theta1 \end{aligned}$$

After we get the value of theta, we can know exactly the coordinate of 3end by using theta and the distance between cen point and 3end point.

$$\begin{aligned} l67 &= sqrtl07^2 + (l06 + 0.027)^2 \\ x3end &= xcen - l67 * sin(theta) \\ y3end &= ycen - l67 * cos(theta) \\ z3end &= zcen + l10 + l08 \end{aligned}$$

When we get the coordinate of 3end, we can calculate the value of theta2, theta3, and theta4. As shown in Fig 7, we define four angles to help us find the value of theta2,3,4. Theta21 can be calculated use arctan2 function, and theta41 is the complementary angle of theta21. Theta22 and theta42 can be calculated by using law of cosines. Then, theta2 is just the sum of theta21 and theta22, and we need to take the opposite number here. Theta3 should be the sum of theta22 and theta42 based on xterior angle theorem. And theta4 is simply achieved using theta42 plus theta41 and minus pi/2. Also, we should take the opposite number here. The theta4 defined in the lab manual is relative to horizontal line, which is not convenient for later calculation. Hence, I define theta4 relative to vertical line here. The following equations demonstrate the detail about how to calculte theta2,3,4.

$$\begin{aligned} xy3end &= sqrtx3end^2 + y3end^2 \\ z &= z3end - l01 \\ xyz3end &= sqrtxy3end^2 + z^2 \\ theta21 &= arctan2(z, xy3end) \\ theta41 &= PI/2 - theta21 \\ theta22 &= arccos((l03^2 + xyz3end^2 - l05^2)/(2 * l03 * xyz3end)) \\ theta42 &= arccos((l05^2 + xyz3end^2 - l03^2)/(2 * l05 * xyz3end)) \\ theta2 &= -(theta21 + theta22) \\ theta3 &= theta22 + theta42 \\ theta4 &= -(theta41 + theta42 - \pi/2) \end{aligned}$$

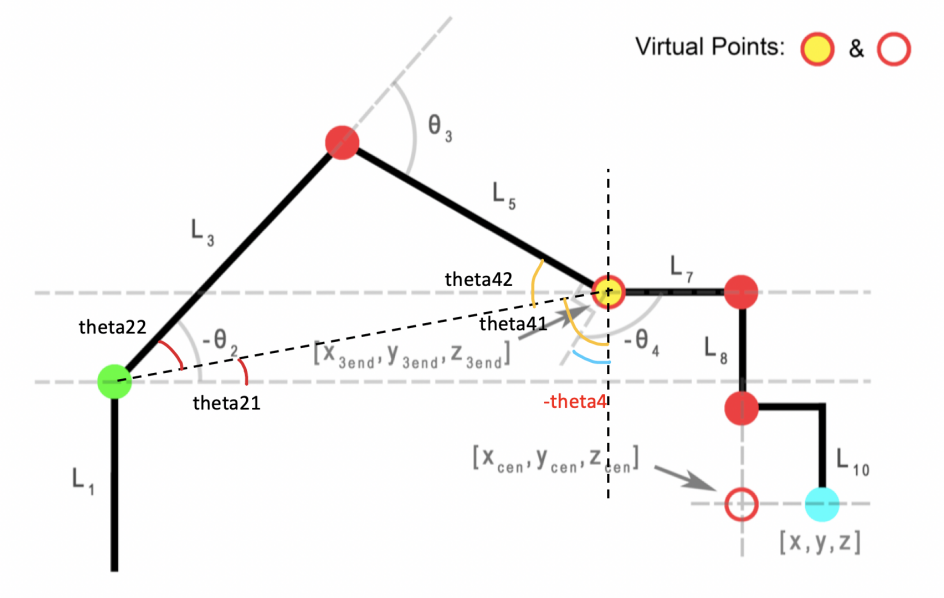

Figure 7: Side View Stick Pictorial of UR3

1.3 theta5

In this lab, theta5 is always -pi/2.

2 Test result

TA gave two points with yaw angles. The forward kinematics gives the predicted destined point in base frame, but we need to transfer it to world frame as we mentioned before.

2.1 Test set 1

- The set of values: (0.15 -0.1 0.25 -45)
- The measured position of the tool frame: [0.155, -0.095, 0.247]
- The scalar error between the measured and expected positions: $7.68 * 10^{-3}$

2.2 Test set 2

- The set of values: (0.2 0.4 0.05 45)
- The measured position of the tool frame: [0.203, 0.402, 0.052]
- The scalar error between the measured and expected positions: $4.12 * 10^{-3}$

```
ur3@ur3-6:~/catkin_yutongx6$ rosrun lab5pkg_py lab5_exec.py 0.15 -0.1 0.25 -45

xWgrip: 0.15, yWgrip: -0.1, zWgrip: 0.25, yaw_WgripDegree: -45

theta21 is: 0.775063964874
theta41 is: 0.795732361921
0.76440171836
0.67417054718 3
theta1 to theta6: [-1.0126054438827126, -1.4757474144855514, 1.5316386943854468,
 -0.055891279899895618, -1.5707963267948966, 1.3435890463096323]

Foward kinematics calculated:

[[  7.07106781e-01   7.07106781e-01  -4.41024339e-17   3.04596194e-01]
 [ -7.07106781e-01   7.07106781e-01   7.06280619e-17  -2.54596194e-01]
 [  0.00000000e+00  -3.46944695e-18   1.00000000e+00   2.50000000e-01]
 [  0.00000000e+00   0.00000000e+00   0.00000000e+00   1.00000000e+00]]

[INFO] [1573678177.150889]: Just published again driver_msg
[INFO] [1573678178.700765]: Destination is reached!
```

Figure 8: Test result for set 1

```
ur3@ur3-6:~/catkin_yutongx6$ rosrun lab5pkg_py lab5_exec.py 0.2 0.4 0.05 45

xWgrip: 0.2, yWgrip: 0.4, zWgrip: 0.05, yaw_WgripDegree: 45

theta21 is: 0.111027327101
theta41 is: 1.45976899969
0.677090810509
0.537910093403
theta1 to theta6: [0.30122178852359965, -0.93802148856998335, 1.8298344688830865
, -0.89181298031310341, -1.5707963267948966, 1.086619951921048]

Foward kinematics calculated:

[[  7.07106781e-01  -7.07106781e-01  -2.12046983e-16   3.54596194e-01]
 [  7.07106781e-01   7.07106781e-01  -6.58777932e-17   2.54596194e-01]
 [  1.11022302e-16   0.00000000e+00   1.00000000e+00   5.00000000e-02]
 [  0.00000000e+00   0.00000000e+00   0.00000000e+00   1.00000000e+00]]

[INFO] [1573678219.768539]: Just published again driver_msg
[INFO] [1573678221.268438]: Destination is reached!
```

Figure 9: Test result for set 2

3 Error analysis

The scalar error between the measured and expected positions can be caused by several reasons. First, the scale of UR3 may not correspond to the approximate dimensions given by the manual. Second, some errors will appear when we use the ruler to measure the position of end-effector. We can try to use the more accurate device to measure the position, such as laser rangefinder.

Appendices

A Python Script: lab5_func.py

```python
#!/usr/bin/env python
import numpy as np
import math
from scipy.linalg import expm
# PI = 3.1415926535
from lab5_header import *

"""
Use 'expm' for matrix exponential.
Angles are in radian, distance are in meters.
"""

def skew(vector):
    return np.array([[0.0, -vector[2], vector[1]],[vector[2], 0.0, -vector[0]],[-
    vector[1], vector[0], 0.0]])

def SBracket(vector,theta):
  new_vector = np.ones((6,1))
  w = np.ones((3,1))
  v = np.ones((3,1))

  for i in range(3):
    w[i,0] = vector[i]
    v[i,0] = vector[3 + i]

  w_bracket = skew(w)
  S_theta = np.hstack([w_bracket*theta, v*theta])
  S_theta = np.vstack([S_theta,np.array([0.0, 0.0, 0.0, 0.0])])
  return S_theta

def Get_MS():
  # ==================== Your code starts here ====================#
  # Fill in the correct values for a1~6 and q1~6, as well as the M matrix
  M = np.eye(4)
  S = np.zeros((6,6))

  pos_1 = np.array([[0.0],[0.0],[0.152]])
  pos_2 = np.array([[0.0],[0.12],[0.152]])
  pos_3 = np.array([[0.244],[0.12],[0.152]])
  pos_4 = np.array([[0.457],[0.027],[0.152]])
  pos_5 = np.array([[0.457],[0.11],[0.152]])
  pos_6 = np.array([[0.54],[0.11],[0.152]])

  pos_W = np.array([[0.0],[0.0],[0.0]])
  pos_E = np.array([[0.54],[0.243],[0.212]])

  P = pos_E - pos_W
```

```
47    R = np.array([[0,-1,0],[0,0,-1],[1,0,0]])
48    M = np.hstack([R, P])
49    M = np.vstack([M,np.array([0.0,0.0,0.0,1.0])])
50
51    joint_positions = []
52    joint_positions.append(pos_1)
53    joint_positions.append(pos_2)
54    joint_positions.append(pos_3)
55    joint_positions.append(pos_4)
56    joint_positions.append(pos_5)
57    joint_positions.append(pos_6)
58
59    q = []
60    for i in range(0,len(joint_positions)):
61      q.append(joint_positions[i]-pos_W)
62
63    #joint rotation axes for the UR3
64    w1 = np.array([[0.0],[0.0],[1.0]])
65    w2 = np.array([[0.0],[1.0],[0.0]])
66    w3 = np.array([[0.0],[1.0],[0.0]])
67    w4 = np.array([[0.0],[1.0],[0.0]])
68    w5 = np.array([[1.0],[0.0],[0.0]])
69    w6 = np.array([[0.0],[1.0],[0.0]])
70
71    w = []
72    w.append(w1)
73    w.append(w2)
74    w.append(w3)
75    w.append(w4)
76    w.append(w5)
77    w.append(w6)
78
79    v = []
80    for i in range(0,len(joint_positions)):
81      v.append(-np.cross(w[i],q[i],axis=0))
82
83
84    # screw axes for each joint on the UR3
85    SA = []
86    for i in range(0,len(joint_positions)):
87      # w_bracket = skew(w[i])
88      Si = np.vstack([w[i], v[i]])
89      # Si = np.vstack([Si, np.array([0.0, 0.0, 0.0, 0.0])])
90      SA.append(Si)
91
92    S = np.hstack([SA[0],SA[1],SA[2],SA[3],SA[4],SA[5]])
93
94    # ===================================================================#
95    return M, S
96
97
98  """
99  Function that calculates encoder numbers for each motor
100 """
101 def lab_fk(theta1, theta2, theta3, theta4, theta5, theta6):
102
103   # Initialize the return_value
104   return_value = [None, None, None, None, None, None]
105
106   print("Foward kinematics calculated:\n")
107
108   # ================== Your code starts here ==================#
```

```
  theta = np.array([theta1,theta2,theta3,theta4,theta5,theta6])
  T = np.eye(4)

  M, S = Get_MS()

  e =[]

  for i in range(6):
    exps = expm(SBracket(S[:,i], theta[i]))
    e.append(exps)

  T = M
  for i in range(5, -1, -1):
    T = e[i].dot(T)


  #===========================================================#

  print(str(T) + "\n")

  return_value[0] = theta1 + PI
  return_value[1] = theta2
  return_value[2] = theta3
  return_value[3] = theta4 - (0.5*PI)
  return_value[4] = theta5
  return_value[5] = theta6

  return return_value


"""
Function that calculates an elbow up Inverse Kinematic solution for the UR3
"""
def lab_invk(xWgrip, yWgrip, zWgrip, yaw_WgripDegree):

    # theta1 to theta6
  thetas = [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]

  l01 = 0.152
  l02 = 0.120
  l03 = 0.244
  l04 = 0.093
  l05 = 0.213
  l06 = 0.083
  l07 = 0.083
  l08 = 0.082
  l09 = 0.0535
  l10 = 0.059   # thickness of aluminum plate is around 0.006

  yaw = yaw_WgripDegree * PI / 180

  xgrip = xWgrip + 0.15
  ygrip = yWgrip - 0.15
  zgrip = zWgrip - 0.008

  xcen = xgrip - l09*math.cos(yaw)
  ycen = ygrip - l09*math.sin(yaw)
  zcen = zgrip

  # theta1
  theta1 = np.arctan2(ycen,xcen)
  l = np.sqrt(xcen**2+ycen**2)
```

```
    theta111 = np.arcsin((l02-l04+l06)/l)
    thetas[0] = theta1 - theta111          # Default value Need to Change

    # theta6
    thetas[5] = PI/2 - yaw + thetas[0]     # Default value Need to Change

    l67 = np.sqrt(l07**2+(l06+0.027)**2)
    theta67 = np.arctan2(l07,(l06+0.027))

    x3end = xcen - l67*np.sin(theta67-thetas[0])
    y3end = ycen - l67*np.cos(theta67-thetas[0])
    z3end = zcen + l10 + l08

    xy3end = np.sqrt(x3end**2 + y3end**2)
    z = z3end - l01
    xyz3end = np.sqrt(xy3end**2 + z**2)

    theta21 = np.arctan2(z,xy3end)
    print("theta21 is: " + str(theta21))
    theta41 = PI/2 - theta21
    print("theta41 is: " + str(theta41))

    theta22 = np.arccos((l03**2 + xyz3end**2 - l05**2) / (2*l03*xyz3end))
    xxxx = (l03**2 + xyz3end**2 - l05**2) / (2*l03*xyz3end)
    print(xxxx)
    theta42 = np.arccos((l05**2 + xyz3end**2 - l03**2) / (2*l05*xyz3end))
    yyyy = (l05**2 + xyz3end**2 - l03**2) / (2*l05*xyz3end)
    print(yyyy)

    thetas[1]= - (theta21 + theta22)        # Default value Need to Change
    thetas[2]= theta22 + theta42            # Default value Need to Change
    thetas[3]= - (theta41 + theta42 - PI/2) # Default value Need to Change, need + (0.5*
      PI) for compensation

    thetas[4]=-PI/2                         # Default value Need to Change

    print("theta1 to theta6: " + str(thetas) + "\n")

    return lab_fk(float(thetas[0]), float(thetas[1]), float(thetas[2]), \
                  float(thetas[3]), float(thetas[4]), float(thetas[5]) )
```