Position or Force Control for Robotic Manipulators: Closing Bottle Cap with OpenMANIPULATOR-X

Tamizhanban A G
B.Tech Electrical Engineering
IITGN
Palaj, Gandhinagar
tamizhanban.ag@iitgn.ac.in

***Abstract*— Robotic manipulators are pivotal in various industrial and domestic applications. This study focuses on closing bottle caps using the OpenMANIPULATOR-X robotic arm. OpenMANIPULATOR-X is a manipulator made for educational purposes, and it is very compatible with ROS. Our objective is to explore force control or position control strategies to achieve precise manipulation. We delve into the configuration space of OpenMANIPULATOR-X, discussing joint angles and degrees of freedom. Forward kinematics equations are derived to determine end-effector positions, while inverse kinematics enables us to compute joint angles from desired positions.**

The ROS (Robot Operating System) framework facilitates control of OpenMANIPULATOR-X. We implement force and position control algorithms, leveraging computer vision (OpenCV) for cap position detection.

Our experimental results reveal trends in end-effector force and position during cap closure. By analysing these trends, we infer the effectiveness of different control approaches. This research contributes to advancing manipulator capabilities, demonstrating the potential for robotic arms to handle intricate tasks beyond industrial settings.

I. Introduction

Robotic manipulators, often called robotic arms, are indispensable tools in modern automation. Their versatility extends from industrial assembly lines to everyday tasks, making them essential components in various domains. In this study, we delve into the intricate world of manipulator control, focusing specifically on the task of closing bottle caps using the OpenMANIPULATOR-X robotic arm.

Our research project centres on a seemingly straightforward yet nontrivial task: closing bottle caps. While humans perform this action effortlessly, replicating it with a robotic arm involves intricate control mechanisms. We aim to explore two fundamental approaches: force control and position control. By understanding the underlying principles and implementing them on the OpenMANIPULATOR-X platform, we contribute to advancing manipulator capabilities.

II. Configuration Space of OpenMANIPULATOR-X

The configuration space is the space of all possible positions and orientations of the robot. In our case, i.e. for an n-joint manipulator, the C-space is an n-dimensional space, where each dimension corresponds to a joint variable. It is always preferable to represent the C-space with the minimal set of parameters and to know the minimal set of parameters; we need to find the degrees of freedom (DoF) of OpenMANIPULATOR-X. Gruebler's formula helps us to calculate the Dof for a system. For spatial mechanisms, it's given by:

$$F = 6 \times (L - 1 - J) + \sum_{i=1}^{J} f_i \qquad (1)$$

where,

F = Degrees of Freedom (DoF) of the mechanism

L = Number of links (including the base)

J = Number of joints

f_i = Degrees of freedom contributed by the i-th joint

The degrees of freedom for joints depend on the types of joints present in the mechanism. In our case, the OpenMANIPULATOR-X contains only revolute joints, and the DoF of all kinds of joints can be observed in Fig. 1.

Joint type	dof f	Constraints c between two planar rigid bodies	Constraints c between two spatial rigid bodies
Revolute (R)	1	2	5
Prismatic (P)	1	2	5
Helical (H)	1	N/A	5
Cylindrical (C)	2	N/A	4
Universal (U)	2	N/A	4
Spherical (S)	3	N/A	3

Fig. 1. The number of DoF f and constraints c provided by common joints. (Source: Adapted from [1])

Applying the obtained data in equation (1), we get the DoF of OpenMANIPULATOR-X as four, so the minimum number of parameters in which we can represent the C-space of the mechanism is four, and it can be represented as follows:

$$C = \{\theta_1, \theta_2, \theta_3, \theta_4\}$$

III. Forward Kinematics

Forward kinematics is a fundamental concept in robotics that determines the position and orientation of a manipulator's end-effector based on its joint parameters. For the OpenMANIPULATOR-X, which features four revolute joints, forward kinematics involves computing the transformation from the base to the end-effector, given the angles of these joints. This is achieved using transformation matrices, which describe the rotation and translation induced by each joint angle.

The overall transformation matrix T from the base frame to the end-effector frame is obtained by multiplying the individual transformation matrices T_i for each joint:

$$T = T_1 T_2 T_3 T_4 \qquad (2)$$

The exact position and orientation of the end-effector are determined from the known joint angles, facilitating the accurate execution of these tasks. Forward kinematics ensures that the manipulator can place its end-effector at the desired location in the workspace. The Robot Operating System (ROS) simplifies forward kinematics for the

OpenMANIPULATOR-X by providing pre-defined topics for inputting joint angles and obtaining the end-effector's position and orientation. Users can send the joint angles to ROS topics and receive the transformation results without calculating the kinematic equations manually.

IV. Inverse Kinematics

Inverse kinematics (IK) is the process of determining the joint angles of a robotic manipulator that achieve a specified end-effector position and orientation. For the OpenMANIPULATOR-X, which has four revolute joints, IK is essential for tasks requiring precise end-effector positioning, such as closing bottle caps. Unlike forward kinematics, which computes the end-effector's pose from known joint angles, IK involves finding joint angles that result in a desired end-effector pose.

The challenge of inverse kinematics is to solve for joint angles. $\{\theta_1, \theta_2, \theta_3, \theta_4\}$ that achieve a given end-effector position $p = [x, y, z]$ and orientation R. This is mathematically represented by:

$$T = \begin{bmatrix} R & p \\ 0^T & 1 \end{bmatrix} \quad (3)$$

where R is the rotation matrix (3×3), p is the position vector (3×1), and 0^T is a row vector of zeros. The transformation matrix T thus combines both the orientation and position of the end-effector, allowing us to obtain the joint angles present in the transformation matrix by mapping the two matrices.

Several advanced solvers can be employed to simplify the complexity of manual IK calculations. Tools such as Kinematics and Dynamics Library (KDL), IKFast, and MoveIt! (we use this in our project) automate this process:

- KDL: Provides a robust framework for defining and solving kinematic chains in ROS, supporting various robot configurations.
- IKFast: Generates optimised, analytical IK solutions for specific robot geometries, ensuring real-time performance.
- MoveIt!: Integrates with ROS to offer comprehensive motion planning, including efficient IK solutions, collision detection, and trajectory planning.

V. Force Reading Through Manipulator Jacobian

The Jacobian matrix is a critical concept in robotics that relates the velocities and forces at the end-effector to the velocities and torques at the joints. For the OpenMANIPULATOR-X, understanding the Jacobian is essential for tasks involving force control and plotting end-effector force graphs. The Jacobian matrix J specifically relates the joint velocities $\dot{\theta}$ to the end-effector's linear and angular velocities:

$$v = J\dot{\theta} \quad (4)$$

$$power = \dot{\theta}^T \tau \quad (5)$$

$$power = v_{tip}^T f_{tip} \quad (6)$$

$$from\ (4), (5) and\ (6) we\ can\ derive:$$

$$\tau = J^T f_{tip} \quad (7)$$

where, τ is the joint torque matrix and f_{tip} is the end-effector force matrix of OpenMANIPULATOR-X. The torque matrix is of dimension (4x1) and f_{tip} is the transpose of $\{f_x, f_y, f_z, \tau_x, \tau_y, \tau_z\}$, so it is of (6x1) dimension. The formula for the dimension of the Jacobian matrix for a spatial mechanism is (6xn), where n is the number of joints. Hence, the dimension of J^T is (4x6).

In our case, we need the force matrix. $\{f_x, f_y, f_z, \tau_x, \tau_y, \tau_z\}$ and to obtain the torque matrix, we derive an equation from Fig. 2 and substitute the current values we get by subscribing to the `/joint_states` topic in ROS.

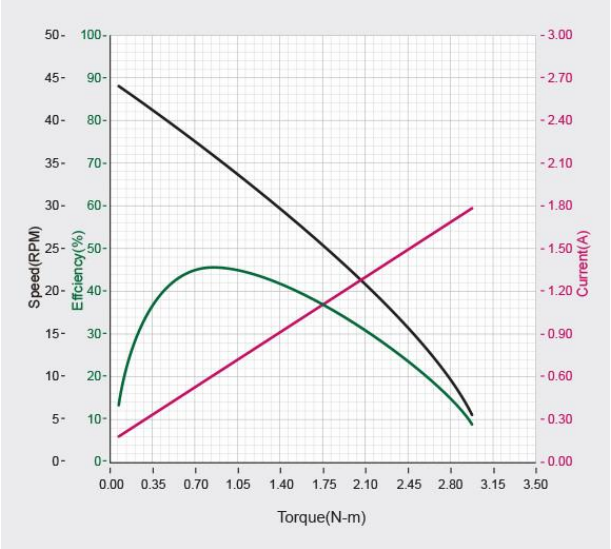

Fig. 2. Performance graph of DYNAMIXEL XM430-W350 motor. (Source: Adapted from [2])

In order to find the Jacobian matrix, we use the help of modules such as PyKDL, which gets the framework of the manipulator and performs mathematical calculations to give us the matrix at the end. We perform SVD on the Jacobian transpose to get the inverse of it since the matrix is not a square matrix. Once we get the Jacobian and torque matrix, the calculations become simple, and we can plot the force vs time graphs to observe the trend.

VI. Methodology

- Moving the end-effector from point A to B using inverse kinematics. The OpenMANIPULATOR-X has certain inbuilt services such as `SetKinematicsPose`, `SetKinematicsPoseRequest`, `SetJointPosition` and `SetJointPositionRequest`. We use them to set the pose and the orientation of the two points and give them a path time to complete the task.
- Once the inverse kinematics work well with the hardware, we implement the code to figure out the centre of a disc placed under the webcam placed over the C-space (since all caps, when viewed from the top, are circular discs). We use the help of OpenCV to get the coordinates of the centre point (x,y) in pixels. We use the `cv2.HoughCircles()` function to figure out circles under the webcam's frame. We alter the parameters inside the function call to make the cap detection flawless.
- We use transformation equations to convert the (x,y) coordinates of the detected cap (in pixels) into coordinates (in metres) of the cap concerning the base link of the OpenMANIPULATOR-X. Once this is done, we can directly give the (x,y) obtained to the inverse kinematics part of the code.

- Now, we apply force control to ensure the manipulator closes the cap tightly. We can also use position control to do the same thing and verify by plotting the force graphs. The same will be explained in the "ALGORITHM USED" topic of the paper.

VII. Challenges Faced

- While figuring out how to perform inverse kinematics to move from point A to point B by simulating the same in RViz, ROS did not call the topics. We learned that Rviz does not monitor all the services and topics.
- While performing force control, to change the torque of the joints, we needed to alter the currents supplied to each of the motors, and there were no publishers and subscribers written to do the same, so we went with position control to perform the same task.
- The manufacturer's provided kinematic solvers could not solve some kinematic poses; hence, we devised a way to make it work, which will be explained in detail in the "ALGORITHM USED" section.

VIII. Algorithm Used

We decided to use position control to accomplish the task. Hence, a new algorithm must be devised to make the manipulator close the cap by applying pressure vertically on the horizontally placed cap. In order to make it work, we first make the end-effector pose vertically above the cap by a distance of two or three centimetres, with a pitch angle of 0 degrees. Then, we make the end-effector close on the cap slowly by giving a z coordinate to the IK solver more than two or three centimetres. Thus, the manipulator will apply more current to the motors to make the end-effector go further, even when the cap is reached.

However, in this method, not all poses support orientation with a pitch angle of 0 degrees, so to tackle that, I found out that the solver was able to solve poses when they were closer to the previous pose. Thus, I made the pitch angle drop uniformly by 5 degrees from its final resting point (3 centimetres above the cap).

IX. Observations and Inference

The observations are made in three cases: when the midpoint's y coordinate is 0, negative and positive. In order to get an idea of what is the frame of reference the observations are taken, Fig. 3 attached below.

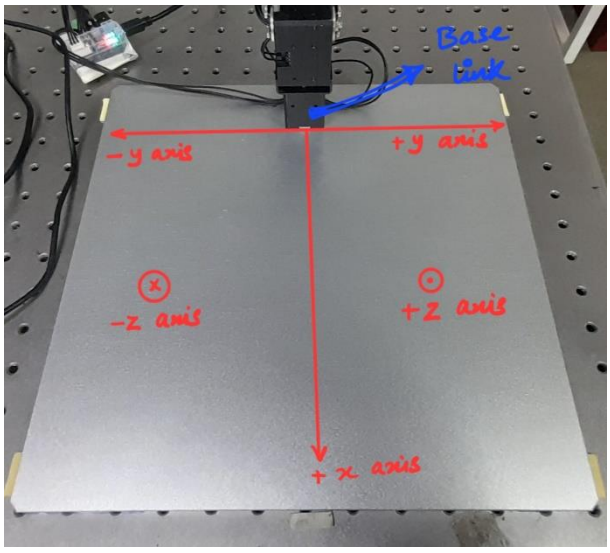

Fig. 3. Base link reference frame or the C-space.

A. *When the y-coordinate is 0*

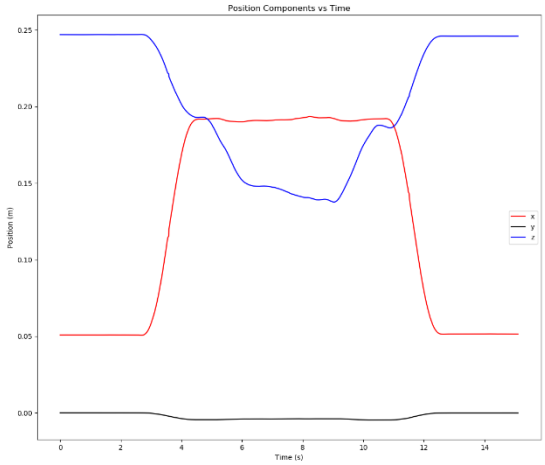

Fig. 4. Position vs time (y=0)

From Fig. 4, we can see that the y coordinate is barely changing, whereas the z coordinate is moving from 25 centimetres to 8 centimetres above the x-y plane since the bottle touches the cap at 9 centimetres. The same is perfectly shown in Fig. 5. where we can see from 2.5 to 8.5 seconds, the Fz value is steeply increasing in the negative z direction, telling us that about 13 N of force is applied in that direction. The force applied to hold the capless bottle is also lesser than holding the bottle with the cap, due to the additional weight of the cap. The other force components, i.e. x and y, also match with their respective position readings.

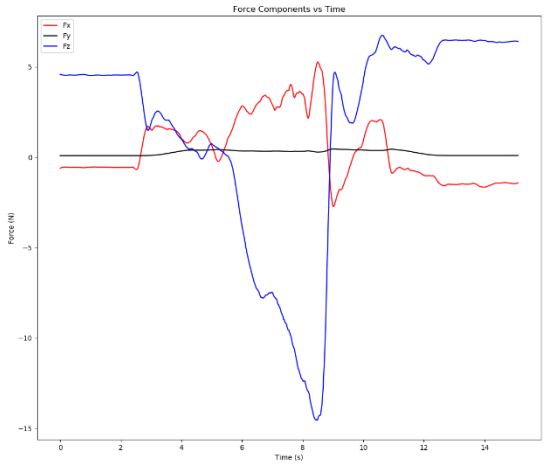

Fig. 5. Force vs time (y=0)

B. *When the y-coordinate is positive*

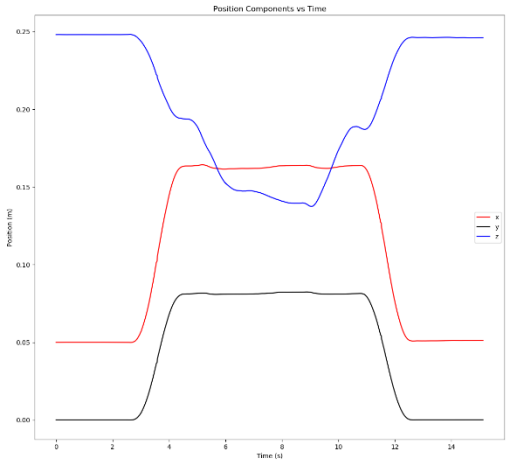

Fig. 6. Position vs time (y is positive)

From Fig. 6, we can see that the y-coordinate is changing positively, moving from a neutral value to a positive value over time. Simultaneously, the z-coordinate moves from 25 centimetres to 8 centimetres above the x-y plane as the bottle touches the cap at 9 centimetres. This behaviour is clearly illustrated in Fig. 7, where from 2.5 to 8.5 seconds, the Fz value is steeply increasing in the negative z direction, indicating that about 16 N of force is applied in that direction. The force applied to hold the capless bottle is also lesser than holding the bottle with the cap, due to the additional weight of the cap. The force components in the x direction remain consistent, while the y-force components reflect the changes corresponding to the positive shift in the y-coordinate.

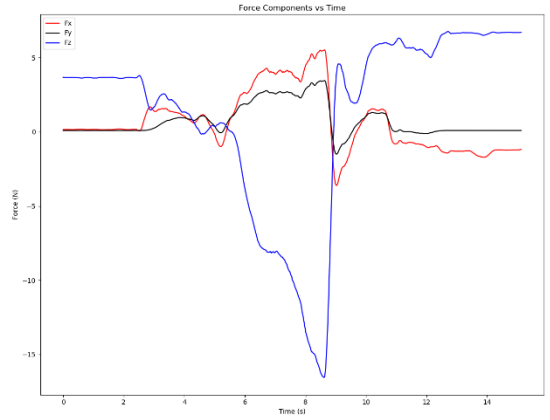

Fig. 7. Force vs time (y is positive)

C. *When the y-coordinate is negative*

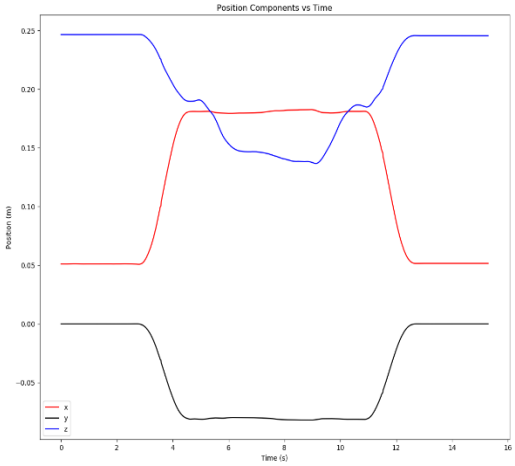

Fig. 8. Position vs time (y is negative)

From Fig. 8, we can see that the y-coordinate is changing negatively, moving from a neutral value to a negative value over time. Simultaneously, the z-coordinate moves from 25 centimetres to 8 centimetres above the x-y plane as the bottle touches the cap at 9 centimetres. This behaviour is clearly illustrated in Fig. 9, where from 2.5 to 8.5 seconds, the Fz value is steeply increasing in the negative z direction, indicating that about 17 N of force is applied in that direction. The force applied to hold the capless bottle is also lesser than holding the bottle with the cap, due to the additional weight of the cap. The force components in the x direction remain consistent, while the y-force components reflect the changes corresponding to the positive shift in the y-coordinate.

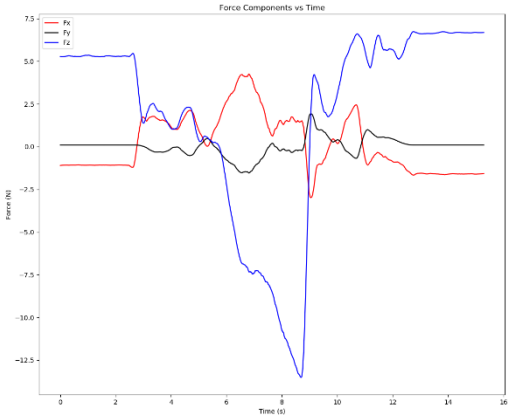

Fig. 9. Force vs time (y is negative)

X. Conclusion

This research on robotic manipulators using the OpenMANIPULATOR-X aimed to optimize the task of closing bottle caps through position control. Our study successfully established a comprehensive understanding of the manipulator's configuration space and demonstrated the application of forward and inverse kinematics to precisely control the robot's movements. By leveraging ROS tools, we simplified forward kinematics, allowing real-time control without manual computations.

We developed and incorporated a suitable position control algorithm that efficiently accomplished the cap-closing task

without relying on force control. This algorithm used precise joint angle calculations to ensure accurate positioning of the end-effector, thus achieving the desired outcomes without the need for force feedback mechanisms. The integration of advanced solvers like KDL and IKFast for inverse kinematics facilitated dynamic adaptation to task variations, further enhancing the robot's ability to perform precise alignments.

In summary, this research highlights the effectiveness of position control in performing intricate tasks like bottle capping with the OpenMANIPULATOR-X. The findings contribute valuable insights into the application of robotic manipulators in automation, demonstrating the potential to achieve complex tasks efficiently using position-based algorithms. Future work could explore further improvements in control strategies and their applications across diverse industrial scenarios.

XI. Acknowledgement

I sincerely thank my advisor, Prof. Harish P. M. Sir, for his invaluable guidance, support, and timely advice throughout this project. His responsible mentorship and prompt responses to my queries were crucial in enabling me to complete the project on time with outstanding results. I also extend my heartfelt gratitude to my friend, Debojit Das, for teaching me key topics, such as the manipulator Jacobian, which significantly contributed to my understanding. Furthermore, I am deeply grateful to IITGN for providing this enriching opportunity through its Summer Research Project Internship, which has greatly enhanced my learning experience and practical skills.

XII. REFERENCES

[1] K. M. Lynch and F. C. Park, *Modern Robotics - Mechanics, Planning, and Control*. Cambridge: Cambridge University Press, 2017.

[2] ROBOTIS, "ROBOTIS e-Manual," ROBOTIS e-Manual. https://emanual.robotis.com/docs/en/dxl/x/xm430-w350/