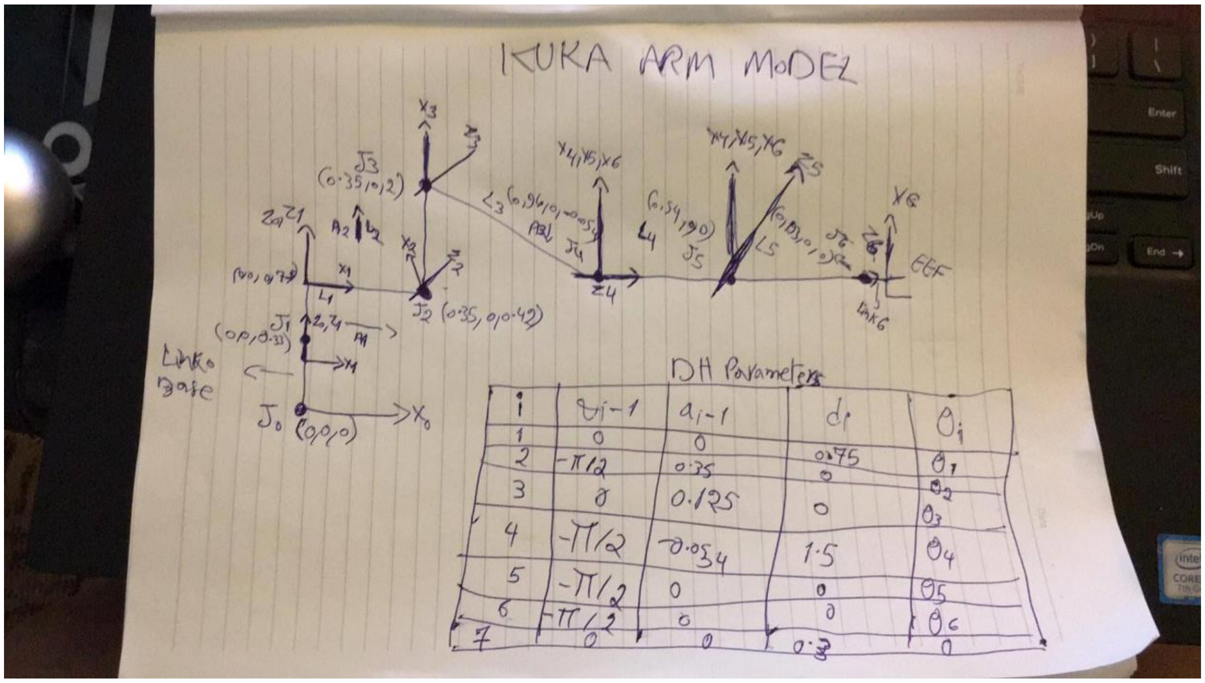
Kinematic Analysis

KUKA ARM MODEL



i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0.75	θ_1
2	$-\pi/2$	0.35	0	θ_2
3	0	0.125	0	θ_3
4	$-\pi/2$	-0.054	1.5	θ_4
5	$-\pi/2$	0	0	θ_5
6	$-\pi/2$	0	0	θ_6
7	0	0	0.3	0

DH Parameters Table

I	Alpha i-1	Ai-1	di-1	0_i
1	**0**	**0**	**0.75**	**0_1**
2	**-Pi/2**	**0.35**	**0**	**0_2**
3	**0**	**0.125**	**0**	**0_3**
4	**-Pi/2**	**-0.054**	**1.5**	**0_4**
5	**Pi/2**	**0**	**0**	**0_5**
6	**-Pi/2**	**0**	**0**	**0_6**
7	**0**	**0**	**0.303**	**0_7**

2-A) Individual transformation matrices about each joint

TF_0_1
Matrix([
[cos(q1), -sin(q1), 0, 0],
[sin(q1), cos(q1), 0, 0],
[0, 0, 1, 0.75],
[0, 0, 0, 1]])

TF_1_2
Matrix([
[sin(q2), cos(q2), 0, 0.35],
[0, 0, 1, 0],
[cos(q2), -sin(q2), 0, 0],

[0, 0, 0, 1]])

TF_2_3
Matrix([
[cos(q3), -sin(q3), 0, 1.25],
[sin(q3), cos(q3), 0, 0],
[0, 0, 1, 0],
[0, 0, 0, 1]])

TF_3_4
Matrix([
[cos(q4), -sin(q4), 0, -0.054],
[0, 0, 1, 1.5],
[-sin(q4), -cos(q4), 0, 0],
[0, 0, 0, 1]])

TF_4_5
Matrix([
[cos(q5), -sin(q5), 0, 0],
[0, 0, -1, 0],
[sin(q5), cos(q5), 0, 0],
[0, 0, 0, 1]])

TF_5_6
Matrix([
[cos(q6), -sin(q6), 0, 0],
[0, 0, 1, 0],
[-sin(q6), -cos(q6), 0, 0],
[0, 0, 0, 1]])

TF_6_7
Matrix([
[1, 0, 0, 0],
[0, 1, 0, 0],
[0, 0, 1, 0.303],
[0, 0, 0, 1]])

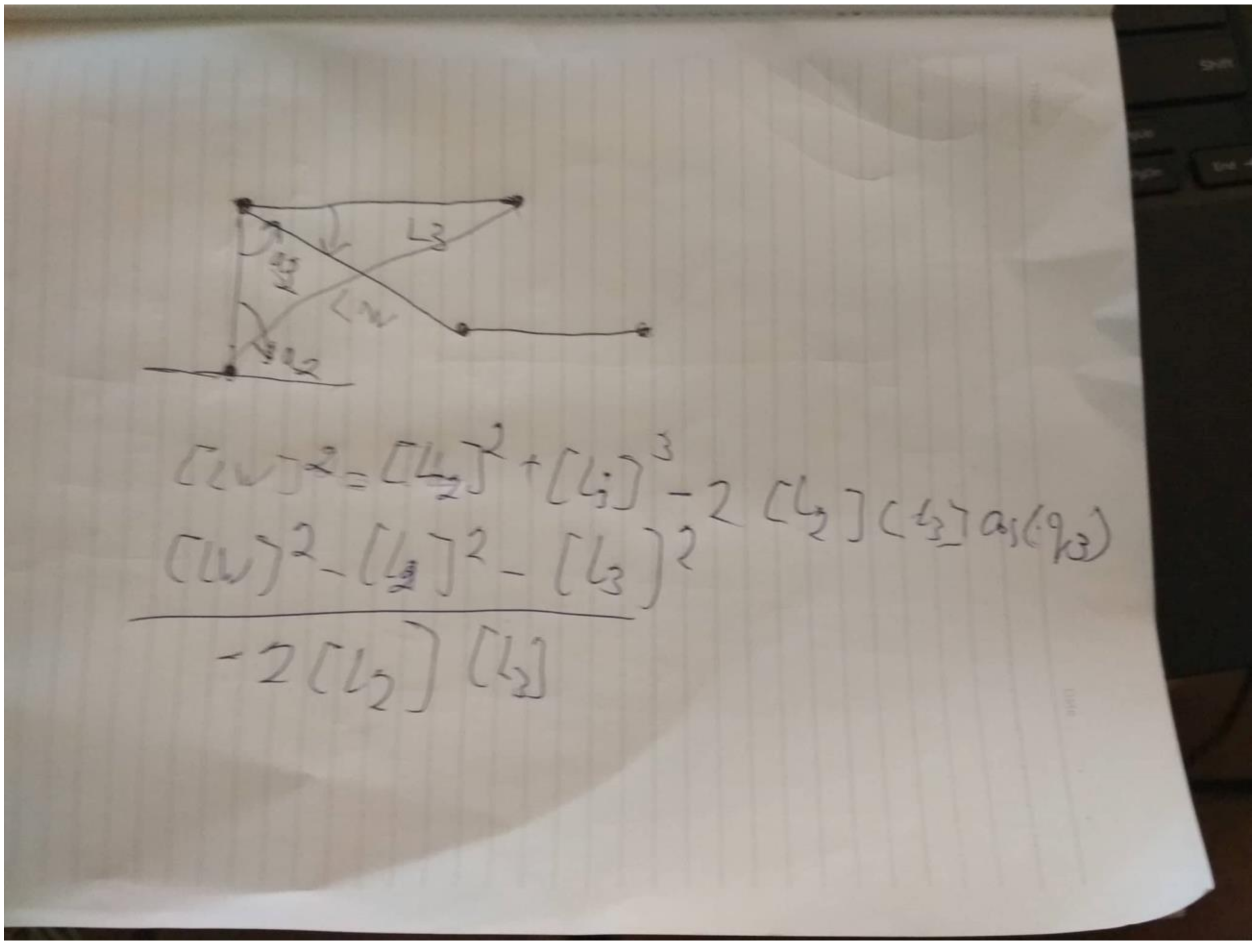
2-B) Homogeneous transform between base_link (T0) and gripper_link (T7) using only the Orientation of gripper-link.

px,py,pz refers to the gripper target location.

TF_0_7

Matrix([
[cos(pitch)*cos(yaw), -sin(yaw)*cos(pitch), sin(pitch), px],
[sin(pitch)*sin(roll)*cos(yaw) + sin(yaw)*cos(roll), -sin(pitch)*sin(roll)*sin(yaw) + cos(roll)*cos(yaw), -sin(roll)*cos(pitch), py],
[-sin(pitch)*cos(roll)*cos(yaw) + sin(roll)*sin(yaw), sin(pitch)*sin(yaw)*cos(roll) + sin(roll)*cos(yaw), cos(pitch)*cos(roll), pz],
[0, 0, 0, 1])

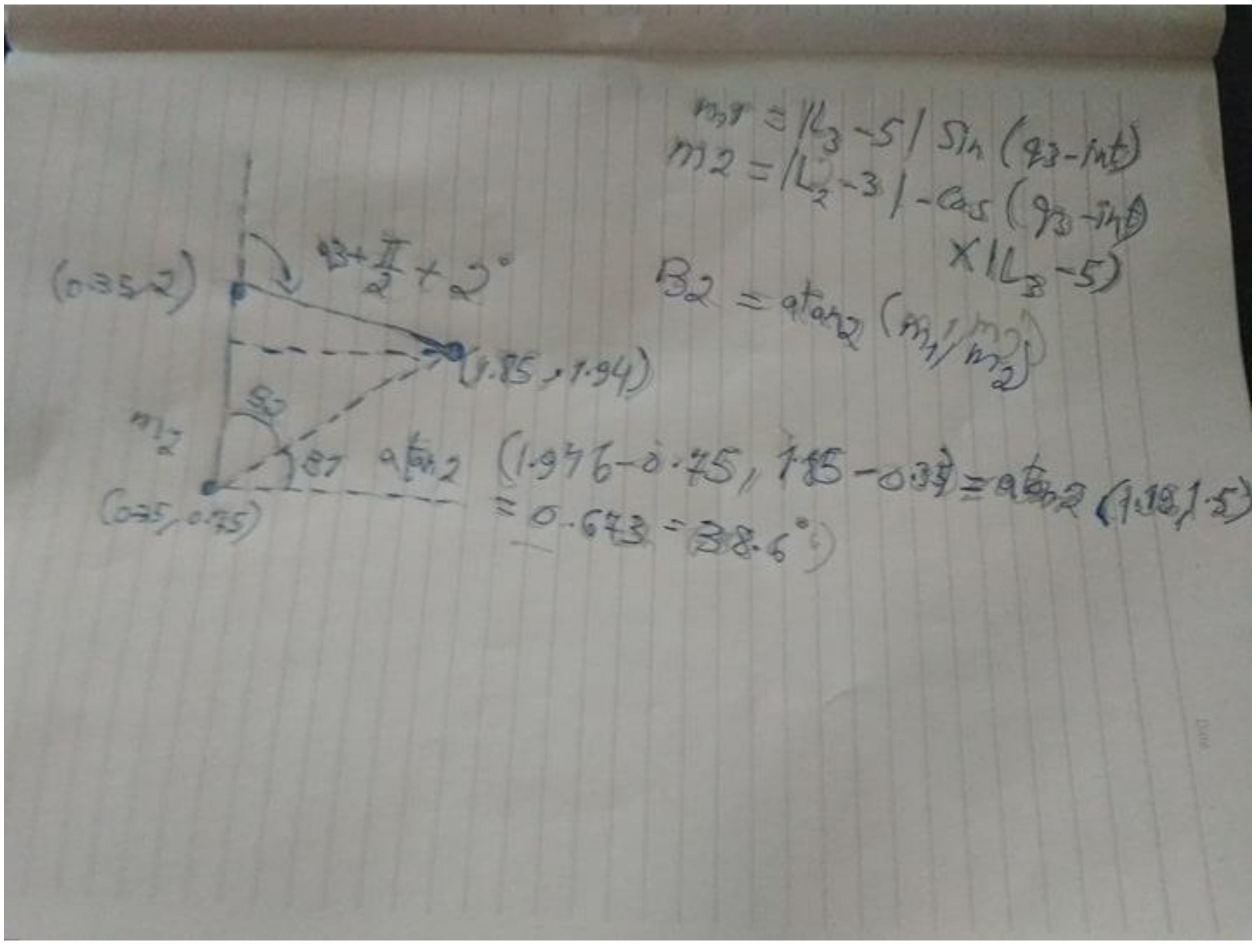
The first part of the code (~lines 33-132) implements the DH table and creates the individual transform matrices around each joint, as discussed earlier in this report. Also as discussed in the lesson I applied intrinsic (body-fixed) rotations to gripper frame (7) to ensure orientations are aligned. First I rotated around z-axis by 180deg and then around y-axis by -90deg.Note that in the skeleton code provided this was done within the for loop that iterates over all requested poses, however I found that doing this caused a major slowdown in the code execution speed. Since the symbolic manipulation is the same each time I reasoned this didn’t need to be inside the loop, so I moved it before and improved the performance.
Next comes the main loop where the inverse kinematics are calculated for each of the requested poses. The first step is to calculate the wrist centre (WC) location (~lines 153-189). The first sub-step is to construct R0_6 based on the target roll, pitch and yaw angles requested of the end effector. This is done by first rotating an amount *roll* around the x-axis, then an amount *pitch* around the y-axis and finally an amount *yaw* around the z-axis, before concatenating the transforms. Then it is just required to translate a distance equivalent to the length of the end effector (0.303m) along the z6 axis, as explained in the lesson notes.
The next step is to calculate theta1 (line 192). This is simple since we just have to project the Z-coordinate of the WC into the ground plane, as explained in lesson 2-19.
Next (~lines 195-216) I calculate the location of joint 2 (since it depends on theta1). I also calculate the length of various segments which will be used in later calculations (NB: All coordinates with the suffix *__0 are joint coordinates at the time when q1=q2=q3=q4=q5=q6=0 which is a useful situation to calculate joint lengths).
The next step is to calculate theta 3 (~lines 217 - 229).I used my testing scripts extensively to test a variety of angles and see if I was generating the right results. Ultimately the correct approach was to calculate a vector between joint 2 and WC and then use the cosine rule to solve for theta3, as shown in Figure 2. It should be noted that there are in fact two possible solutions for theta3 (theta3 on line 224 and theta3_2 on line 225). I picked the first solution (positive square root) since the other one (negative square root) would lead to a theta3 exceeding the specified joint limits. I confirmed via the forward kinematics that this led to the correct final position of the end effector.



$$[L_w]^2 = [L_2]^2 + [L_3]^3 - 2[L_2][L_3]\cos(q_3)$$

$$\frac{[L_w]^2 - [L_2]^2 - [L_3]^2}{-2[L_2][L_3]}$$

The next step was to calculate theta2 (~lines 231 - 239), which was also extremely challenging.
The trick here, as shown in the following figure, was to identify two separate angles, beta1 and beta2 that were easier to calculate and then use them to help calculate theta2.


(0.35, 2)
$q_3 + \frac{\pi}{2} + 2°$
(1.85, 1.94)
(0.35, 0.75)
m_2
$m_1 = |L_3 - 5| \sin(q_3 - \text{int})$
$m_2 = |L_2 - 3| - \cos(q_3 - \text{int}) \times |L_3 - 5|$
$B_2 = \text{atan2}(m_1 / m_2)$
$\text{atan2}(1.946 - 0.75, 1.85 - 0.35) = \text{atan2}(1.19, 1.5) = 0.673 = 38.6°$

Simplified model of joints 2 - 4 used to calculate theta2

How I fixed the orientation problem (theta4, theta5, theta6)?

Once theta 1, 3 and 2 and were correctly calculated and hence the wrist centre correctly positioned, **the next main step is to calculate angles 4, 5 and 6 which set the end effector orientation. The first sub-step was to numerically evaluate the transform from joint 0 to joint 3 (TF_03) by substituting the calculated values of theta 1, 2 and 3. From this it is simple task to extract the rotational component, R0_3 to see the orientation of the WC. Next we can simply calculated the required rotation from WC to end effector, R3_6, by multiplying the inverse of R0_3 by R0_6. Once R3_6 is known we can calculate theta 4, 5 and 6 by extracting the Euler angles. This code was taken straight from that lesson's exercise without changes.**

Results Discussion:

I spent over 80 hours going through additional readings and practicing other stuff to understand the kinematics correctly. I struggled a lot but when the arm correctly picked the object off the shelf and Successfully dropped it into the dropbox it was a great relief for me.

However, It doesn't pick up successfully every single time.

Suggestions For Improvement

Sometimes the arm arrives at the correct point, in the correct orientation, however it comes in from the side rather than from the top, and hence knocks the object of the shelf before it has the chance to pick it up. Since it is clear the final orientation of the arm is correct, I don't believe this is an issue with my inverse kinematics calculation and code, but rather an issue with the motion planning algorithm. This assertion is stronger since I also have noticed similar behaviour in the demo mode, not just when my code is running. Furthermore the motion planning algorithm sometimes returns an error such as 'no route can be found – planning failed' which again suggests some issues with this section of the code. So if I had more time I would like to be able to dive into the motion planning code and fix this, and not just work on the inverse kinematics side.

Another issue I had was with the performance of Gazebo, the code performs extremely slowly. I think some of this issue is due to the heavy graphics loading, but also some of it comes from the many advanced maths and symbolic operations that have to be run more every inverse kinematics point. I moved many items outside the main loop in the code so they are only run once, rather than for every point, but it still seems not enough. If I have more time I would like to further optimise the code so it runs more smoothly. This in turn would allow me to further test the arm in more different situations, and hence further improve the accuracy of the calculated angles.