Robotic Manipulation

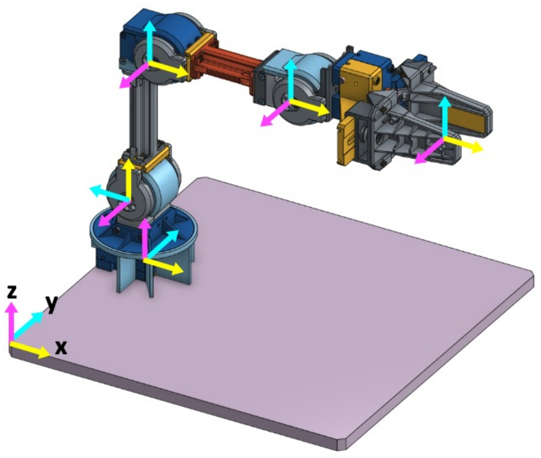
Ziduan Li

CID: 01864493

1 Task 1 - Model the Robot

1.1 Task 1a - Frame Assignment and DH Table

The robotic arm we are using has five degrees of freedom (5 DOF). Four of the actuators are responsible for rotating the arm around their respective axes, while the fifth actuator at the end of the arm controls the opening and closing of the gripper.When assigning frames, we start by establishing a coordinate system for each joint. The axis around which each actuator rotates is designated as the z-axis. Within the plane perpendicular to this, we then define the x and y axes. We position the origin of the coordinate system at the location of each joint, and as depicted in Figure 1a, we establish the coordinate framework accordingly. With such assignment, we are able to derive a clear and simple DH table since all arm components lie essentially in the same plane, except for one joint that introduces a 90-degree rotation.



(a) Coordinate system for the robot arm

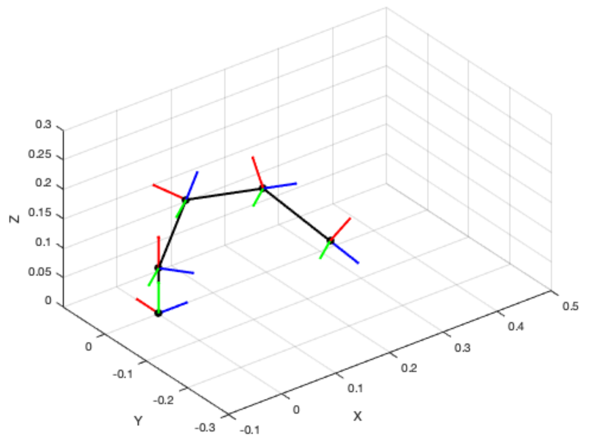
i	α_{i-1}	a_{i-1}	d_i	θ_i
0	0	0	0.077	θ_0
1	$\frac{\pi}{2}$	0	0	θ_1
2	0	0.13	0	θ_2
3	0	0.124	0	θ_3
4	0	0.126	0	θ_4

(b) DH Table

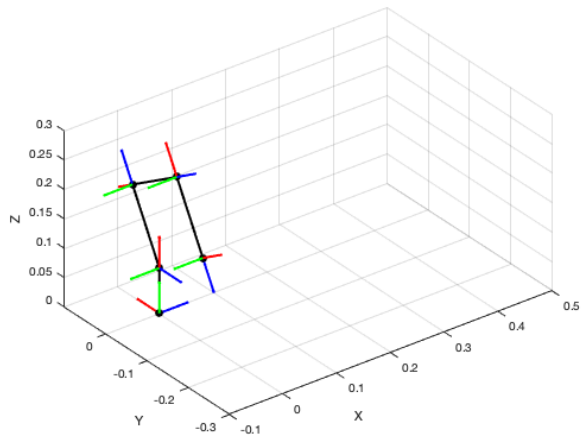
Figure 1

1.2 Task 1b - Graphical Simulation

Figure 2 displays the simulation of the robot based on the previous DH table. In this simulation, the z-axis is represented by a green line, while the x and y axes are depicted by blue and red lines, respectively. Figure 2a and Figure 2b demonstrate the outcomes of executing forward kinematics as the theta values progressively increase.



(a)

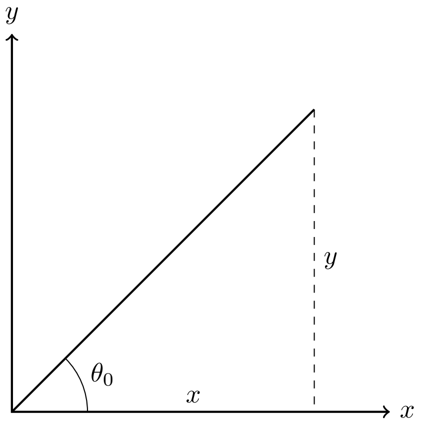


(b)

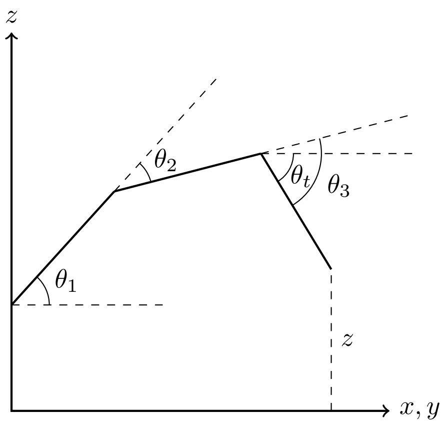
Figure 2: Robot arm simulation

1.3 Task 1c - Inverse Kinematics

We have applied the analytical method to derive the inverse kinematics function of the robot arm. This method starts with the known coordinates (x, y, z) of the end effector, which represents the target position for the robot arm. In addition, we specify the angle between the end joint of the robot arm and the horizontal plane, referred to as theta target (θ_t). This angle is also the angle between the gripper and the horizontal plane. When the gripper points downward, θ_t is set as a negative value. This signifies that the angle is measured clockwise from the horizontal to the gripper's orientation. These coordinates and theta target serve as inputs to the function, allowing us to derive the necessary calculations for the inverse kinematics starting from the position of the end effector.



(a) Top View of the Robot



(b) Side View of the Robot

Figure 3

As shown in Figure 3, we see how each θ angle is distributed. Our goal with the inverse kinematics function is to determine the necessary θ values to reach the intended position. From Figure 3a, we can see that θ_0 can be easily calculated using $\tan^{-1}\left(\frac{y}{x}\right)$. Calculating other θ values appears to be more complex. However, it's important to observe that the projections of all the links onto the xy-plane always align on a straight line, indicating that when we look at the entire robot arm from the side, it can be perceived as a geometric shape on a two-dimensional plane. Based on the image in Figure 3b, we can employ a geometric approach to solve for each θ. By utilizing known link lengths and adding auxiliary lines, we can apply various trigonometric formulas. This method allows us to precisely determine the necessary angles for reaching the target position, using a combination of geometry and trigonometry.

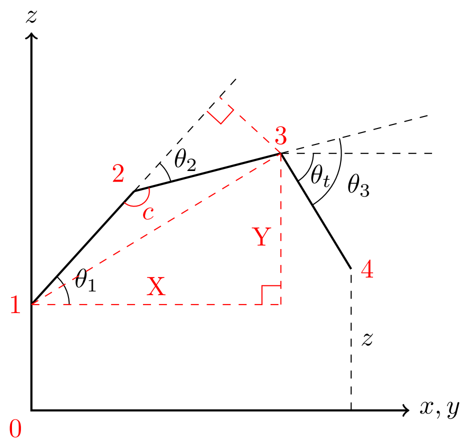


Figure 4

In Figure 4, we've outlined the auxiliary lines and identified the joints. Focusing on joint 2 and the links it connects, we draw a line between joints 1 and 3 to form an obtuse triangle with angle c. The lengths of the sides forming this angle are known, corresponding to the link lengths. A bit of geometry gives us the length of the base opposite angle c, and from there, we can apply the law of cosines to calculate $\cos(c)$. This allows us to determine θ_2. To calculate the length of the opposite side of angle c, we've drawn another two auxiliary lines X and Y. They constitute the two perpendicular sides of a right triangle and we can thus calculate the length of its hypotenuse which is the opposite side of angle c. The length X is the projection of the end effector on the xy-plane minus the projection of the last link on the same line. Since we know the target angle θ_t and the length of the last link, it's easy to calculate the projection with $\cos(\theta_\mathrm{t}) \times 0.126$. Hence, $X = \sqrt{x^2 + y^2} - 0.126\cos(\theta_\mathrm{t})$. The length Y is the z-coordinate value of the end effector minus the length of the first link perpendicular to the xy-plane, plus the projection of the last link along the z-axis. Since θ_t is a negative value, the final expression for Y is $Y = z - 0.077 - 0.14\sin(\theta_\mathrm{t})$. Next, we can use the cosine rule to calculate $\cos(c)$ with the function:

$$\cos(c) = \frac{(Y)^2 + (X)^2 - (0.13)^2 - (0.124)^2}{2 \times 0.13 \times 0.124},$$

Given that $\sin^2(c)+\cos^2(c) = 1$, we can derive that $\sin(c) = \pm\sqrt{1 - \cos^2(c)}$. The $\tan(\theta_2)$ is given by $\sin(\theta_2)/\cos(\theta_2)$. Since c and θ_2 are supplementary angles, and here θ_2 is defined as a negative rotation clockwise relative to the extension of link2, $\sin(\theta_2)$ takes a negative value. Therefore, we conclude $\sin(c) = -\sqrt{1 - \cos^2(c)}$ and $\theta_2 = \tan^{-1}\left(\frac{\sin(c)}{\cos(c)}\right)$.

Next, we draw a perpendicular line from joint3 to the extension of link2 as shown in Figure 4. This creates another right triangle with two side lengths $k1 = 0.124\sin(\theta_2)$ and $k2 = 0.13 + 0.124\cos(\theta_2)$. Then, we can derive the value of θ_1 with the function $\theta_1 = \tan^{-1}\left(\frac{Y}{X}\right) - \tan^{-1}\left(\frac{k1}{k2}\right)$.

For θ_3, it is derived from θ_t and the angle between link3 and the horizontal plane. This angle can be calculated by the difference between θ_1 and θ_2 using the Corresponding Angles Theorem. Here, since θ_1 represents the angle of rotation of link2 in the counterclockwise direction, it is considered positive, while θ_2, being in the opposite direction, is negative. Therefore, the angle between θ_1 and θ_2 is calculated as $\theta_1 + \theta_2$. The final formula for θ_3 is $\theta_3 = \theta_\mathrm{t} - (\theta_1 + \theta_2)$.

1.4 Task 1d - Square Drawing

The input to the inverse kinematics function includes the target coordinate values and θ_t, hence we used MATLAB code to generate the positions of points that make up squares on each plane. Initially, we determined the coordinates of each square's four vertices. We then connected these vertices with line segments, which were subdivided into uniformly spaced points. These point coordinates were recorded, and input sequentially into the inverse kinematics function through a loop, thereby generating the final square images as shown below.

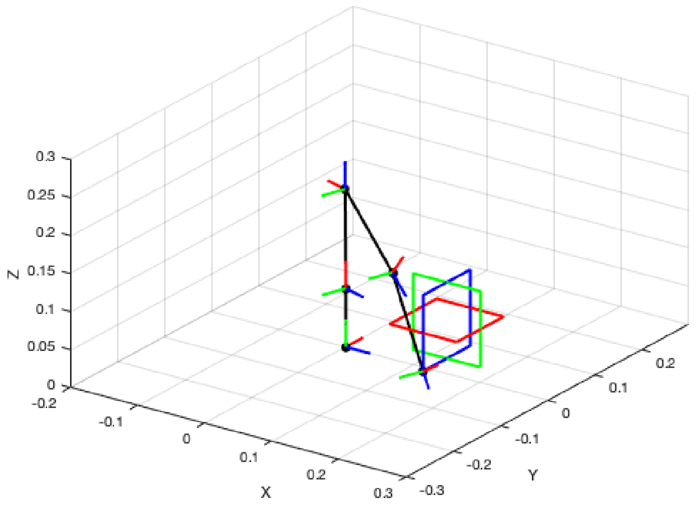


Figure 5

2 Task 2– Pick and Place

When applying the calculated inverse kinematics equations to the actual robotic arm, it is important to consider several factors. First, we need to determine how to convert the output values from the equations into the value that the robotic arm's controller can read. Additionally, we must take into account the physical errors and system offsets that exist in practical operations.

We can determine the input values for each actuator through the use of DynamixelWizard, which are real numbers ranging from 0 to 4095, corresponding to an angle range of 0 to 360 degrees. This means we need to convert the calculated theta values to corresponding values between 0 and 4095. This can be simply achieved by dividing each theta value by 2π and then multiplying by 4096. Furthermore, we need to measure the angle values returned by each actuator when the robotic arm is in its initial state, which is the position where all thetas are zero. This helps us correctly convert the theta values returned from inverse kinematics function into the required rotation angles for the actuators. The following equations correspond to the angles that actuators 0-3 need to rotate, as calculated.

$$\begin{aligned}
\text{theta0} &= 2048 + \frac{\theta_0}{2\pi} \times 4096 \\
\text{theta1} &= 3072 - \frac{\theta_1}{2\pi} \times 4096 \\
\text{theta2} &= 1024 - \frac{\theta_2}{2\pi} \times 4096 \\
\text{theta3} &= 2048 - \frac{\theta_3}{2\pi} \times 4096
\end{aligned}$$

For the actuator that controls the opening and closing of the gripper, we measured the returned angle values in both the open and closed states using DynamixelWizard. We have set 1600 as the closed position to grip the block, and 2850 as the open position.

We have also implemented profile velocity control for each actuator. This control allows us to manage the acceleration and deceleration at the beginning and end of each movement of the robotic arm, ensuring the smoothness and precision of the actions.

By entering the respective actuator angle values into the goal position control function, the motors start movement toward the desired angles and positions. During the subsequent cube picking, rotating, and stacking tasks, we consistently input the target coordinates for the robotic arm and the preferred orientation angle for the gripper (dependent on the theta target). These inputs are processed through the inverse kinematics function. The calculated theta values are then converted and submitted to the goal position control, enabling to complete the tasks.

A MATLAB Code for Inverse Kinematics

```matlab
function [theta_array] = inv_kin_robot(x, y, z, theta_target)

% Finding theta0
theta0 = atan2(y, x);

% Determine projection of the robot arm on x-y plane
x_y = sqrt(x^2 + y^2);

X = x_y - 0.126*cos(theta_target);
Y = z - 0.077 - 0.126*sin(theta_target);

% Finding theta2
cs = ((Y)^2 + (X)^2 - (0.13)^2 - (0.124)^2) / (2 * 0.13 * 0.124);
sn = -sqrt(1 - (c2)^2);
theta2 = atan2(sn, cs);

% Finding theta1
k1 = 0.13 + 0.124 * cos(theta2);
k2 = 0.124 * sin(theta2);
theta1 = atan2(Y, X) - atan2(k2, k1);

% Finding theta3
theta3 = theta_target - (theta1 + theta2);

theta_array = [0, theta0, theta1, theta2, theta3];

end
```