Closed-loop Control for a Subway Seat Cleaning Robot Manipulator

Boshu Lei
School of Engineering and Applied Science
University of Pennsylvania
Philadelphia, Pennsylvania 19104
Email: leiboshu@seas.upenn.edu

Tianyou Wang
School of Engineering and Applied Science
University of Pennsylvania
Philadelphia, Pennsylvania 19104
Email: wty518@seas.upenn.edu

Tianyi Chen
School of Engineering and Applied Science
University of Pennsylvania
Philadelphia, Pennsylvania 19104
Email: tychen32@seas.upenn.edu

***Abstract*—Seat cleaning requires adequate contact forces applied to the curved surfaces to wipe out stains and dirt. At the same time, high efficiency is preferred for task completion to generate effective profit margins, resulting in fast end-effector motions. To accomplish such tasks, we employed an extended Cartesian impedance control algorithm, which includes geometrical constraints and enables explicit force tracking in a hybrid manner. The adopted unified framework features compliant behavior in the (free) motion task directions and explicit force tracking in the constrained directions. We tested our algorithm in both MATLAB and Gazebo simulations to show that our implementation can effectively control robot arms with different configurations to apply desired contact forces on a variety of surfaces while maintaining accurate and fast motions.**

I. Introduction

Subway systems, vital components of urban transportation, seamlessly connect millions of commuters daily. However, the persistent challenge of maintaining cleanliness within these networks extends beyond mere aesthetics, impacting passenger experience and public health. Nevertheless, the current solution, mostly manual cleaning efforts, prove to be labor-intensive and temporally challenging, exacerbated by the constant influx of dirt, debris, and germs.

This paper delves into the integration of robotic technologies as a transformative solution to address the cleanliness challenges inherent in subway systems. Specifically focusing on seat cleaning, a critical aspect, where conventional methods, such as classical impedance control [1] and hybrid motion-force control [2], encounter challenges including failure of providing accurate force and motion control simultaneously and high-impact corrective actions after the loss of contact with the surface due to the motion tracking error, we explore an innovative solution [3]. Our adopted approach involves the implementation of an extended Cartesian impedance control algorithm, which includes geometrical constraints and enables explicit force tracking in a hybrid manner. Such a unified framework features compliant behavior in the (free) motion task directions and explicit force tracking in the constrained directions.

This innovative algorithm ensures precise and expeditious motions, effectively controlling robot arms with diverse configurations to apply desired contact forces on various surfaces. By overcoming the shortcomings of conventional methodologies, our research represents a paradigm shift in the maintenance of subway cleanliness. We meticulously tested our algorithm in both MATLAB and Gazebo simulations. The outcomes of our simulations showcase the transformative potential of our proposed robotic solution, offering a glimpse into a future where subway environments consistently exude cleanliness, inviting passengers into a pristine and liberating commuting experience, free from the constraints of manual labor.

II. Method

A. Fundamentals

The dynamics of the manipulator can be expressed as

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau + \tau^{\text{ext}} \tag{1}$$

where τ is the joint torque input for the system. The compliant end-effector behavior can be achieved using the classical impedance control

$$\tau_{\text{imp}} = J(q)^T F_{\text{imp}} + g(q) \tag{2}$$

B. Control Design

To decouple the control actions in the motion and force directions. Here the stacked Jacobian matrix $\bar{J}(q)$ is given by

$$\begin{pmatrix} \dot{\Phi} \\ \dot{x} \\ v \end{pmatrix} = \underbrace{\begin{pmatrix} J_{\dot{\Phi}}(q) \\ J_{\dot{x}}(q) \end{pmatrix}}_{\bar{J}(q)} \dot{q} \tag{3}$$

where $\dot{\Phi}$ is the normal direction of the contact surface and $\dot{x}$ is the tangential direction. The constraints for forces are defined in $\dot{\Phi}$ direction while the constraints for the motion of the end effector are defined in $\dot{x}$ direction.

To achieve the hybrid force-motion control, the control command is

$$\tau = g + \bar{J}(q)^T \begin{pmatrix} F_{\text{ctrl}}^{\dot{\Phi}} \\ F_{\text{ctrl}}^{\dot{x}} \end{pmatrix} \tag{4}$$

containing gravity compensation. The control force in the motion directions can be generated with PD control as

$$F_{\text{ctrl}} = M_x \ddot{x}_{\text{des}} + C_x \dot{x}_{\text{des}} - K_x \tilde{x} - D_x \dot{x} \tag{5}$$

x_{des} describes the desired task-space coordinates for the unconstrained space at the end effector. M_x and C_x are task-space inertia and Coriolis and centrifugal matrices in the motion direction, respectively, which are calculated by

$$M_x = J_{\dot{x}}^{-1}(q) M J_{\dot{x}}^{-1}(q); \tag{6}$$

$$C_x \dot{x}_{\text{des}} = J_{\dot{x}}(q)^{-T} C \dot{q}_{\text{des}} - M_x \dot{J}_{\dot{x}}(q) \dot{q}_{\text{des}}; \tag{7}$$

For the calculation of $F_{\text{ctrl}}^{\dot{\Phi}}$, the control target is

$$\hat{F}_{\dot{\Phi}}^{\text{ext}} - F^{\text{des}}(t) = 0 \tag{8}$$

where the predicted model-based force $\hat{F}_{\dot{\Phi}}^{\text{ext}}$ can be derived through the following steps:

1) In the direction of $\dot{\Phi}$, we have

$$\ddot{\Phi} = J_{\dot{\Phi}} \ddot{q} + \dot{J}_{\dot{\Phi}} \dot{q} = 0 \tag{9}$$

2) Represent τ^{ext} in the operational space in the dynamic function, we have

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau - \bar{J}(q)^T \begin{pmatrix} F_{\dot{\Phi}}^{\text{ext}} \\ F_{\dot{x}}^{\text{ext}} \end{pmatrix} \tag{10}$$

3) Transform the above equation, we have

$$\begin{aligned} \hat{F}_{\dot{\Phi}}^{\text{ext}} =& \Lambda_{\dot{\Phi}} J_{\dot{\Phi}} M^{-1}(\tau - g - C\dot{q}) + \Lambda_{\dot{\Phi}} \dot{J}_{\dot{\Phi}} \dot{q} \\ & - \Lambda_{\dot{\Phi}} J_{\dot{\Phi}} M^{-1}(J_{\dot{x}}^T F_{\dot{x}}^{\text{ext}}) \end{aligned} \tag{11}$$

where

$$\Lambda_{\dot{\Phi}} = (J_{\dot{\Phi}} M^{-1} J_{\dot{\Phi}}^T)^{-1} \tag{12}$$

4) Finally, represent τ in the operational space, we have

$$\begin{aligned} \hat{F}_{\dot{\Phi}}^{\text{ext}} =& \Lambda_{\dot{\Phi}} J_{\dot{\Phi}} M^{-1}(\bar{J}(q)^T \begin{pmatrix} F_{\dot{\Phi}}^{\text{ctrl}} \\ F_{\dot{x}}^{\text{ctrl}} \end{pmatrix} - C\dot{q}) \\ & + \Lambda_{\dot{\Phi}} \dot{J}_{\dot{\Phi}} \dot{q} - \Lambda_{\dot{\Phi}} J_{\dot{\Phi}} M^{-1}(J_{\dot{x}}^T F_{\dot{x}}^{\text{ext}}) \end{aligned} \tag{13}$$

Thus, the control force in the constraint force direction can be represented as

$$\begin{aligned} F_{\dot{\Phi}}^{\text{ctrl}} =& + F^{\text{des}}(t) \\ & - \Lambda_{\dot{\Phi}} J_{\dot{\Phi}} M^{-1} J_{\dot{x}}^T F_{\dot{x}}^{\text{ctrl}} \\ & + \Lambda_{\dot{\Phi}} J_{\dot{\Phi}} M^{-1} J_{\dot{x}}^T F_{\dot{x}}^{\text{ext}} \\ & + \Lambda_{\dot{\Phi}} (J_{\dot{\Phi}} M^{-1} C - \dot{J}_{\dot{\Phi}}) \dot{q} \end{aligned} \tag{14}$$

Moreover, to reduce steady-state errors, an additional PI control term is added to the control input:

$$F_{\text{PI}} = -k_{\text{P}} \left(F_{\dot{\Phi}}^{\text{ext}} - F^{\text{des}}(t) \right) - k_{\text{I}} \int \left(F_{\dot{\Phi}}^{\text{ext}} - F^{\text{des}}(t) \right) dt \tag{15}$$

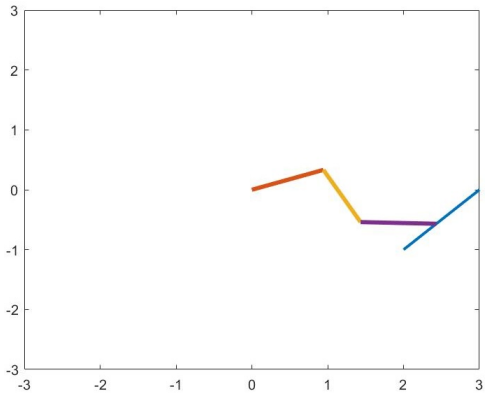

Fig. 1. **An example of the matlab simulation environment.** A three-link robot (red, yellow and purple line) moves on the surface (blue line).

III. Experiment

In this section, we first introduce our environment setup in part A. We test our algorithm on a simplified Matlab environment and then move to a more complex robot arm. Afterwards, we report the implementation details and certain parameters for the algorithm in part B. Finally, we show both the desired force and force measured by sensor which indicates our algorithm can successfully apply given amount of force on the surface.

A. Simulator Setup

1) MATLAB Simulation: We simulate our algorithm in Matlab using a three-link robot. We choose a slope with 45° to apply force on. We assume the robot to be planar. For simplicity, the length of each link is set to 1m and mass is set to 1 kg. Since the end point should be constrained on the surface, the constraint:

$$c(q) = \dot{\Phi}^T \begin{bmatrix} x \\ y \end{bmatrix} = 0 \tag{16}$$

where $\dot{\Phi}$ is the normal direction of the surface. The first and second order time derivative should also be zero.

$$\dot{c}(q) = \dot{\Phi}^T J(q) \dot{q} = 0 \tag{17}$$

$$\ddot{c}(q) = \dot{\Phi}^T [\dot{J}(q)\dot{q} + J(q)\ddot{q}] = 0 \tag{18}$$

From system dynamic Eq. 1 and constraint Eq. 18, we can solve for $\ddot{q}$ and external force N.

$$\begin{cases} M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau + J(q)^T N \\ \dot{\Phi}^T [\dot{J}(q)\dot{q} + J(q)\ddot{q}] = 0 \end{cases} \tag{19}$$

We use Euler method to solve numerically solve the differential equation. The time step is set to 0.001. If the end point is above the surface due to numerical error, then the reative force is 0.

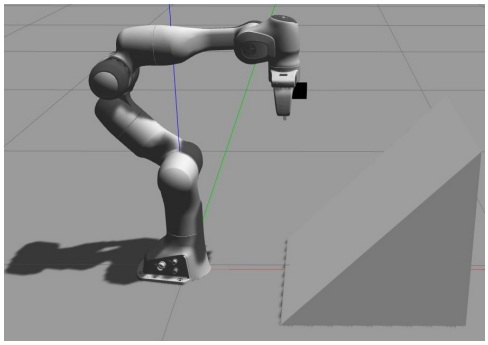

Fig. 2. **An example of the gazebo simulation environment.** A triangular surface if placed 0.8m in front of the robot arm. The length and height of the surface is 0.5m.

2) Gazebo Simulation: In gazebo simulator, we use the 7 DoF Panda Franka arm for our task. Since the original gripper has complex shape and may lead to multiple contact points with the surface, we replace it with a cylinder to simply the contact simulation. The surface is a slope of $45°$. Gazebo contact sensor is attached to the slope to measure the force exerted by the arm. In order to prevent the noise from extreme stiffness, the ODE parameter **soft_erp** is set to 1.0, **kp** is set to $1e^6$ and **min_depth** is set to 0.001. The simulation setup is shown in Fig 2.

B. Implementation Details

For both the gazebo and matlab implementation, K_x and D_x in Eq. 5 are set to 100 and 50 respectively. The proportional and integral gain are set to 0.5 and 0.01 respectively. In order to keep the end-effector perpendicular to the surface, an additional angular torque is added to the output torque. Given the desired rotation R_{des} and current rotation R, we compute the angular difference using Eq. 20.

$$\boldsymbol{\theta}^{\text{diff}} = \left[\frac{(R^T R_{des} - R_{des}^T R)}{2}\right]^{\vee} \tag{20}$$

where $[\cdot]^{\vee}$ is the mapping from a skew-symmetric matrix to a vector. Then the additional angular torque is given by:

$$\tau_{\text{ang}} = K_p J_\omega(q)^T \boldsymbol{\theta}^{\text{diff}} \tag{21}$$

C. Results

We design three trajectories for testing. The details of these trajectories are given below:

1) Constant velocity equals $10^{-3}m/s$ down the slope and the desired force is 20 N.
2) Constant velocity equals $10^{-3}m/s$ down the slope and the desired force is 30 N.
3) Constant acceleration equals $5e{-}4m/s^2$ down the slope and the desired force is 20 N.

The results of three trajectories in Gazebo simulator are summarized in Fig. 4. For position tracking objective, our controller can successfully track the desired trajectory. For trajectory 1 and 3, the mean tracking error is 0.013 and 0.017 m respectively. For trajectory 2, we find that there is constant lag between current position and desired position. The mean tracking error is 0.025m.

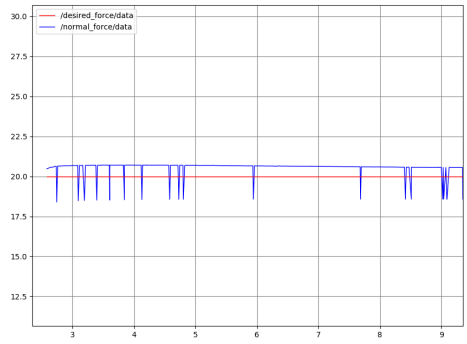

Fig. 3. **An example of sensor noise.** Here we put the end effector on the slope surface. We try to apply a constant 20N force perpendicular to the surface. The robot remains static over the process. The blue curve is the data read from contact sensor.

For the force objective, we find spikes on the blue curve. We ascribe them to the sensor noise. In Fig. 3, we show a static constant force case. Although each joint's effort is stable, there are spikes in the curve, indicating sensor noise. In Fig. 4, the largest error happens in the beginning. This is because the robot arm switches from position control mode to torque control mode. For the contact sensor output, out controller converges to the desired force in around 2 seconds and oscillate near the desired force. The maximum force error for 3 trajectories are 5, 2.8 and 4.2 N, respectively.

The results of matlab simulation is summarized in Fig. 5. For position tracking objective, our controller can also successfully track the desired trajectory, with maximum tracking error of 0.013m. For the force objective, we can also find spikes in the reactive force curve. This is due to the reason that Euler method introduces small error and the end point leaves the surface, resulting in 0 reactive force. The reactive force converges to the desired force in the given time. The final stable error is 2.72N, 2.34N and 2.98N, respectively.

IV. Conclusion

In conclusion, our study underscores the imperative of addressing cleanliness challenges in subway systems. The limitations of manual cleaning methods have prompted our exploration of robotic technologies, particularly focusing on seat cleaning. Our proposed extended Cartesian impedance control algorithm implementation, tested in MATLAB and Gazebo simulations, represents a transformative solution. By ensuring precise and efficient motions, this innovative approach offers a paradigm shift in maintaining subway cleanliness. The envisioned future holds promise for consistently pristine subway environments, providing passengers with a liberating commuting experience, free from the constraints of manual labor.

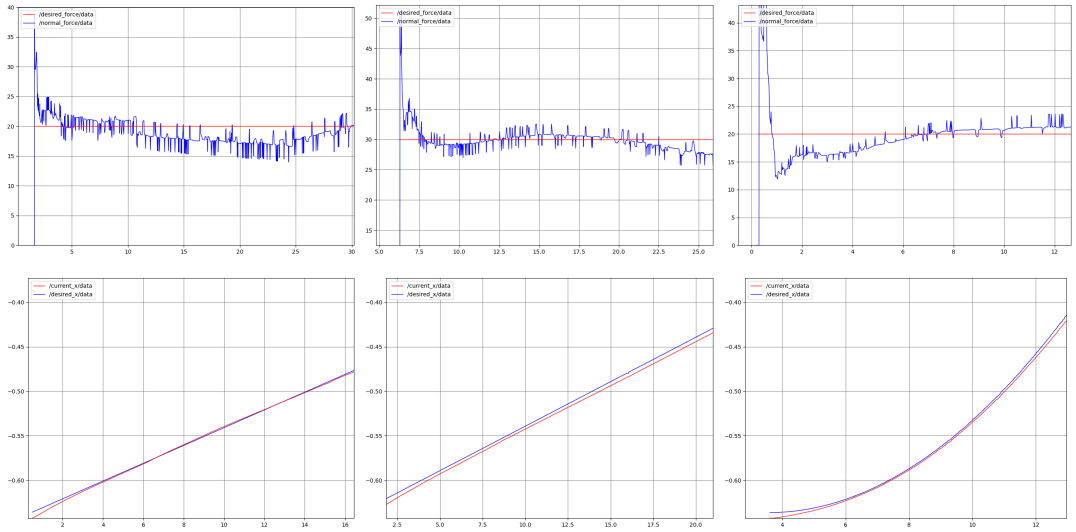

Fig. 4. **Gazebo simulation results.** The left column is the result of trajectory 1. The middle column is the result of trajectory 2. The right column is the result of trajectory 3. The first shows the results of our force controller. The red curve is the desired force and the blue curve is the normal returned by contact sensor. The second row shows the position tracking results on the desired direction. The red curve is the current x position and the blue curve is the desired end effector x position.

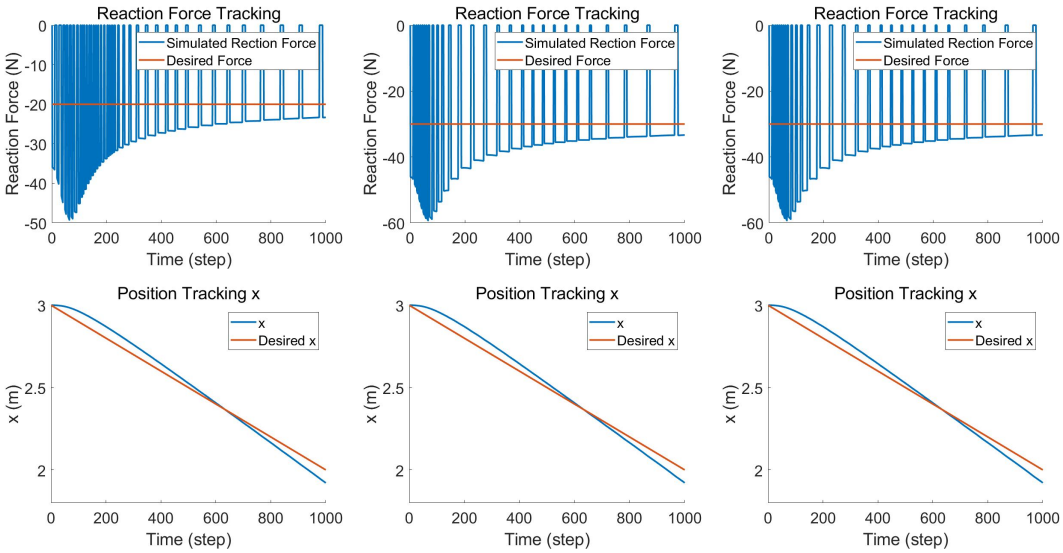

Fig. 5. **MATLAB simulation results.** The left column is the result of trajectory 1. The middle column is the result of trajectory 2. The right column is the result of trajectory 3. The first shows the results of our force controller. The red curve is the desired force and the blue curve is the normal returned by contact sensor. The values are negative because they reflect the forces on the surface, which are on the opposite direction to the forces exerted by the end effector. The second row shows the position tracking results on the desired direction. The red curve is the current x position and the blue curve is the desired end effector x position.

For future work, we would like to implement our algorithms on specifically designed subway car cleaning robot and conduct experiments on actual subway seat surfaces.

ACKNOWLEDGMENT

The authors would like to thank Dr. Michael Posa, our course instructor, for patiently answering our questions related to project implementation, Dr. Nadia Figueroa, for guidance in Franka robot arm related questions and theory derivation, and Wei-cheng Huang, our senior friend, for helping us derive certain formulae.

REFERENCES

[1] N. Hogan, "Impedance control: An approach to manipulation: Part i, part ii, part iii," 1985.

[2] A. Bajo and N. Simaan, "Hybrid motion/force control of multi-backbone continuum robots," *The International journal of robotics research*, vol. 35, no. 4, pp. 422–434, 2016.

[3] M. Iskandar, C. Ott, A. Albu-Schäffer, B. Siciliano, and A. Dietrich, "Hybrid force-impedance control for fast end-effector motions," *IEEE Robotics and Automation Letters*, 2023.