writeup

After many epic fail, this time it is so close to hit the target!!

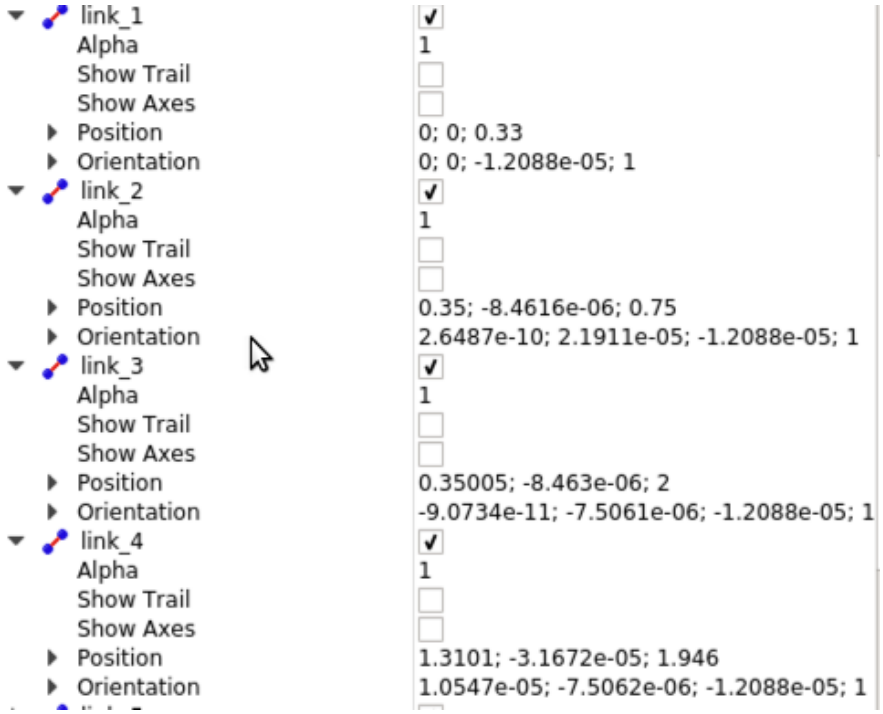
https://vimeo.com/259490068

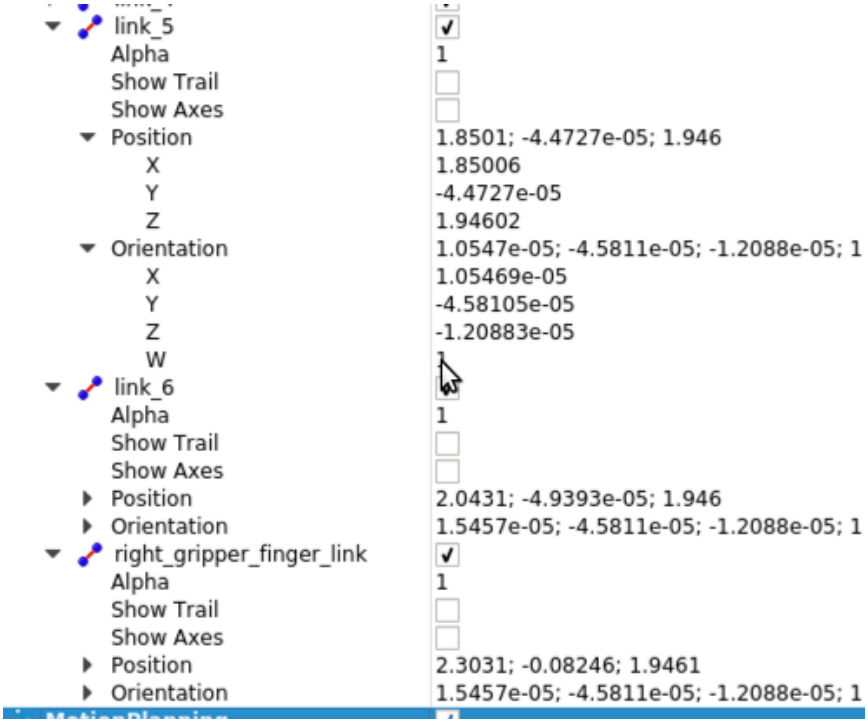
Kinematic Analysis

1. Run the forward_kinematics demo and evaluate the kr210.urdf.xacro file to perform kinematic analysis of Kuka KR210 robot and derive its DH parameters.

You can find a screenshot when we run the robotic arm in the demo mode. The numerical values for the parameters is in the kr210.urdf.xarco file. It contains a lot of section. Movable joints are defined in the joints section in urdf file.

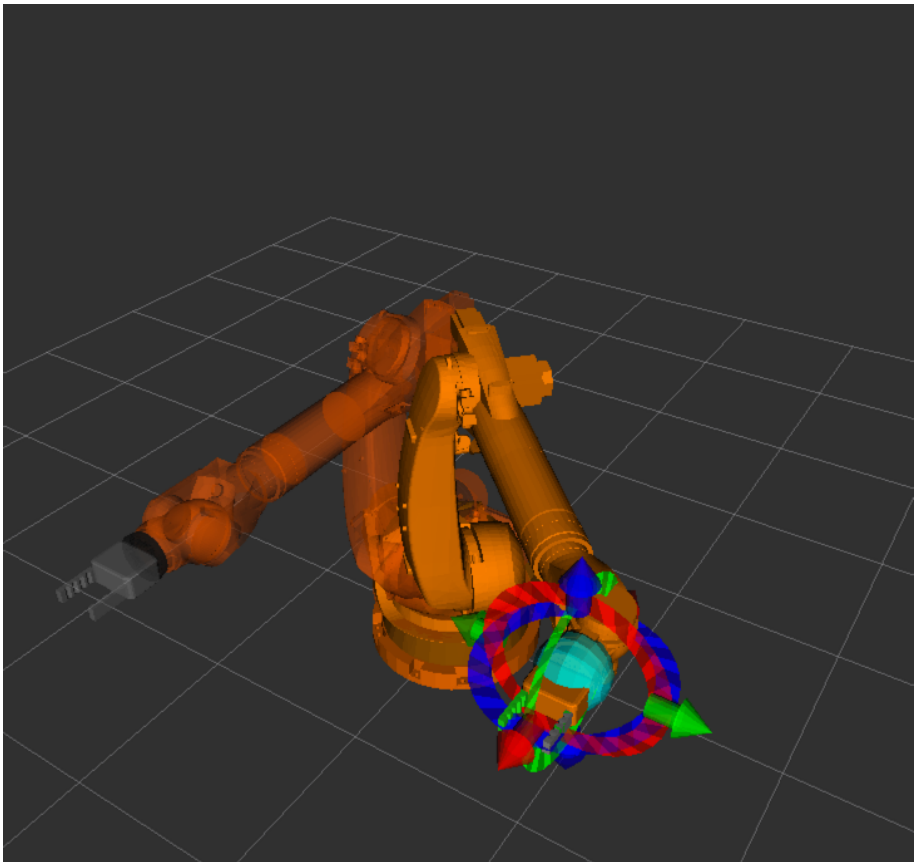
You can find the parameter values for each link below.





You can interact with robotic arm when you run the following command.

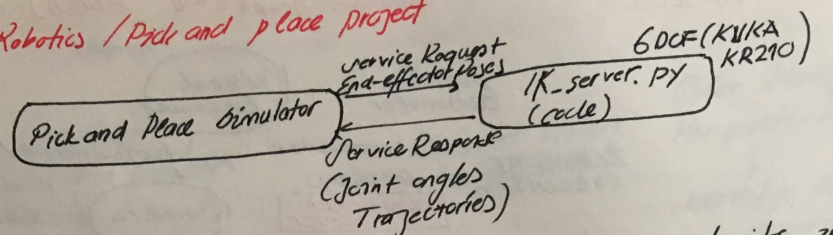
```
roslaunch kr210_claw_moveit demo.launch
```



2. Using the DH parameter table you derived earlier, create individual transformation matrices about each joint. In addition, also generate a generalized homogeneous transform between base_link and gripper_link using only end-effector(gripper) pose.

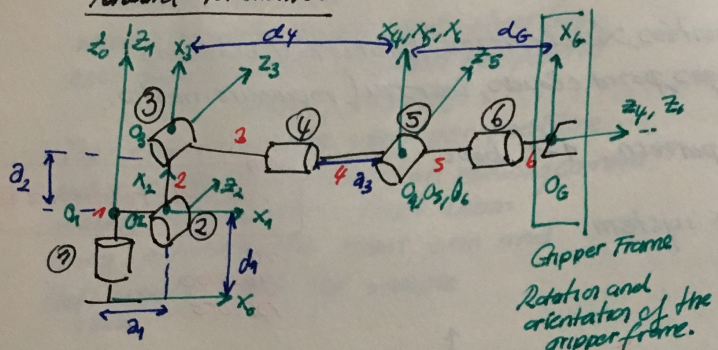
You can find the required explanation in my notes below.

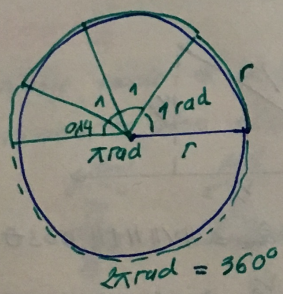
Robotics / Pick and place project



sketch robotic arm to its zero configuration

Forward Kinematics





$2\pi r = 360°$

$1\ rad = \frac{180°}{\pi}$

$5\ rad = 5 * \frac{180°}{\pi}$

- Degrees of freedom
- Joint angle
- Reference frame
- Serial manipulators
- configuration space
- Joint space

kuka_arm/urdf/kr210.urdf.xacro →

Euler angles → Kompozit bir hareket roll, pitch, yaw gibi üç hareketle ifade edilebilinir.

$$ {}^{A}_{B}R_{XYZ} = R_z(\alpha)\, R_y(\beta)\, R_x(\gamma) $$

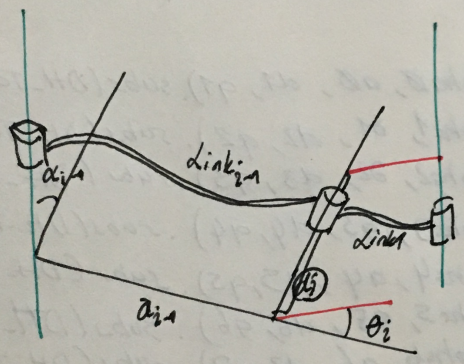
Attaching a reference frame to each link
Homogeneous transform from the fixed base (link 0) to link 1 ... link (n-1) to link n

$$ {}^{0}_{N}T = {}^{0}_{1}T\,{}^{1}_{2}T\,{}^{2}_{3}T \dots {}^{N-1}_{N}T $$

each transform requires six independent parameters
frame i relative to i-1
three for position
three for orientation

Denavit-Hartenberg (DH)
only requires 4 parameters to describe position and orientation of neighbouring reference frames.

→ There are different conventions
We use the one in John J. Craig's book



We can see all four parameters and how to calculate them.

Now, lets apply this and calculate DH table for Kuka KR210

	i	α_{i-1}	a_{i-1}	d_i	θ_i
T^0_1	0	0	0	0,75	q_1
T^1_2	1	$-\pi/2$	0,35	0	$-\pi/2 + q_2$
T^2_3	2	0	1,25	0	q_3
T^3_4	3	$-\pi/2$	0,054	1,5	q_4
T^4_5	4	$\pi/2$	0	0	q_5
T^5_6	5	$-\pi/2$	0	0	q_6
T^6_7	6	0	0	0,303	0

explanation: how to calculate this matrix

TF DH Transformation matrix

$$
{}^{i-1}_{i}T \rightarrow \begin{bmatrix} \cos(q) & -\sin(q) & 0 & a \\ \sin(q)*\cos(\alpha) & \cos(q)*\cos(\alpha) & -\sin(\alpha) & -\sin(\alpha)*d \\ \sin(q)*\sin(\alpha) & \cos(q)*\sin(\alpha) & \cos(\alpha) & \cos(\alpha)*d \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^{i-1}_{i}T = R_x(\alpha_{i-1})\, D_x(a_{i-1})\, R_z(\theta_i)\, D_z(d_i)
$$

constructed as a sequence of four basic transformation two rotation and two transformation as shown above

You can use this transformation matrix for each joint. just replace the α, a, d and q values.

```
T0_1 = TF_Matrix(alpha0, a0, d1, q1).subs(DH_Table)
T1_2 = TF_Matrix(alpha1, a1, d2, q2).subs(DH_Table)
T2_3 = TF_Matrix(alpha2, a2, d3, q3).subs(DH_Table)
T3_4 = TF_Matrix(alpha3, a3, d4, q4).subs(DH_Table)
T4_5 = TF_Matrix(alpha4, a4, d5, q5).subs(DH_Table)
T5_6 = TF_Matrix(alpha5, a5, d6, q6).subs(DH_Table)
T6_EE = TF_Matrix(alpha6, a6, d7, q7).subs(DH_Table)
```

And transform matrix from base to gripper.

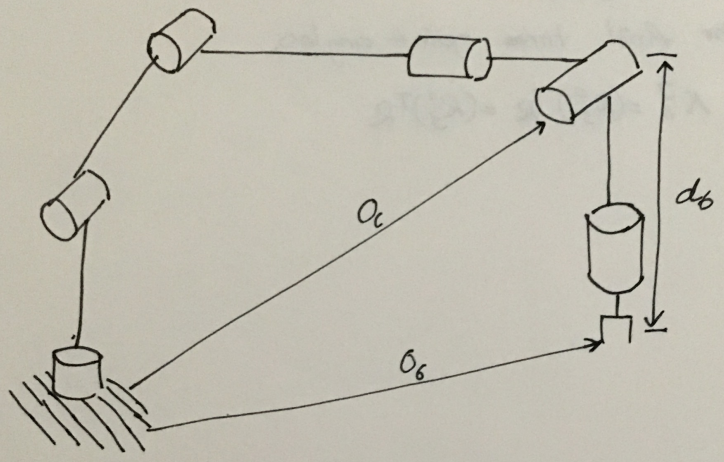
```
T0_EE = T0_1 * T1_2 * T2_3 * T3_4 * T4_5 * T5_6 * T6_EE
```

3. Decouple Inverse Kinematics problem into Inverse Position Kinematics and inverse Orientation Kinematics; doing so derive the equations to calculate all individual joint angles.

WC → Wrist Center
EE → End Effector

Decoupling Inverse Kinematics into Inverse Position Kinematics and Inverse Rotation Kinematics.

The robotic arm has 6 DoF serial manipulator.
First three joints could be use to control the position of the wrist center
The last three joints would orient the end effector as needed



Two sets of equations representing rotational and positional equations

$$R_6^0(q_1, \ldots, q_6) = R$$
$$O_6^0(q_1, \ldots, q_6) = O$$

The origin of the tool frame (O → desired Coordinates)

$$O = O_c^0 + d_6 R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

R express the direction of Z_6

$$O_c^0 = O - d_6 R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Leave O_c^0 alone at left side of the eq

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \begin{bmatrix} O_x - d_6 r_{13} \\ O_y - d_6 r_{23} \\ O_z - d_6 r_{33} \end{bmatrix}$$

$O = [O_x \; O_y \; O_z]^T$, $O_c^0 = [x_c \; y_c \; z_c]^T$

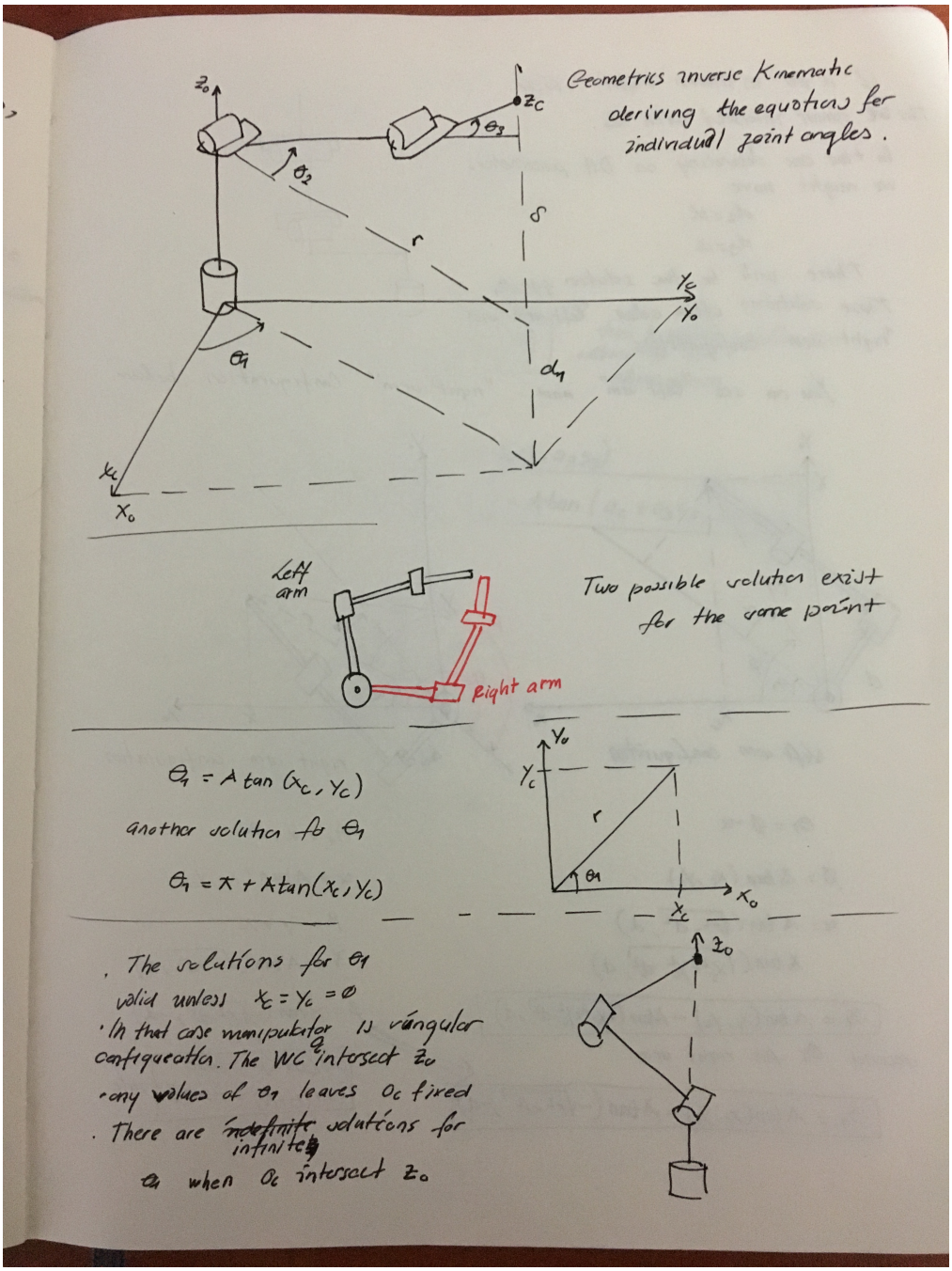
since $[x_c \; y_c \; z_c]^T$ are determined from the first three joints angles, our forward kinematics expression now allow us to solve for the first three joint angles decoupled from the final three

$$R = R_3^0 R_6^3$$

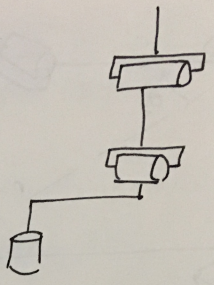
To solve the final three joint angles.

$$R_6^3 = (R_3^0)^{-1} R = (R_3^0)^T R$$

Deriving geometric inverse calculations for individual joint angles.



If $d \neq 0$ as shown right side



The wc cannot intersect with Z_0.

In that case depending on DH parameters we might have

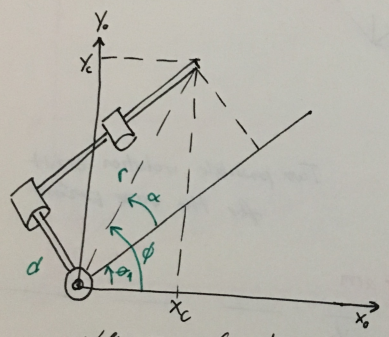
$d_2 = d$

$d_3 = d$

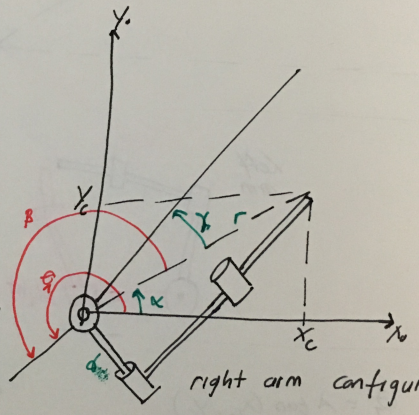
There will be two solution for θ_1

These solutions often called "left arm" and "right arm" configurations.

You can see "left arm" and "right arm" configuration below.



left arm configuration



right arm configuration.

$\theta_1 = \phi - \alpha$

$\phi = A\tan(x_c, y_c)$

$\alpha = A\tan\left(\sqrt{r^2 - d^2}, d\right)$

$A\tan\left(\sqrt{x_c^2 + y_c^2 - d^2}, d\right)$

$$\theta_1 = A\tan(x_c, y_c) - A\tan\left(\sqrt{x_c^2 + y_c^2 - d^2}, d\right)$$

second θ_1 for right arm.

$$\theta_1 = A\tan(x_c, y_c) - A\tan\left(-\sqrt{r^2 - d^2}, -d\right)$$

$\theta_1 = \alpha + \beta$

$\alpha = A\tan(x_c, y_c)$

$\beta = \gamma + \pi$

$\gamma = A\tan\left(\sqrt{r^2 - d^2}, d\right)$

$\beta = A\tan\left(-\sqrt{r^2 - d^2}, -d\right)$

$\cos(\theta + \pi) = -\cos\theta$

$\sin(\theta + \pi) = -\sin\theta$ rule

$$\cos\theta_3 = \frac{r^2 + s^2 - a_2^2 - a_3^2}{2a_2 a_3}$$

$$= \frac{x_c^2 + y_c^2 - d^2 + z_c^2 - a_2^2 - a_3^2}{2a_2 a_3} := D$$

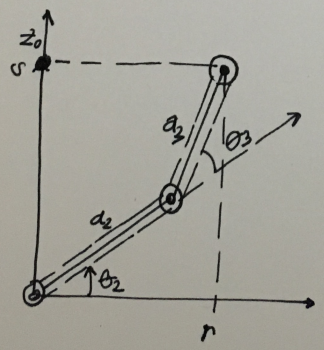
since $r^2 = x_c^2 + y_c^2 - d^2$ and $s = z_c$

$$\theta_3 = A\tan\left(D, \pm\sqrt{1-D^2}\right)$$

two possible solution for elbow up and elbow down

$$\theta_2 = A\tan(r,s) - A\tan(a_2 + a_3 c_3, a_3 s_3)$$

$$\theta_2 = A\tan\left(\sqrt{x_c^2 + y_c^2 - d^2}, z_c\right) - A\tan\left(a_2 + a_3 c_3, a_3 s_3\right)$$



inverse orientation

$$R_6^0(q_1,\ldots,q_6) = R$$

$$O_c^0(q_1,\ldots,q_6) = O$$

$$O = O_c^0 + d_6 R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$O_c^0 = O - d_6 R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Components of the end-effector position O are denoted O_x, O_y, O_z
components of the wrist-center O_c^0 are denoted x_c, y_c, z_c

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \begin{bmatrix} O_x - d_6 r_{13} \\ O_y - d_6 r_{23} \\ O_z - d_6 r_{33} \end{bmatrix}$$

$$R = R_3^0 R_6^3$$

$$R_6^3 = (R_3^0)^{-1} R = (R_3^0)^T R$$

The final three joint angles can be found as a set of Euler Angles corresponding to R_6^3

$$O = \begin{bmatrix} O_x \\ O_y \\ O_z \end{bmatrix} \qquad R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

$$x_c = O_x - d_6 r_{13}$$

$$y_c = O_y - d_6 r_{23}$$

$$z_c = O_z - d_6 r_{33}$$

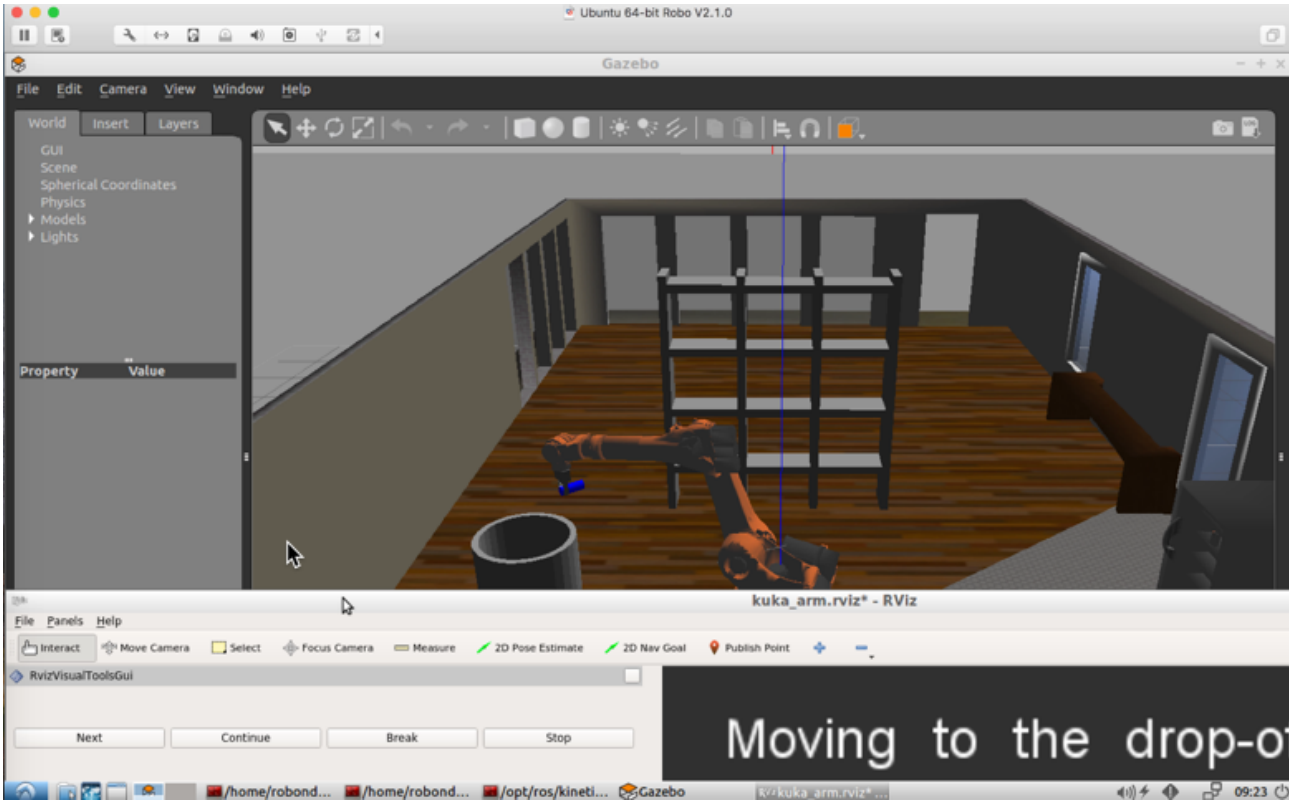
$$\theta_4 = \text{Atan}(c_1 c_{23} r_{13} + s_1 c_{23} r_{23} + s_{23} r_{33}, -c_1 s_{23} r_{13} - s_1 s_{23} r_{23} + c_{23} r_{33})$$

$$\theta_5 = \text{Atan}(s_1 r_{13} - c_1 r_{23}, \pm\sqrt{1 - (s_1 r_{13} - c_1 r_{23})^2})$$

$$\theta_6 = \text{Atan}(-s_1 r_{11} + c_1 r_{21}, s_1 r_{12} - c_1 r_{22})$$

Fill in the `IK_server.py` file with properly commented python code for calculating Inverse Kinematics based on previously performed Kinematic Analysis. Your code must guide the robot to successfully complete 8/10 pick and place cycles. Briefly discuss the code you implemented and your results.

You can check source code for the detailed explanation. I have added detailed explantion to the source code.



Also, I have uploaded a video which robotic arm successfully carry object to the target location. You can find it in the following link.

https://vimeo.com/259565752

optimizations:

There might be a lot of pose in the trajectory. Ve evaluate all of them in a for loop, but forward kinematics could be build prior to for loop and only evaluation work inside the loop. Also, You can create a class structure and initialize it when IK_server is starting.

Another optimization technique, you can save transformation matrices and rotation matrices as pickle file and load them. Also, some people use numpy to make everything to get a better performance.

references

http://d.umn.edu/~rlindek1/ME4135_11/ch6.pdf

https://thydzik.com/academic/robotics-315/chap4.pdf

http://web.eecs.umich.edu/~ocj/courses/autorob/autorob_10_ik_closedform.pdf

http://www.uio.no/studier/emner/matnat/ifi/INF3480/v14/undervisningsmatriale/lec08_inv_jaco.pdf

http://www4.cs.umanitoba.ca/~jacky/Robotics/Papers/spong_kinematics.pdf

http://www-scf.usc.edu/~csci545/slides/Lec6_Inverse_VelocityKinematicsI_short.pdf

https://pdfs.semanticscholar.org/c8da/a491968cc3e03a389078d7490d9173a87ffc.pdf

https://pdfs.semanticscholar.org/9c9c/9e9ab184183c912ca9f755b8f14a2f682ab3.pdf