ROBOT LIBRARIAN

Krishna Suresh Bhatu (UID: 116304764)

December 19, 2018

Abstract

This is the report on final project of ENPM 662. The project is titled as Robot Librarian. It is the prototype simulation for the Robot Librarian project. This product will be a robotic assistant to the librarian and help to maintain the arrangements of the books in the library.

1 Introduction

For prototyping the simulation we will be using the KUKA YouBot moving robotic arm model. The simulation will be done in the V-rep software and the software is written in MATLAB. The program will run with the MATLAB API for V-rep, forming the connecting link between the two softwares.
The project includes the calculations of the forward and inverse kinematics for the KUKA YouBot. The equations obtained from the forward and inverse kinematics will be verified by a MATLAB script and also by simulation in V-Rep. Also, the project includes the demonstration of the robot performing the task of picking and placing the object from one place to another. This will form the base for the future development of the product.

2 Motivation

For the past many years, the automation industry has been creeping into every industry and this is helping to make the process more efficient and cost effective. And recently it has entered the areas of household and human interaction. This has been possible because of the advancements in the controls of the systems incorporating the features of human safety for providing the robot a work-space in between the people.
Thus, this project of Robot Librarian will make the life of librarian easy and also benefit the library by efficient managing of the books in the library. But this is not the main motivation behind the project.A library is a place with long alleys and high bookshelves.Surprisingly a calm and peaceful place like a library can also be prone to accidents. This is mainly because of the high bookshelves for which the librarian have to climb up the ladder to manage the books, which unfortunately sometimes lead to accidents of slipping and falling off the ladder. And it also becomes difficult for the librarian when it comes to lifting heavy books and placing them at their place which sometimes can be at the top of the shelf.
Second problem is that when the documents or the artifacts in the library are

very important with a historical perspective which want to be preserved from and form of decay. So such documents should be handled with care. Even the oils from the human hand can be unsafe for the old document. Thus, such documents can be handled by the robot which will handle such documents with great precision and care.
Third problem is when the library is big enough and very busy, it becomes difficult for the librarians to keep track of the books and arrange them back on the shelves. And generally such tasks cannot be done while the library is working and is very busy. So the robot can do these jobs after the library is closed and with the QR tag attached to each book, it can know the exact location where the book is to be places and also knows the quickest path to reach that destination. Thus it will manage the books with great time-efficiency during the after hours and reduce the jobs of the librarian.
Hence, these are the motivation for making these project. Where yet again the robotic system performs the tasks for the human, not by taking their jobs, but by assisting them in their jobs.

3 Assumptions

- All the joints of the robot are rigid.
- For the simulation purpose we assume that the book and the shelve both are cuboids.
- We assume that the robot already has perception and path planning capabilities, to identify the environment and to move within the working environment.
- For simulation the size of the robot is small so we assume that the height of shelve is reachable to robot, but for practical scenerios we can develop much larger robot to reach the high book shelves, but the forward and inverse kinematics for such robot with 5DOF will be the same.
- We assume that the surface is flat and the motion of the joints is smooth.
- Any external disturbances are ignored. (Robot will always reach the exact desired angle)
- The mass of the load is ignored.

4 Theory and Calculations

The following is the KUKA YouBot which we will be using for our project, to demonstrate the working of the Robot Librarian. It is the 5-DOF robotic arm with all the joints which are revolute. It has a gripper attached at the end-effector. It also has a moving base with omni-directional wheels. Thus the robot can move in both X and Y directions. Following are the actual dimentions for the robot and we will be using the same for simulation:

Figure 1: KUKA YouBot.

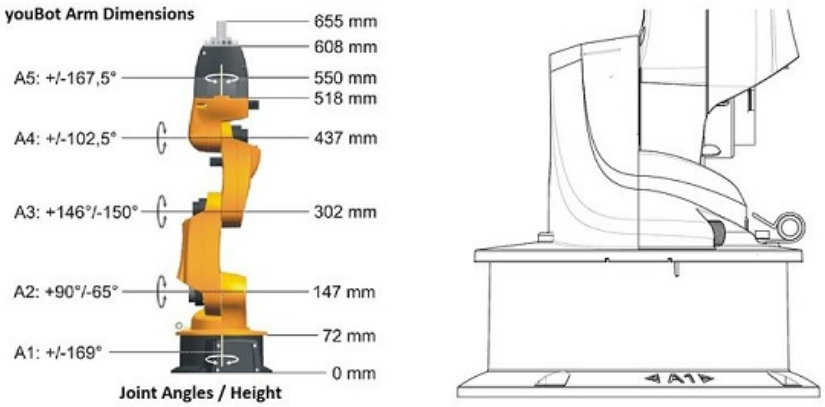


Figure 2: KUKA YouBot Dimentions.

4.1 Forward Kinematics

Forward kinematics is used to compute the final position of the end effector of the robot with the specified joint angles or position using the kinematic equations of the robot.
We will be using the DH (Denavit-Hartenberg) method to compute the forward kinematics. This method considers four parameters for calculating the transformation matrix which will determine the end effector position and orientation. The parameters considered are the rotation and displacement in the X and Z direction for all the joints. Thus, we calculate the transformation matrix for each joint using these parameters. Following is how we calculate the transformation matrix.

$$A_i = Rot_{z,\theta_i} Trans_{z,d_i} Trans_{x,a_i} Rot_{x,\alpha_i}$$

$$= \begin{bmatrix} c_{\theta_1} & -s_{\theta_1} & 0 & 0 \\ s_{\theta_1} & c_{\theta_1} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{\alpha_1} & -s_{\alpha_1} & 0 \\ 0 & s_{\alpha_1} & c_{\alpha_1} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} c_{\theta_1} & -s_{\theta_1} c_{\alpha_1} & s_{\theta_1} s_{\alpha_1} & a_i c_{\theta_1} \\ s_{\theta_1} & c_{\theta_1} c_{\alpha_1} & -c_{\theta_1} s_{\alpha_1} & a_i s_{\theta_1} \\ 0 & s_{\alpha_1} & -c_{\alpha_1} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(1)

After computing the transformation matrix for all the joints, we can multiply all the matrices to get the final transformation matrix of the robotic arm.
For example, we consider a robotic arm with 5 joints like KUKA YouBot. So, the transformation matrix can be computed as follows;

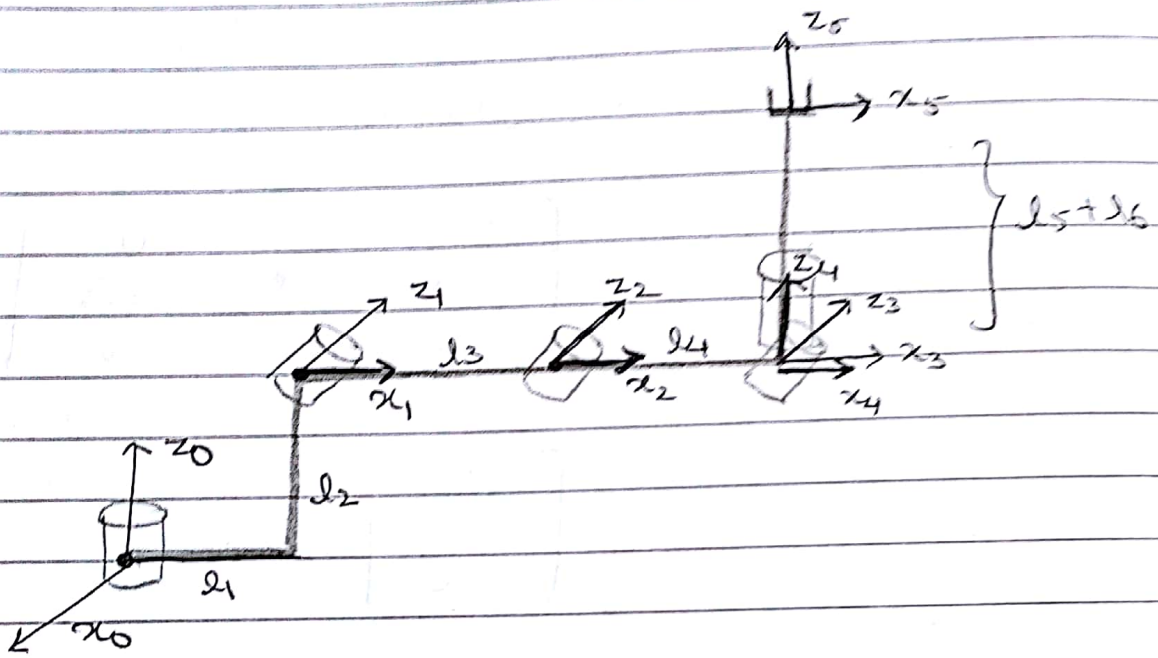
$$T_5^0 = A_1 A_2 A_3 A_4 A_5 \tag{2}$$

Now, we will start calculating the forward kinematics for our robot
Joint Axis Frames
Folowing images shows the frame structure of the joints which will be used to determine the DH Table and then used to find the transformation matrix.
DH Table
This is the DH table which shows the transformation of one joint to another.

T	θ_i	$\dot{i}$	i	α_i
0 to 1	$\theta_1 + 90$	l_2	l_1	-90
1 to 2	$\theta_2 - 90$	0	l_3	0
2 to 3	θ_3	0	l_4	0
3 to 4	$\theta_4 + 90$	0	0	90
4 to 5	$\theta_5 - 90$	$l_5 + l_6$	0	90

Now we calculate the transformation matrix for each joint. And T_5^0 is directly calculated in MATLAB code when required.

Frame Diagram



DH Table is given by;

T	θ	d	a	α
$0 \rightarrow 1$	θ_1+90	l_2	l_1	-90
$1 \rightarrow 2$	θ_2	0	l_3	0
$2 \rightarrow 3$	θ_3	0	l_4	0
$3 \rightarrow 4$	θ_4	0	0	90
$4 \rightarrow 5$	θ_5	l_5+l_6	0	0

The Transformation equations are given as follows;

$$A_1 = \begin{bmatrix} -\sin\theta_1 & 0 & -\cos\theta_1 & -l_1\sin\theta_1 \\ \cos\theta_1 & 0 & -\sin\theta_1 & l_1\cos\theta_1 \\ 0 & -1 & 0 & l_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & l_3\cos\theta_2 \\ \sin\theta_2 & \cos\theta_2 & 0 & l_3\sin\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 & 0 & l_4\cos\theta_3 \\ \sin\theta_3 & \cos\theta_3 & 0 & l_4\sin\theta_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_4 = \begin{bmatrix} \cos\theta_4 & 0 & \sin\theta_4 & 0 \\ \sin\theta_4 & 0 & -\cos\theta_4 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_5 = \begin{bmatrix} \cos\theta_5 & -\sin\theta_5 & 0 & 0 \\ \sin\theta_5 & \cos\theta_5 & 0 & 0 \\ 0 & 0 & 1 & l_5+l_6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

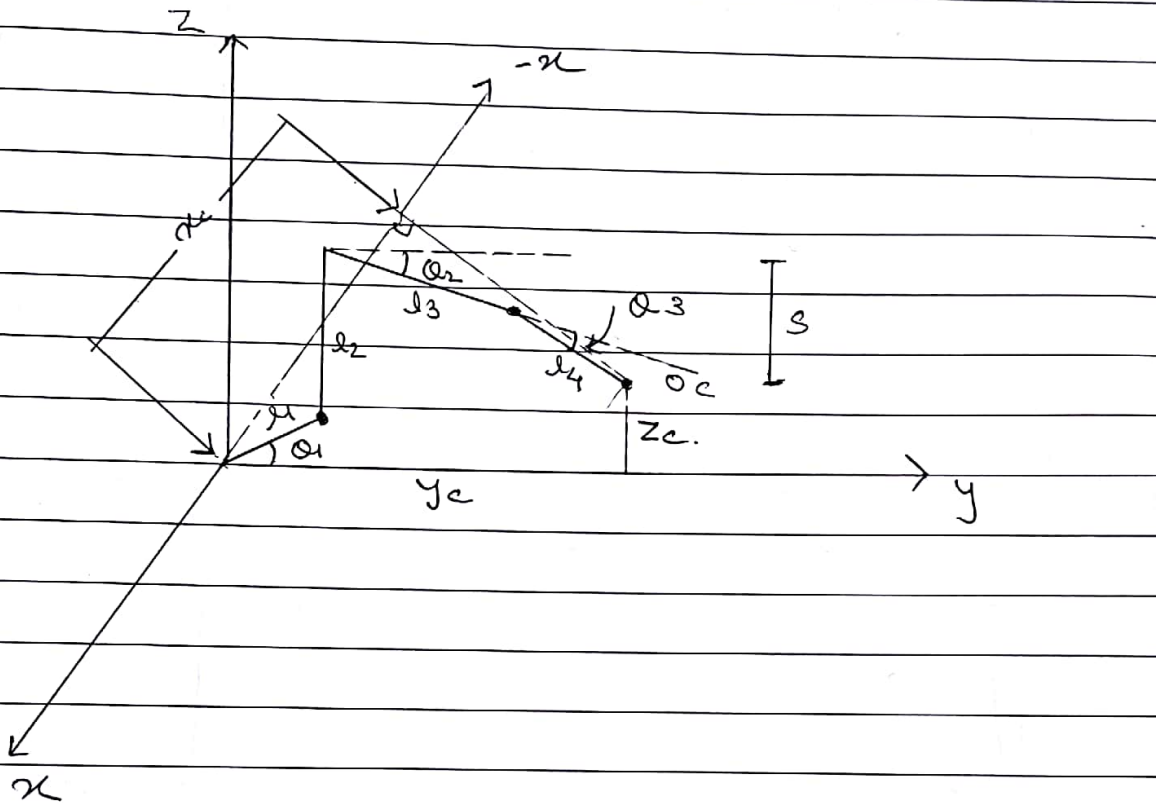
$\therefore \quad T = A_1 A_2 A_3 A_4 A_5$

T Matrix can be calculated in MATLAB directly.

4.2 Inverse Kinematics

Inverse kinematics is the calculation of the joint angles or joints positions when the end effector position and orientation are given. It is the method used to control the robotic arm and translate to the desired location when the end effector(desired position) is known.
We will calculate the inverse kinematics with the method of kinematic decoupling. It is the method used when we have more than 3 joints, as the simple analytical method is very difficult to compute. In this method we first seperate out the 3 joint and find the relative joint angles using simple geometric approach. Then by using the angles that we found we for the transformation matrix from base frame to joint 2 frame R_3^0. Then we use the input rotation matrix and the found rotation matrix to compute R_5^3
$R_5^3 = (R_3^0)^{-1} * R_{in}$ Thus we can get the other two angles by analysing the found matrix. Following are the inverse kinematics calculations:

Check for the calculations on the next page.



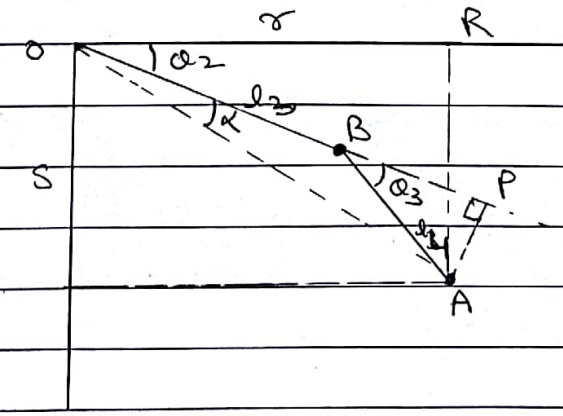
From Figure we get

$$r = \sqrt{x_c^2 + y_c^2}$$ (r is projection of Oc on x-y plane)

and, $S = l_2 - z_c$.

From Figure; $\theta_1 = \text{atan2}(-x_c, y_c)$

From above figure we can say;



In ΔAPB; we can say that;

$$BP = l_4 \cos\theta_3$$

$PA = l_4 \sin\theta_3$.

Now in ΔOAP;

By pythagoras theorem;

$$OA^2 = OP^2 + PA^2$$

$$\therefore OA^2 = (l_3 + l_4\cos\theta_3)^2 + (l_4\sin\theta_3)^2$$

But in ΔORA;

$$OA^2 = r^2 + s^2$$

Hence;

Substituting in above equation, we get;

$$r^2 + s^2 = l_3^2 + 2l_3l_4\cos\theta_3 + l_4^2\cos^2\theta_3 + l_4^2\sin^2\theta_3$$

$$\therefore r^2 + s^2 = l_3^2 + 2l_3l_4\cos\theta_3 + l_4^2$$

$\ldots$ (As; $\sin^2\theta + \cos^2\theta = 1$)

$$\therefore r^2 + s^2 - l_3^2 - l_4^2 = 2l_3l_4\cos\theta_3$$

$$\therefore \cos\theta_3 = \frac{r^2 + s^2 - l_3^2 - l_4^2}{2l_3l_4}$$

$\therefore$ Let $\cos\theta_3 = D$

Hence we can say;

$$\theta_3 = a\tan2\left(\sqrt{1-D^2}, D\right)$$

Now consider ΔOAR where;

$\angle ROA = \theta_2 + \alpha$.

We get α from ΔOPA

$$\therefore \quad \alpha = \text{atan2}\left(l_4 \sin\theta_3, (l_3 + l_4\cos\theta_3)\right)$$

Now in Δ OAR we can say,

$$\tan(\theta_2 + \alpha) = \frac{s}{r}$$

$$\therefore \quad \theta_2 + \alpha = \text{atan2}(s, r)$$

$$\therefore \quad \theta_2 = \text{atan}(s, r) - \text{atan2}\left(l_4 \sin\theta_3, (l_3 + l_4\cos\theta_3)\right)$$

Hence, we find the equations for θ_1, θ_2 and θ_3.

Now we calculate for θ_4 and θ_5. For which we take rotation matrix from joint 1 to joint 3

i.e $\quad R_3^0 = R_1^0 R_2^1 R_3^2$

Where we put values obtained from θ_1, θ_2 and θ_3

Before that we compare the matrix R_{65}^3 such as;

$$R_5^3 = (R_3^0)^{-1} R_{in}$$

Rin is the input rotation matrix

Now, from which we get the following;

$$\theta_5 = \text{atan2}\left(R_6^3(3,1), R_6^3(3,2)\right)$$

$$\theta_4 = \text{atan2}\left(R_6^3(2,3), R_6^3(1,3)\right)$$

5 Testing and Validation

This section is divided into 5 sections. Where the first three parts are the validation of the code and last two are the simulation.

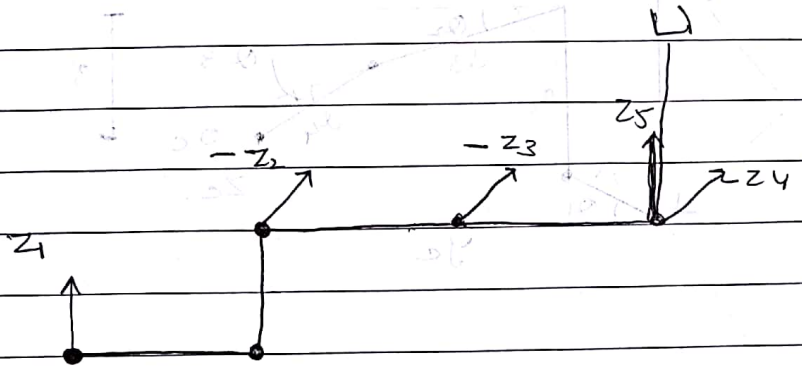
5.0.1 Validation of equations

In the first section we validate the formation of the forward and inverse kinematics equations. This is done by feeding the forward kinematics output, that is the position and the orientation of the end effector to the input of the inverse kinematics. And thus we shoud get the same joint angles. If so then the formulation of the forward and inverse kinematics are verified.
Following pages has the code that shows the above validation.

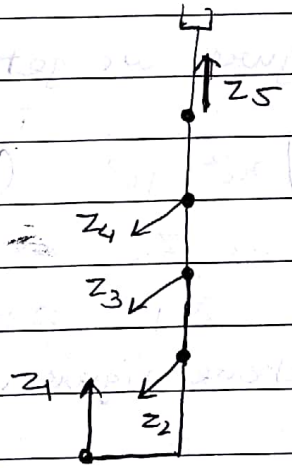
__/__/____

Correction Factor Required for Simulation

Our Model;



V-Rep KUKA YouBot Model;



Hence as there is a different in frame structures; we have to apply some correction factors for the simulation.

Hence, to reach the topmost point as shown in fig2.

We have to rotate $-z_2$ by -90°

i.e z_2 by $+90^\circ$

And rotate $-z_4$ by $+90$

i.e z_4 by -90°

```
clc;
clear;
%length param
l1 = 0.033;
l2 = 0.1012;
l3 = 0.155;
l4 = 0.1348;
l5 = 0.1087;
l6 = 0.085;
%input angles for forward Kinematics
%Test 1
q1 = deg2rad(60);
q2 = deg2rad(30);
q3 = deg2rad(-45);
q4 = deg2rad(45);
q5 = deg2rad(0);
%Test 2
%Test 3
inputAngles = [radtodeg(q1);radtodeg(q2);radtodeg(q3);radtodeg(q4);radtodeg(q5)];
disp("inputAngles = ")
```

```
inputAngles =
```

```
disp(inputAngles)
```

```
   60.0000
   30.0000
  -45.0000
   45.0000
         0
```

```
n = deg2rad(90);
n1 = deg2rad(-90);
T1 = [cos(n+q1) -sin(n+q1)*cos(n1) sin(n+q1)*sin(n1) l1*cos(n+q1);...
    sin(n+q1) cos(n+q1)*cos(n1) -cos(n+q1)*sin(n1) l1*sin(n+q1);...
    0 sin(n1) cos(n1) l2;...
    0 0 0 1];
n1 = 0;
T2 = [cos(q2) -sin(q2)*cos(n1) sin(q2)*sin(n1) l3*cos(q2);...
    sin(q2) cos(q2)*cos(n1) -cos(q2)*sin(n1) l3*sin(q2);...
    0 sin(n1) cos(n1) 0;...
    0 0 0 1];
T3 = [cos(q3) -sin(q3)*cos(n1) sin(q3)*sin(n1) l4*cos(q3);...
    sin(q3) cos(q3)*cos(n1) -cos(q3)*sin(n1) l4*sin(q3);...
    0 sin(n1) cos(n1) 0;...
    0 0 0 1];
n = deg2rad(-90);
n1 = deg2rad(90);
T4 = [cos(q4) -sin(q4)*cos(n1) sin(q4)*sin(n1) 0*cos(q4);...
    sin(q4) cos(q4)*cos(n1) -cos(q4)*sin(n1) 0*sin(q4);...
    0 sin(n1) cos(n1) 0;...
    0 0 0 1];
n =0;
T5 = [cos(q5) -sin(q5)*cos(n) sin(q5)*sin(n) 0*cos(q5);...
    sin(q5) cos(q5)*cos(n) -cos(q5)*sin(n) 0*sin(q5);...
    0 sin(n) cos(n) l5+l6;...
    0 0 0 1];

Tt = T1*T2*T3*T4*T5;


Rf = Tt(1:3,1:3);
```

```
Rf = 3×3
   -0.7500   -0.5000   -0.4330
    0.4330   -0.8660    0.2500
   -0.5000    0.0000    0.8660
```

```
Of = Tt(1:3,4);
```

```
Of = 3×1
   -0.3415
    0.1971
    0.2263
```

```
q1 =  atan2(Tt(2,4),Tt(1,4));
xc1 = Tt(1,4);
yc1 = Tt(2,4);
zc1 = Tt(3,4);



%This is inverse Kinematics
% We input the rotation matrix Rf and orientation from the forward
% kinematics to validate the result
Rin = Rf;
ox = Of(1,1);
oy = Of(2,1);
oz = Of(3,1);

xc = ox - (l5+l6)*Rin(1,3);
yc = oy - (l5+l6)*Rin(2,3);
zc = oz - (l5+l6)*Rin(3,3);
r = sqrt(power(xc,2)+power(yc,2)) - l1;
%disp(r);
s =    - zc + l2;
%disp(s);
D = (power(s,2) + power(r,2) - power(l3,2) - power(l4,2))/(2*l3*l4);
%disp(D);
q1 =  atan2(-xc,yc);
if(power(D,2) > 1)
  q3 = 0;
  q2 = 0;
else
  if (q3 < 0)
  a3 = atan2(-sqrt(1-power(D,2)),D);
  a2 = atan2(r,s) - atan2((l4*sin(q3)),(l3+l4*cos(q3))) ;
  else
  a3 = atan2(sqrt(1-power(D,2)),D);
  a2 = atan2(r,s) - atan2((l4*sin(a3)),(l3+l4*cos(a3))) ;
  end
end
a1=q1;
a2=q2;
a3=q3;
%Substitute the values of new q1,q2,q3 to get R03
n = deg2rad(90);
TN1 = [cos(n+a1) -sin(n+a1)*cos(-n) sin(n+a1)*sin(-n) l1*cos(n+a1);...
    sin(n+a1) cos(n+a1)*cos(-n) -cos(n+a1)*sin(-n) l1*sin(n+a1);...
    0 sin(-n) cos(-n) l2;...
    0 0 0 1];
n = deg2rad(-90);
TN2 = [cos(n+a2) -sin(n+a2)*cos(0) sin(n+a2)*sin(0) l3*cos(n+a2);...
    sin(n+a2) cos(n+a2)*cos(0) -cos(n+a2)*sin(0) l3*sin(n+a2);...
    0 sin(0) cos(0) 0;...
    0 0 0 1];
n = deg2rad(0);
TN3 = [cos(n+a3) -sin(n+a3)*cos(0) sin(n+a3)*sin(0) l4*cos(n+a3);...
    sin(n+a3) cos(n+a3)*cos(0) -cos(n+a3)*sin(0) l4*sin(n+a3);...
    0 sin(0) cos(0) 0;...
    0 0 0 1];
T03 = TN1*TN2*TN3;
R03 = T03(1:3,1:3);
R03t = transpose(R03);
R36 = R03t*Rin;
a5 = atan2(R36(3,1),R36(3,2));
a4 =  atan2(R36(2,3),R36(1,3));
%Now we display the output from the inverse kinematics
outputAngles = [radtodeg(a1);radtodeg(a2);radtodeg(a3);radtodeg(a4);radtodeg(a5)];
disp("outputAngles = ")
```

```
 outputAngles =
```

```
disp(outputAngles)
```

```
   60.0000
   30.0000
  -45.0000
   45.0000
   -0.0000
```

As the frames selected by us are different then the frames given by the V-Rep software, so we apply a correction factor explained on the next page.

5.0.2 FK Validation of equations with simulation

In the second section, we validate the Forward kinematics with the simulation of the KUKA YouBot in the V-Rep software.
For the simulation we will specify the given set of values and for the same set of values with some correction factor fed to the simulation should generate the result which we expect. This is validated by comparing the position of the end effector from the simulation and the position that we get from the transfer function obtained by the forward kinematics.
For example, we are considering the top most position that the bot can reach.

```
clc;
clear;
%length param
l1 = 0.033;
l2 = 0.1012;
l3 = 0.155;
l4 = 0.1348;
l5 = 0.1087;
l6 = 0.085;
%input angles for forward Kinematics
q1 = deg2rad(0); % 0 0  -90
q2 = deg2rad(-90);% -90 -135 -45
q3 = deg2rad(0);% 0 30 30
q4 = deg2rad(90); % 90 90 45
q5 = deg2rad(0);
inputAngles = [rad2deg(q1);rad2deg(q2);rad2deg(q3);rad2deg(q4);rad2deg(q5)];
disp("inputAngles = ")
```

```
 inputAngles =
```

```
disp(inputAngles)
```

```
     0
   -90
     0
    90
     0
```

```
%Equation of sforward kinematics
n = deg2rad(90);
n1 = deg2rad(-90);
T1 = [cos(n+q1) -sin(n+q1)*cos(n1) sin(n+q1)*sin(n1) l1*cos(n+q1);...
    sin(n+q1) cos(n+q1)*cos(n1) -cos(n+q1)*sin(n1) l1*sin(n+q1);...
    0 sin(n1) cos(n1) l2;...
    0 0 0 1];
n1 = 0;
T2 = [cos(q2) -sin(q2)*cos(n1) sin(q2)*sin(n1) l3*cos(q2);...
    sin(q2) cos(q2)*cos(n1) -cos(q2)*sin(n1) l3*sin(q2);...
    0 sin(n1) cos(n1) 0;...
    0 0 0 1];
T3 = [cos(q3) -sin(q3)*cos(n1) sin(q3)*sin(n1) l4*cos(q3);...
    sin(q3) cos(q3)*cos(n1) -cos(q3)*sin(n1) l4*sin(q3);...
    0 sin(n1) cos(n1) 0;...
    0 0 0 1];
n = deg2rad(-90);
n1 = deg2rad(90);
T4 = [cos(q4) -sin(q4)*cos(n1) sin(q4)*sin(n1) 0*cos(q4);...
    sin(q4) cos(q4)*cos(n1) -cos(q4)*sin(n1) 0*sin(q4);...
    0 sin(n1) cos(n1) 0;...
    0 0 0 1];
n =0;
T5 = [cos(q5) -sin(q5)*cos(n) sin(q5)*sin(n) 0*cos(q5);...
    sin(q5) cos(q5)*cos(n) -cos(q5)*sin(n) 0*sin(q5);...
    0 sin(n) cos(n) l5+l6;...
    0 0 0 1];

Tt = T1*T2*T3*T4*T5;


Rf = Tt(1:3,1:3);
Of = Tt(1:3,4);
disp(Of)
```

```
    0.0000
    0.0330
    0.5847
```

```
q1 =  atan2(Tt(2,4),Tt(1,4));
xc1 = Tt(1,4);
yc1 = Tt(2,4);
zc1 = Tt(3,4);



%This is inverse Kinematics
```

```
% We input the rotation matrix Rf and orientation from the forward
% kinematics to validate the result
Rin = Rf;
ox = Of(1,1);
oy = Of(2,1);
oz = Of(3,1);


xc = ox - (l5+l6)*Rin(1,3);
yc = oy - (l5+l6)*Rin(2,3);
zc = oz - (l5+l6)*Rin(3,3);
r = sqrt(power(xc,2)+power(yc,2)) - l1;
%disp(r);
s =    - zc + l2 ;
%disp(s);
D = (power(s,2) + power(r,2) - power(l3,2) - power(l4,2))/(2*l3*l4);
%disp(D);
a1 =  atan2(-xc,yc);
if(power(D,2) > 1)
  a3 = 0;
  a2 = 0;
else
  if (q3<0)
  a3 = atan2(-sqrt(1-power(D,2)),D);
  a2 = atan2(s,r) - atan2((l4*sin(q3)),(l3+l4*cos(q3))) ;
  else
  a3 = atan2(sqrt(1-power(D,2)),D)
  a2 = atan2(s,r) - atan2((l4*sin(q3)),(l3+l4*cos(q3)))
  end
end
```

```
a3 = 1.4901e-08
a2 = -1.5708
```

```
%Substitute the values of new q1,q2,q3 to get R03
n = deg2rad(90);
TN1 = [cos(n+a1) -sin(n+a1)*cos(-n) sin(n+a1)*sin(-n) l1*cos(n+a1);...
    sin(n+a1) cos(n+a1)*cos(-n) -cos(n+a1)*sin(-n) l1*sin(n+a1);...
    0 sin(-n) cos(-n) l2;...
    0 0 0 1];
n = deg2rad(-90);
TN2 = [cos(n+a2) -sin(n+a2)*cos(0) sin(n+a2)*sin(0) l3*cos(n+a2);...
    sin(n+a2) cos(n+a2)*cos(0) -cos(n+a2)*sin(0) l3*sin(n+a2);...
    0 sin(0) cos(0) 0;...
    0 0 0 1];
n = deg2rad(0);
TN3 = [cos(n+a3) -sin(n+a3)*cos(0) sin(n+a3)*sin(0) l4*cos(n+a3);...
    sin(n+a3) cos(n+a3)*cos(0) -cos(n+a3)*sin(0) l4*sin(n+a3);...
    0 sin(0) cos(0) 0;...
    0 0 0 1];
T03 = TN1*TN2*TN3;
R03 = T03(1:3,1:3);
R03t = transpose(R03);
R36 = R03t*Rin;
a5 = atan2(R36(3,1),R36(3,2));
a4 =  atan2(R36(2,3),R36(1,3));
%Now we display the output from the inverse kinematics


vrep=remApi('remoteApi'); % using the prototype file (remoteApiProto.m)
```

Note: always make sure you use the corresponding remoteApi library
(i.e. 32bit Matlab will not work with 64bit remoteApi, and vice-versa)

```
vrep.simxFinish(-1); % just in case, close all opened connections
clientID=vrep.simxStart('127.0.0.1',19999,true,true,5000,5);
if (clientID > -1 )
   disp("Connected");
   for i = 0:5
     a=(['youBotArmJoint',num2str(i),'#0']);
     [returnCode,joint(i+1)]=vrep.simxGetObjectHandle(clientID,a,vrep.simx_opmode_blocking);
   end
  [returnCode,gripper1]=vrep.simxGetObjectHandle(clientID,'youBotGripperJoint1#0',vrep.simx_opmode_blocking);
  [returnCode,gripper2]=vrep.simxGetObjectHandle(clientID,'youBotGripperJoint2#0',vrep.simx_opmode_blocking);
  %[returnCode,position123] = vrep.simxGetJointPosition(clientID,gripper2,vrep.simx_opmode_blocking)
  %[returnCode]=vrep.simxSetJointTargetPosition(clientID,gripper1,position123*(-0.5),vrep.simx_opmode_blocking);
  i = 1;
  [returnCode]= vrep.simxSetJointTargetVelocity(clientID,gripper2,0.5,vrep.simx_opmode_blocking);
```

```
 [returnCode]=vrep.simxSetJointTargetVelocity(clientID,joint(1),0.5,vrep.simx_opmode_oneshot);
 [returnCode]=vrep.simxSetJointTargetPosition(clientID,joint(2),0,vrep.simx_opmode_oneshot);

 while (i < 20000)
  %for i = 0:5
    [returnCode]=vrep.simxSetJointPosition(clientID,joint(1),deg2rad(0),vrep.simx_opmode_oneshot);% 0 0 90
    [returnCode]=vrep.simxSetJointPosition(clientID,joint(2),deg2rad(0),vrep.simx_opmode_oneshot);%0 45 45
    [returnCode]=vrep.simxSetJointPosition(clientID,joint(3),deg2rad(0),vrep.simx_opmode_oneshot);% 0 -30 -30
    [returnCode]=vrep.simxSetJointPosition(clientID,joint(4),deg2rad(0),vrep.simx_opmode_oneshot); % 0  0 45
    [returnCode]=vrep.simxSetJointPosition(clientID,joint(5),deg2rad(0),vrep.simx_opmode_oneshot);
    if(i == 10000)
    [returnCode,eulerAngles]=vrep.simxGetObjectOrientation(clientID,gripper1,joint(1),vrep.simx_opmode_blocking);
    %disp(eulerAngles);
    [returnCode,position]=vrep.simxGetObjectPosition(clientID,gripper1,joint(1),vrep.simx_opmode_blocking);
    disp(position);
    end
    i=i+1;
 end

 vrep.simxFinish(-1);

end
```

```
Connected
  Column 1

   -0.0010

  Column 2

    0.0337

  Column 3

    0.5847
```

```
vrep.delete();
%the sign of the position will be different because of the difference in the selection of the frames
```

5.0.3 IK validation of the equations with simulation

For this simulation we will specify the orientation and the rotation matrix of the end effector frame and they are fed to the inverse kinematics equation. Thus the angles found are fed to the simulation and the desired position of the robotic arm is achieved.
Thus, this will validate the inverse kinematics with the simulation. For the simulation we use the consition when the value of q3 is 30 degrees.

```
clc;
clear;
%length param
l1 = 0.033;
l2 = 0.1012;
l3 = 0.155;
l4 = 0.1348;
l5 = 0.1087;
l6 = 0.085;
%This is inverse Kinematics
% We input the rotation matrix Rf and orientation from the forward
% kinematics to validate the resul
q3 = 3;
%Condition for q3 = 30

Rin  = [0 -1 0; 0.866 0 0.5;-0.5 0 0.866];
ox = 0;
oy = 0.4016;
oz = 0.2015;


%Condition for initial position
%{
Rin = [0 -1 -0;1 0 0; 0 -0 1];
ox = 0;
oy = 0.033;
oz = 0.5848;
%}
xc = ox - (l5+l6)*Rin(1,3);
yc = oy - (l5+l6)*Rin(2,3);
zc = oz - (l5+l6)*Rin(3,3);

r = sqrt(power(xc,2)+power(yc,2)) - l1;
%disp(r);
s =    - zc + l2;
%disp(s);
D = (power(s,2) + power(r,2) - power(l3,2) - power(l4,2))/(2*l3*l4);
%disp(D);
a1 =  atan2(-xc,yc);
disp(rad2deg(a1));
```

```
    0
```

```
if(power(D,2) > 1)
  a3 = 0;
  a2 = 0;
else
  if (q3<0)
  a3 = atan2(-sqrt(1-power(D,2)),D);
  a2 = atan2(s,r) - atan2((l4*sin(a3)),(l3+l4*cos(a3))) ;
  else
  a3 = atan2(sqrt(1-power(D,2)),D);
  a2 = atan2(s,r) - atan2((l4*sin(a3)),(l3+l4*cos(a3)));
  end
end
%Substitute the values of new q1,q2,q3 to get R03
n = deg2rad(90);
TN1 = [cos(n+a1) -sin(n+a1)*cos(-n) sin(n+a1)*sin(-n) l1*cos(n+a1);...
    sin(n+a1) cos(n+a1)*cos(-n) -cos(n+a1)*sin(-n) l1*sin(n+a1);...
    0 sin(-n) cos(-n) l2;...
    0 0 0 1];
n = deg2rad(-90);
TN2 = [cos(n+a2) -sin(n+a2)*cos(0) sin(n+a2)*sin(0) l3*cos(n+a2);...
    sin(n+a2) cos(n+a2)*cos(0) -cos(n+a2)*sin(0) l3*sin(n+a2);...
    0 sin(0) cos(0) 0;...
    0 0 0 1];
n = deg2rad(0);
TN3 = [cos(n+a3) -sin(n+a3)*cos(0) sin(n+a3)*sin(0) l4*cos(n+a3);...
    sin(n+a3) cos(n+a3)*cos(0) -cos(n+a3)*sin(0) l4*sin(n+a3);...
    0 sin(0) cos(0) 0;...
    0 0 0 1];
T03 = TN1*TN2*TN3;
R03 = T03(1:3,1:3);
R03t = transpose(R03);
R36 = R03t*Rin;
```

```
a5 = atan2(R36(3,1),R36(3,2));
a4 =  atan2(R36(2,3),R36(1,3));
%Now we display the output from the inverse kinematics
vrep=remApi('remoteApi'); % using the prototype file (remoteApiProto.m)
```

Note: always make sure you use the corresponding remoteApi library
(i.e. 32bit Matlab will not work with 64bit remoteApi, and vice-versa)

```
vrep.simxFinish(-1); % just in case, close all opened connections
clientID=vrep.simxStart('127.0.0.1',19999,true,true,5000,5);
if (clientID > -1 )
  disp("Connected");
  for i = 0:5
    a=(['youBotArmJoint',num2str(i),'#0']);
    [returnCode,joint(i+1)]=vrep.simxGetObjectHandle(clientID,a,vrep.simx_opmode_blocking);
  end
 [returnCode,gripper1]=vrep.simxGetObjectHandle(clientID,'youBotGripperJoint1#0',vrep.simx_opmode_blocking);
 [returnCode,gripper2]=vrep.simxGetObjectHandle(clientID,'youBotGripperJoint2#0',vrep.simx_opmode_blocking);
 %[returnCode,position123] = vrep.simxGetJointPosition(clientID,gripper2,vrep.simx_opmode_blocking)
 %[returnCode]=vrep.simxSetJointTargetPosition(clientID,gripper1,position123*(-0.5),vrep.simx_opmode_blocking);
 i = 1;
 [returnCode]= vrep.simxSetJointTargetVelocity(clientID,gripper2,0.5,vrep.simx_opmode_blocking);
 [returnCode]=vrep.simxSetJointTargetVelocity(clientID,joint(1),0.5,vrep.simx_opmode_oneshot);
 [returnCode]=vrep.simxSetJointTargetPosition(clientID,joint(2),0,vrep.simx_opmode_oneshot);

 while (i < 20000)
  %for i = 0:5
    [returnCode]=vrep.simxSetJointPosition(clientID,joint(1),a1,vrep.simx_opmode_oneshot);% 0 0 90
    [returnCode]=vrep.simxSetJointPosition(clientID,joint(2),a2,vrep.simx_opmode_oneshot);%0 45 45
    [returnCode]=vrep.simxSetJointPosition(clientID,joint(3),a3,vrep.simx_opmode_oneshot);% 0 -30 -30
    [returnCode]=vrep.simxSetJointPosition(clientID,joint(4),a4,vrep.simx_opmode_oneshot); % 0  0 45
    [returnCode]=vrep.simxSetJointPosition(clientID,joint(5),a5,vrep.simx_opmode_oneshot);
    if(i == 10000)
    [returnCode,eulerAngles]=vrep.simxGetObjectOrientation(clientID,gripper1,joint(1),vrep.simx_opmode_blocking);
    %disp(eulerAngles);
    [returnCode,position]=vrep.simxGetObjectPosition(clientID,gripper1,joint(1),vrep.simx_opmode_blocking);
    %disp(position);
    end
    i=i+1;
 end

 vrep.simxFinish(-1);

end
```

Connected

```
vrep.delete();
```

For the simulation of the robot, the project is included with the V-Rep scenes and the matlab files. For running the simulations, follow the instructions given in the readme.
For accessing the code, go to the following GitHub repository
https://github.com/KrishnaBhatu/Robot-Librarian

References

1) https://www.academia.edu/23871359/Derivation$_o f_K inematic_E quations_f or_t he_K UKA_y ouBot_a nd_i mpleme$
2)$https://www.wikipedia.org$
3)$RobotModelingandControl, BookbyMarkW.Spong$