Analytical Project Report: Nut tightening robotic arm manipulator

Yu Xia
Robotics
University of California, Riverside
yxia072@ucr.edu

Fnu Heman
Robotics
University of California, Riverside
fhema002@ucr.edu

***Abstract*—The robot was designed to solve the problem of tightening screws in tight places . It's 4 Dof robot arm(RRPR).Forward kinematics and reverse kinematics are analyzed Forward Kinematics(PoE,D-H),Inverse Kinematics (by numerical),Velocity Kinematics and Statics. the Code Link: https://github.com/orangelee89/EE_283.git**

I. Introduction

In life, this situation often occurs because of space constraints cause screwing is difficult to complete, to solve this problem.It's difficult to find one tool that solves a variety of problems.So ,we designed a 4Dof robot arm. The robot arm can complete the task in a small space. For example, Fig.1.

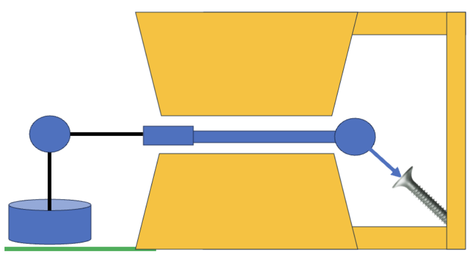

Fig. 1. Case Example

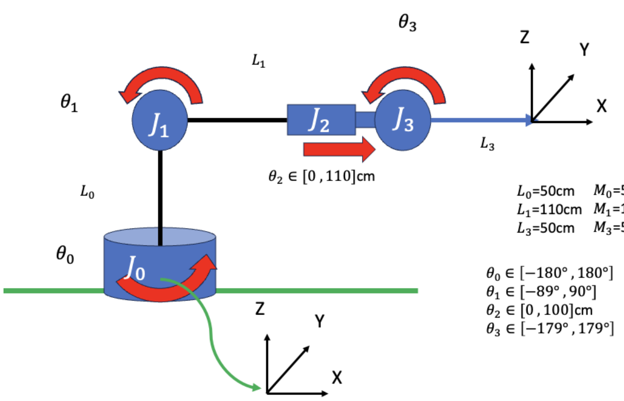

Fig. 2. Configuration and home position

As shown in Fig.2, our robot arm consists of 1 prismatic joint and 3 revolute joints. The limit of each revolute joints are $\theta_0 \in [180°, 180°]$ and $\theta_1 \in [89°, 90°], \theta_3 \in [179°, 179°]$, and the limit of prismatic joint is [0, 100]cm. The length of L_0 is 50cm, L_1 is 110cm and L_3 is 50cm. The weight of L_0 , L_1 and L_3 is 50g, 100g, and 50g respectively. To simplify the calculation, we assume mass of each link are all concentrated at their end point. By these configuration, the robot arm can handle different heights within its working scope.

II. TECHNICAL APPROACH

A. Forward Kinematics(PoE)

Forward kinematics is used to calculate the end effector transformation given joint state $\theta = (\theta_1, \theta_2, \theta_3, \theta_4)$. In our project, PoE is used to calculate the forward kinematics.

According to the configuration in Fig 2, we write down the zero position transformation matrix M below:

$$M = \begin{bmatrix} 1 & 0 & 0 & (L_1 + L_3) \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & L_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

We can then determine the screw axis of each joint, as shown in the table below:

i	ω_i	q_i	v_i
0	$(0, 0, 1)$	$(0, 0, 0)$	$(0, 0, 0)$
1	$(0, -1, 0)$	$0, 0, L_0$	$(L_0, 0, 0)$
2	$(1, 0, 0)$	$-$	$(0, 0, 1)$
3	$(0, -1, 0)$	$(L_1, 0, L_0)$	$(L_0, 0, L_1)$

Where the ω_i is the screw axis of joints. The end-effector transformation $T_{sb}(\theta)$ can be written as:

$$T_{sb}(\theta) = e^{[S_0]\theta_0} e^{[S_1]\theta_1} e^{[S_2]\theta_2} e^{[S_3]\theta_3} M$$

Where

$$S_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} S_2 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} S_3 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ L_1 \\ 0 \end{bmatrix} S_4 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ L_1 \\ -L_2 \end{bmatrix}$$

By putting the desired value of joint angles θ in $T_{sb}(\theta)$ we can get the position of the end effector.

B. Forward Kinematics(D-H)

D-H Parameters:calculate the position and orientation $T(\theta)$, robot's end-effector from its joint coordinates θ. Set up each joint's frame,as Fig.3.

where

$$T(\theta) = T_{0n}(\theta_{0,\ldots,n}) = T_{01}(\theta_1)T_{12}(\theta_2)\cdots T_{n-1,n}(\theta_n)$$

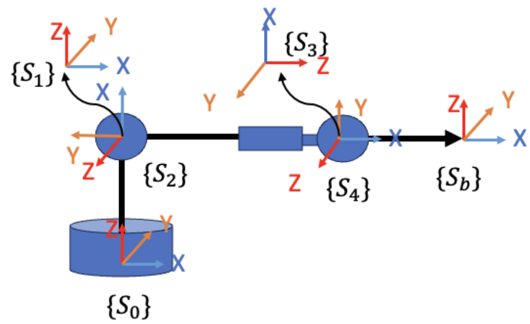

Fig. 3. D-H Setup frames.

So,we get D-H Parameters as Fig.4.In this case, we use Fig.4 to get $T_{i-1,i}$.

C. Inverse Kinematics

Given the end effector position, inverse kinematics is used to find out the joint state $\theta = (\theta_1, \theta_2, \theta_3, \theta_4)$ to support the position. The end-point position can be calculated by trigonometry but it will be very tedious and can be difficult especially due to the existence of multiple solutions. For solving the inverse kinematics in a general way we apply the gradient descent method.

The goal for this approach is to minimize an objective function E(θ), which is quadratic to the error between the real position and computed position, see Eq.1 and 2.

$$x(\theta) : \begin{bmatrix} x \\ z \end{bmatrix} = \begin{bmatrix} L_1 \cos\theta_1 + \theta_2 \cos\theta_1 + L_3 \cos(\theta_1 + \theta_2) \\ L_1 \sin\theta_1 + \theta_2 \sin\theta_1 + L_3 \cos(\theta_1 + \theta_2) \end{bmatrix} \tag{1}$$

where $x = x^2 + y^2$.

$$E(\theta) = \frac{1}{2}(x(\theta) - X)^\top \underbrace{(x(\theta) - X)}_{K} \tag{2}$$

α_{i-1}	a_{i-1}	d_i	φ_i
0	0	L_0	θ_0
90°	0	0	θ_1
90°	0	$L_1 + \theta_2$	0
-90°	0	0	θ_3-90°
-90°	L_3	0	0

Fig. 4. D-H Parameters

where $\bar{x}$ is the desired position.

First, we calculate the Jacobian matrix $\mathbf{J} \in \mathbb{R}^{2\times3}$ and the result is shown in Eq. 3.

$$\mathbf{J} = \frac{dx}{d\theta} = \begin{bmatrix} \frac{dx_x}{d\theta_1} & \frac{dx_x}{d\theta_2} & \frac{dx_x}{d\theta_3} \\ \frac{dx_z}{d\theta_1} & \frac{dx_z}{d\theta_2} & \frac{dd_z}{d\theta_3} \end{bmatrix}$$
$$\begin{bmatrix} -sin\theta_1(L_1 + \theta_2) - \sin(\theta_1 + \theta_3) L_3 & \cos\theta_1 & -\sin(\theta_1 + \theta_3) L_3 \\ sin\theta_1(L_1 + \theta_2) + \cos(\theta_1 + \theta_3) L_3 & \sin\theta_1 & +\cos(\theta_1 + \theta_3) L_3 \end{bmatrix} \tag{3}$$

By the chain rule, the gradient of the objective function $E(\theta)$ can be decomposed into Jacobian transposed multiplied by the v term, see Eq. 4.

$$\nabla E = \frac{dE}{d\theta} = \underbrace{\frac{dx^\top}{d\theta}}_{m\times n} \cdot \underbrace{\frac{dE}{dx}}_{n\times 1} = \frac{dx^\top}{d\theta} \cdot K = J^\top K \tag{4}$$

In the gradient descent method, if we follow the opposite direction of the gradient, we can reach to the minimum value. Therefore, we start with $x_0 = [0, 0, 0]^T$ as the initial value for x_k, and set the learning rate α as 0.001. Then we iteratively update the θ using the gradient descent rule (Eq. 5) until the error between x_k and x_{k+1} is converged. In our case, the error threshold we set is 0.0001.

$$x_{i+1} = x_i - \alpha * \nabla E \tag{5}$$

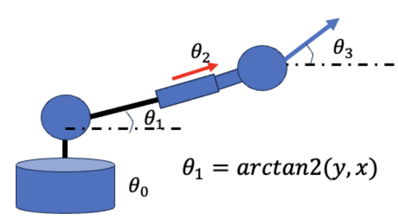

Fig. 5. Configuration of the robot arm.

By gradient descent method, we easily computed $[\theta_1 2, \theta_2, \theta_3]$. The last joint value, θ_1 can also be calculated by $\theta_1 = tan^{-1}(x/y)$ (Refer Fig. 5).

We use gradient descent to solve this problem, so the angle obtained may not be the optimal solution. The solution is closely related to the initialized value.

D. Velocity Kinematics and Statics

Velocity kinematics is used to derive the twist of end-effector from given joint positions and velocities. For this, we need to calculate $J(\theta) \in \mathbb{R}^{m\times n}$.

$$\dot{x} = \frac{\partial f(\theta)}{\partial \theta}\dot{\theta} = J(\theta)\dot{\theta} \tag{6}$$

Jacobian can be easily computed by using the screw axis for every joint. We find the screw axis for every joint and keep the previous joint screw intact for computing the next joint screw.The configuration as shown in Fig.6.

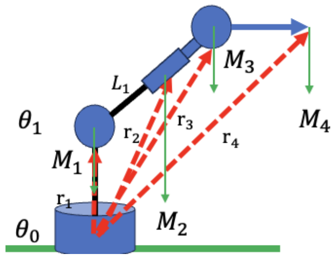

Fig. 6. Configuration of the robot arm

We compute the space Jacobian $J_s(\theta) \in \mathbb{R}^{6\times n}$ that relates joint rate vector $\dot{\theta}$ to spatial twist $\mathcal{V}_s$ via $\mathcal{V}_s = J_s(\theta)\dot{\theta}$. The geometric Jacobian can be derived as shown below.

$$J_s = [J_{S_0} \quad J_{S_1} \quad J_{S_2} \quad J_{S_3}] \tag{7}$$

Here

$$J_{S_0} = S_0 \tag{8}$$
$$J_{S_1} = [Ad_{e^{[S_0]\theta_0}}]S_2 \tag{9}$$
$$J_{S_2} = [Ad_{e^{[S_0]\theta_0}e^{[S_1]\theta_1}}]S_3 \tag{10}$$
$$J_{S_3} = [Ad_{e^{[S_0]\theta_0}e^{[S_1]\theta_1}e^{[S_2]\theta_2}}]S_4 \tag{11}$$

Our computed Jacobian is shown below.

$$J_s = \begin{bmatrix} 0 & 0 & 0 & 0 \\ -1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \\ 22 & 1 & 5 & 0 \\ 0 & 0 & 0 & 0 \\ -6 & 0 & 0 & 0 \end{bmatrix} \tag{12}$$

We can now calculate the end effector twist according to the fixed frame given the joint velocity $\dot{\theta}$ as

$$\mathcal{V}_s = J_s(\theta)\dot{\theta} \tag{13}$$

Similarly using the priciples of conservation of power we can show that

$$\tau = J^T(\theta)\mathcal{F}_s \tag{14}$$

If an external wrench $-\mathcal{F}$ is applied to the end effector when the robot is at the equilibrium with joint values θ the above equation calculates the joint torques needed to generate the opposing wrench $\mathcal{F}$ keeping the robot at equilibrium.

Let us take the same case as Fig.6. The one of joint variables θ as shown below which is a practical case encountered while tightening.

$$\theta = \begin{bmatrix} 0 & \frac{\pi}{4} & 1 & \frac{-\pi}{4} \end{bmatrix}^T$$

For this case, we will find forces and moments for each link. The wrench $\mathcal{F}_i$ for any Joint i can be thus computed as

$$\mathcal{F}_i = \begin{bmatrix} m_{b_i} \\ F_i \end{bmatrix} = \begin{bmatrix} r_i \times F_i \\ F_i \end{bmatrix} \tag{15}$$

where r_i is the vector pointing from the space body frame to the centre of mass of Link i. For link 1 the moment is 0 as the force vector and centre of mass coincide. Therefore, for Link 1 we have $F_0 = [0 \quad 0 \quad 0.5]^T$ due to gravity and $m_b0 = [0 \quad 0 \quad 0]^T$. Similarly, computing for link 2 and 3 we get r_i, m_i.

$$r_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad r_1 = \begin{bmatrix} 0 \\ 0 \\ L_0 \end{bmatrix} \quad r_2 = \begin{bmatrix} L_1 cos\frac{\pi}{4} \\ 0 \\ L_0 + L_1 cos\frac{\pi}{4} \end{bmatrix} \tag{16}$$

$$r_3 = \begin{bmatrix} (L_1+\theta_2)cos\frac{\pi}{4} \\ 0 \\ L_0 + (L_1+\theta_2)cos\frac{\pi}{4} \end{bmatrix} \tag{17}$$

$$r_4 = \begin{bmatrix} (L_1+\theta_2)cos\frac{\pi}{4} + L_3) \\ 0 \\ (L_1+\theta_2)cos\frac{\pi}{4} + L_3 + L_0 \end{bmatrix} \tag{18}$$

$$m_0 = \begin{bmatrix} 0 \\ 0 \\ 0.5 \end{bmatrix} \quad m_1 = \begin{bmatrix} 0 \\ 0 \\ 0.5 \end{bmatrix} \quad m_2 = \begin{bmatrix} 0 \\ 0 \\ 0.5 \end{bmatrix}$$

$$m_3 = \begin{bmatrix} 0 \\ 0 \\ 0.01 \end{bmatrix} \quad m_4 = \begin{bmatrix} 0 \\ 0 \\ 0.5 \end{bmatrix}$$

After using the above equation and substituting values of lengths we find the corresponding individual wrench force for each link below.

$$\mathcal{F}_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \mathcal{F}_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0.5 \end{bmatrix} \mathcal{F}_2 = \begin{bmatrix} 0 \\ -3.889 \\ 0 \\ 0 \\ 0 \\ 0.5 \end{bmatrix} \mathcal{F}_3 = \begin{bmatrix} 0 \\ -8.4852 \\ 0 \\ 0 \\ 0 \\ 0.01 \end{bmatrix}$$

$$\mathcal{F}_4 = \begin{bmatrix} 0 \\ -6,7426 \\ 0 \\ 0 \\ 0 \\ 0.5 \end{bmatrix} \mathcal{F}_{net} = \begin{bmatrix} 0 \\ -19.11 \\ 0 \\ 0 \\ 0 \\ 3 \end{bmatrix}$$

Here $\mathcal{F}_{net} = \mathcal{F}_1 + \cdots + \mathcal{F}_4$. Now using Equation 14 we can compute the final torque of every motor by substituting calculated values

$$\tau = J_s^T(\theta_g)\mathcal{F}_{net} \tag{19}$$

III. ILLUSTRATION OF OBTAINED RESULTS

We have fully implemented all the content above in Python so that we can easily verify our results. The code can be found on our GitHub repository: https://github.com/orangelee89/EE_283.git.

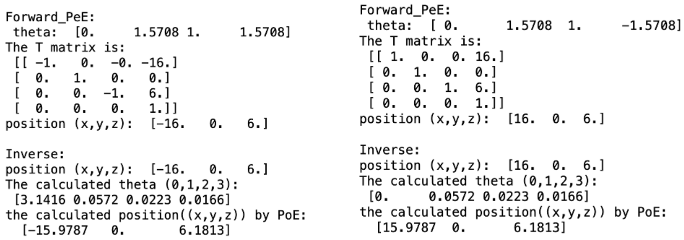

Fig. 7. Results of forward and inverse kinematics.

A. Forward Kinematics & Inverse Kinematics

To test forward kinematics and inverse kinematics, we calculate the end effector positions p according to an arbitary state θ, then we calculate the state $\hat{\theta}$ from p by inverse kinematics. Finally, we use the $\hat{\theta}$ to calculate the $\hat{p}$ again and compare the difference between $\hat{p}$ and p.

We take the case as Fig. 6, given joints values $\theta = [0, 1.5708, 1, -1.5708]$, the end effector positions p is $[16, 0, 6]$. By gradient descent, we get $\hat{\theta} = [0, 0.0572, 0.0223, 0.0166]$. Then, the recomputed $\hat{p}$ is $[15.9787, 0, 6.1813]$. The reprojection error is 1×10^{-4}, which is very close to the original end effector position. The calculated theta is different with given theta,due to unique solutions.Both two case are the case of unique solutions.

B. Velocity kinematics and Statics

For the velocity kinematics we refer to 13 and set value θ and $\dot{\theta}$. Using Equation 14 we can compute the final torque of every motor to keep the system in equilibrium by substituting calculated values

$$\tau = J_s^T(\theta_g)\mathcal{F}_{net} \tag{20}$$

$$\tau = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 22 & 0 & -6 \end{bmatrix}^T \begin{bmatrix} 0 \\ -19.11 \\ 0 \\ 0 \\ 0 \\ 3 \end{bmatrix} \tag{21}$$

$$= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -2 & 0 & 0 \\ 0 & 1 & 0 & -0.60 & -1.4 & \end{bmatrix} \begin{bmatrix} 0 \\ 25.6 \\ 0 \\ 0 \\ 0 \\ 24.5 \end{bmatrix} \tag{22}$$

$$\tau = \begin{bmatrix} 0 & 19.11 & 0 & 1.11 \end{bmatrix}^T \tag{23}$$

Using the above calculation we can see that torque for joint 0 is $\tau_0 = 0$ N-m , torque for joint 1 is $\tau_1 = 10.11$ N-m, torque for joint 2 is $\tau_2 = 0$ N-m and torque for joint 4 is $\tau_3 = 11.11$ N-m.

IV. REFERENCES

[1] Kevin Lynch and Frank Park. 2017. Modern Robotics: Mechanics, Planning, and Control.

[2] NumPy quickstart - NumPy v1.21 Manual. (n.d.). Retrieved December 11, 2021, from https://numpy.org/doc/stable/user/quickstart.html.

[3] Guan Zhaoxin, Li Han, Zuo Zhijiang,Pan Libo. Design a Robot System for Tomato Picking Based on YOLO v5