Low-Level Control of Kinova Gen3 Using a Foot Interface: System Control Performance

Xiaolong Han[1]. Supervisor (s): Professor Etienne Burdet, Dorian Verdel, Yanpei Huang

***Abstract*—This article first introduces the design and evaluation of a control strategy for autonomous surgical robots under Low-Level control mode. Subsequently, two control strategies using a Foot Interface for teleoperation are proposed and assessed. Experimental studies evaluate the control accuracy and response speed of the robotic arm's end-effector. Results demonstrate the feasibility of using the Foot Interface for teleoperation in surgical systems. In the Position Mapping mode shows a time delay of approximately 0.015 seconds, while the Velocity Mapping mode shows a time delay of about 0.018 seconds. These results suggest that the Foot Interface likely provides adequate real-time performance and response speed for controlling the robotic arm. The findings offer valuable insights for future development of control strategies in teleoperation robotic-assisted surgical systems.**

***Index Terms*—Microsurgery, Human Machine Interface, Foot Interface, Low-Level Control, Kinova Gen3.**

I. Introduction

THE teleoperation robotic-assisted surgery system is one of the application method of robotic-assisted surgical systems in surgical procedures, which involves remotely controlling the surgical robot through communication networks to achieve real-time remote surgery, treatment, and diagnosis [1]. In teleoperation robotic-assisted surgery systems, surgeons typically utilize a Human-Machine Interface (HMI) to remotely control and switch surgical instruments. For instance, the Da Vinci system [2] utilize the master console for remote operation; the Raven-II laparascopic robot [3] utilizes infrared stereoscopic camera for teleoperation.

April 2020, Huang et al had proposed a HMI (a passive 4-DoF Foot Interface) controlled by foot movements [4]. Operators can utilize this Foot Interface to control robotic surgical instruments through foot movements. During this process, the operator's foot movements are converted into control commands, which theoretically enable the control of the robotic arm's movements in four-degree-of-freedom and be used in microsurgery. In the field of microsurgery, teleoperation robotic-assisted surgery systems manipulate utilize microscope to delicate tissues or fragile structures like blood vessels [5]. Due to the extremely high demands for surgical precision and real-time control during microsurgical procedures, the control accuracy and responsiveness of teleoperation robotic-assisted surgery systems used in microsurgery become critically important. This necessitates higher standards of control precision and real-time performance when using a Foot Interface as the HMI to control the movements of robotic arm.

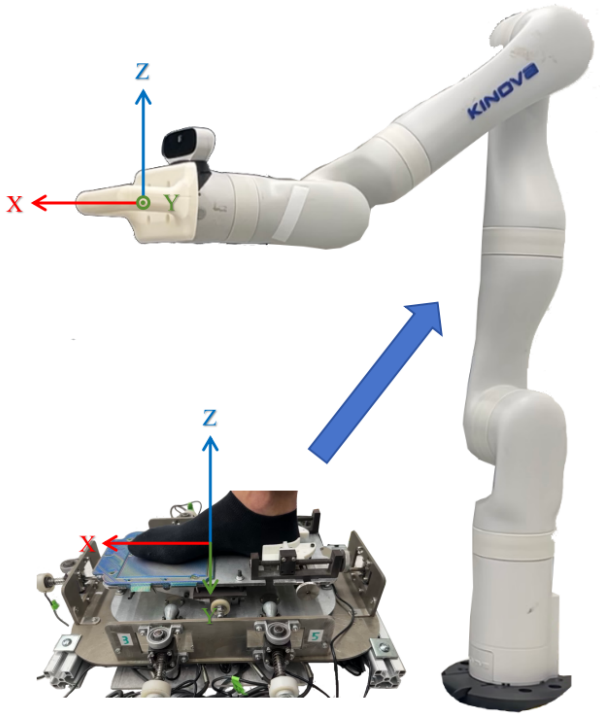

Fig. 1. System diagram, the system consists of a 4-DoF Foot Interface and a 7-DoF Kinova Gen3 robotic arm. The sensor data of Foot Interface communicates with the robotic arm through serial port.

To further evaluate the feasibility of using the Foot Interface for controlling the actual robotic arm, this article focuses on mapping three degrees of freedom data of the 4-DoF Foot Interface into control commands to achieve motion control in Low-Level control of the robotic arm within the X, Y, Z-axis in the Cartesian coordinate system. Low-Level control refers to a method of directly controlling each actuator of a robotic arm during motion, offering faster command transmission and finer control granularity compared to operating the robotic arm through a High-Level control interface. Such as the control response frequency in Low-Kevel control of Kinova Gen3 robotic arm can reach up to 1000 Hz [6]. In the Low-Level control mode, the system can directly adjust parameters such as the speed and torque of each actuator, ensuring that the robot maintains precise movement trajectories and stable operational forces during teleoperation, thereby enhancing surgical safety and outcomes. Furthermore, Low-Level control enables real-time feedback control to follow the surgeon's commands

[1]Xiaolong Han is with the Department of Bioengineering, Imperial College London, SW7 2AZ, United Kingdom, (*email: xiaolong.han23@imperial.ac.uk*)

in real time to respond to tissue deformation and movement at the surgical site [7].

In this article, we first proposed an autonomous surgical robot control strategy to achieve point-to-point autonomous motion control of a 7-DoF Kinova Gen3 robotic arm in Low-Level control mode and evaluated the real-time motion trajectory accuracy. Then, we propose two teleoperation surgical robot control strategies for utilizing a Foot Interface to control the motion of a 7-DoF Kinova Gen3 robotic arm under a Low-Level control mode: Position Mapping and Velocity Mapping. The control performance of these two strategies is systematically evaluated. In the Position Mapping strategy, the real-time position of the Foot Interface is mapped as the target position for the robotic arm's end-effector. In the Velocity Mapping strategy, the real-time position of the Foot Interface is mapped as the linear velocity for the robotic arm's end-effector movement. The main contributions of the paper are as follows:

- Introducing a control strategy of Kinova Gen3 in Low-Level control mode.
- Introducing two control strategies for teleoperation by using Foot Interface, named Position Mapping Mode and Velocity Mapping Mode.

The article is structured as follows: Section II introduces the methodology. Section III details the experiments setting and results. Section IV discusses the findings of experiment results.

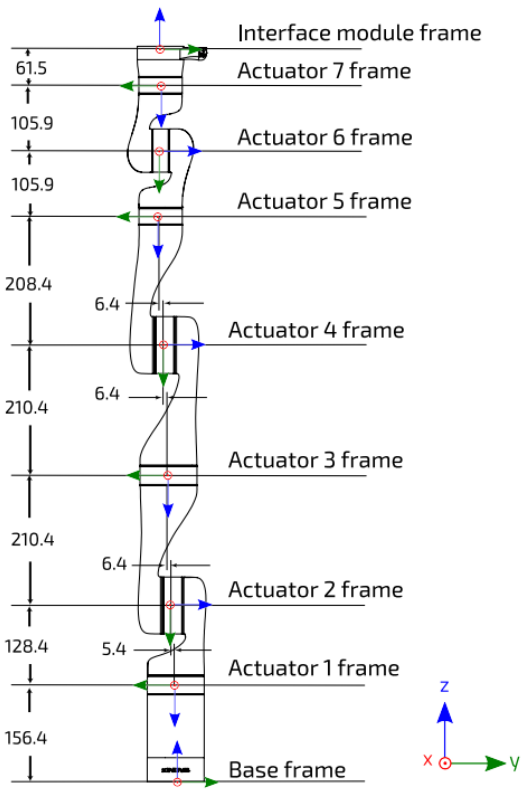

Fig. 2. Reference Frame for each Joint(Actuator) of 7-DoF Kinova Gen3 Robot arm [6].

II. Methodology

A. Kinematics

To implement motion control of the 7-DoF redundant robotic arm under Low-Level control, the forward and inverse kinematic model is first established. Schematic diagram and reference frames for the base and each joints of the 7-DoF Kinova Gen3 as demonstrated in Fig.2.

1) Forward Kinematic: Forward kinematic refers to the mapping from joint space to task space. The objective of forward kinematic is to calculate the pose (position and orientation) of the end-effector by given the joint angles of the robotic arm. In constructing the forward kinematic model of a robotic arm, Homogeneous Transformation matrices and Denavit-Hartenberg(DH) parameters are typically used [8]. The DH parameters define the relative position between each joints of the robotic arm, including the link length a_i, link twist α_i, joint offset d_i, and joint angle θ_i.

For two consecutive joints, the Homogeneous Transformation matrix is defined as follows:

$$ {}^{i-1}\mathbf{T_i} = \begin{bmatrix} \cos(\theta_i) & -\cos(\alpha_i)\sin(\theta_i) & \sin(\alpha_i)\sin(\theta_i) & a_i\cos(\theta_i) \\ \sin(\theta_i) & \cos(\alpha_i)\cos(\theta_i) & -\sin(\alpha_i)\cos(\theta_i) & a_i\sin(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1} $$

The Homogeneous Transformation matrix can be expressed in terms of a rotation matrix and a position vector, where ${}^{i-1}\mathbf{R_i}$ represents the rotational information from frame $i-1$ to frame i, and ${}^{i-1}\mathbf{P_i}$ represents the translational information from frame $i-1$ to frame i.

$$ {}^{i-1}\mathbf{T_i} = \begin{bmatrix} {}^{i-1}\mathbf{R_i} & {}^{i-1}\mathbf{P_i} \\ 0 & 1 \end{bmatrix} \tag{2} $$

Due to the chain rule governing Homogeneous Transformations in robotic arms, the complete transformation from the base to the interface module of 7-DoF Kinova Gen3 can be described as:

$$ {}^{B}\mathbf{T}_{\text{TOOL}} = {}^{B}\mathbf{T_1}\,{}^{1}\mathbf{T_2}\,{}^{2}\mathbf{T_3}\,{}^{3}\mathbf{T_4}\,{}^{4}\mathbf{T_5}\,{}^{5}\mathbf{T_6}\,{}^{6}\mathbf{T_7}\,{}^{7}\mathbf{T}_{\text{TOOL}} \tag{3} $$

where *B* express base frame, *TOOL* express interface module frame. The process as demonstrated in Algorithm 1. DH parameters of 7-DoF Kinova Gen3 as demonstrated in [6]:

2) Inverse Kinematic: Inverse kinematic refers to the mapping from task space to joint space. The objective of inverse kinematic is to calculate the required joint angles of given desired pose (position and orientation). The commonly used methods for solving the inverse kinematics of robotic arms is numerical method [9]. Common numerical methods include the Newton-Raphson method [10] and the Jacobian inverse method [11]. In this article, we utilize a Jacobian-based iterative algorithm to solve the inverse kinematics problem, known as Closed-Loop Inverse Kinematics (CLIK) [12]. Unlike traditional numerical methods based on the Jacobian

Algorithm 1 Forward Kinematics Calculation Process

Require: Joint angles $\theta_1, \theta_2, \ldots, \theta_7$, Denavit-Hartenberg parameters $\mathbf{DH}$

Ensure: End-effector pose $\mathbf{T}_{\text{TOOL}}$

Initialize the overall transformation matrix $\mathbf{T}_{\text{TOOL}} \leftarrow \mathbf{I_{4\times 4}}$ (Identity matrix)
for $i = 1, 2, \ldots, 7$ **do**
  Compute the transformation matrix $\mathbf{T_i^{i-1}}$ using the corresponding DH parameters and joint angle θ_i
  Update the overall transformation matrix $\mathbf{T}_{\text{TOOL}} \leftarrow \mathbf{T}_{\text{TOOL}} \times \mathbf{T_i^{i-1}}$
end for
Extract the end-effector position $\mathbf{P}_{\text{TOOL}} \leftarrow \mathbf{T}_{\text{TOOL}}(1:3, 4)$
Extract the end-effector orientation $\mathbf{R}_{\text{TOOL}} \leftarrow \mathbf{T}_{\text{TOOL}}(1:3, 1:3)$

matrix, the CLIK method incorporates a closed-loop feedback mechanism in each iteration, considering the current error and progressively approaching the desired pose of end-effector.

The fundamental idea of CLIK is to adjust the joint angles so that the actual pose of the robot's end-effector gradually converges to the target pose. The process as demonstrated in Algorithm 2:

Algorithm 2 Closed-Loop Inverse Kinematics (CLIK)

Require: Desired end-effector pose T_d, initial joint configuration θ_0, maximum iterations N, tolerance ϵ

Ensure: Joint angles θ that achieve the desired end-effector pose

Initialize joint configuration $\theta \leftarrow \theta_0$
for $i = 1, 2, \ldots, N$ **do**
  Compute the forward kinematic $T(\theta)$
  Calculate the error $e \leftarrow T_d - T(\theta)$
  if $\|e\| < \epsilon$ **then**
    break
  end if
  Compute the Jacobian $\mathbf{J}(\boldsymbol{\theta})$ and pseudo-inverse $\mathbf{J}^{+}(\boldsymbol{\theta})$
  Compute the joint angle update $\Delta\theta \leftarrow J(\theta)^{\dagger} e$
  Update the joint configuration $\theta \leftarrow \theta + \Delta\theta$
end for

By utilizing the known initial joint configuration and the desired pose of end-effector, this method can effectively solve the inverse kinematics problem. It is worth noting that the solution obtained through the CLIK method is a minimum-norm solution, implying that it selects the joint configuration with the smallest possible change among multiple potential solutions.

3) Jacobian: The Jacobian matrix is used to define the relationship between the movement of each robotic arm joint and the end-effector's motion in space. It characterizes the non-linear connection between the joint velocities of the robotic arm and the linear and angular velocities of the end-effector.

The positional Jacobian is calculated from the partial derivatives of the end-effector's position with respect to the joint variables, while the rotational Jacobian comes from the partial derivatives of the end-effector's rotation matrix with respect to the joint variables. A simplified method of computation involves deriving the Jacobian matrix progressively using the homogeneous transformation matrices.

$$\mathbf{J_i} = \begin{bmatrix} \mathbf{T_0^{i-1}}(1:3,3) \times \left(\mathbf{T_{re}^{b}}(1:3,4) - \mathbf{T_0^{i-1}}(1:3,4)\right) \\ \mathbf{T_0^{i-1}}(1:3,3) \end{bmatrix}, \quad (i \in [1,7]) \tag{4}$$

- $\mathbf{J_i}$: The i-th column of the Jacobian matrix.
- $\mathbf{T_0^{i-1}}(\mathbf{1:3,3})$: The third column of the rotation matrix from the base to the $(i-1)$-th joint.
- $\mathbf{T_{re}^{b}}(\mathbf{1:3,4})$: The position vector of the reference frame, representing the desired position of the end-effector.
- $\mathbf{T_0^{i-1}}(\mathbf{1:3,4})$: The position vector from the base to the $(i-1)$-th joint.

The calculation process of Jacobian matrix as demonstrated in Algorithm 3:

Algorithm 3 Jacobian Calculation using Homogeneous Transformation Matrices

Require: Homogeneous transformation matrices $\mathbf{T_0^1}, \mathbf{T_1^2}, \mathbf{T_2^3}, \mathbf{T_3^4}, \mathbf{T_4^5}, \mathbf{T_5^6}, \mathbf{T_6^7}, \mathbf{T_7^{TOOL}}$

Ensure: Jacobian matrix $\mathbf{J}$

Initialize the Jacobian matrix $\mathbf{J} \leftarrow \mathbf{0}$
for $i = 1, 2, \ldots, 7$ **do**
  Compute the transformation matrix $\mathbf{T_0^i}$ by multiplying the corresponding transformation matrices
  Extract the rotation matrix $\mathbf{R_0^i} \leftarrow \mathbf{T_0^i}(1:3, 1:3)$
  Extract the position vector $\mathbf{P_0^i} \leftarrow \mathbf{T_0^i}(1:3, 4)$
  Compute the linear velocity part: $\mathbf{J_v^i} \leftarrow \mathbf{R_0^i}(1:3,3) \times (\mathbf{P}_{\text{TOOL}} - \mathbf{P_0^i})$
  Compute the angular velocity part: $\mathbf{J_\omega^i} \leftarrow \mathbf{R_0^i}(1:3,3)$
  Update the Jacobian matrix $\mathbf{J}(:, i) \leftarrow [\mathbf{J_v^i}; \mathbf{J_\omega^i}]$
end for

The calculation of the Pseudo-inverse Jacobian matrix is:

$$\mathbf{J}^{+} = \mathbf{J}^{T} \cdot (\mathbf{J} \cdot \mathbf{J}^{T})^{-1} \tag{5}$$

In forward kinematics, the Jacobian matrix is used to determine the end-effector's velocity based on the given joint velocities. Conversely, in inverse kinematics, it is used to calculate the joint velocities needed to achieve a desired end-effector velocity. The relationship between these velocities is described as follows:

$$\mathbf{V}_{\text{end-effector}} = \mathbf{J} \cdot \dot{\boldsymbol{\theta}} \tag{6}$$

where V is the task-space velocity, and $\dot{\boldsymbol{\theta}}$ is the joint-space velocity.

B. Motion Planning

Motion planning is performed to assess the motion control performance of the Kinova Gen3 robotic arm under Low-Level control. This process has Autonomous Motion Planning and Trajectory Tracking.

1) Autonomous Motion Planning: In autonomous motion planning, this article employs the Cartesian-space trajectory planning method. In Cartesian-space, a fifth-order polynomial [13] is used to plan the trajectory, which allows for better control of the end-effector's path in physical space and ensures that the position, velocity, and acceleration vary continuously over time. The standard formula for the fifth-order polynomial is defined as follows, with its first and second derivatives representing velocity and acceleration as functions of time, respectively.

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 \tag{7}$$

Autonomous motion planning process as demonstrated in Algorithm 4.

Algorithm 4 Autonomous motion planning in Cartesian Space using fifth-Order Polynomial

Require: Initial pose $\mathbf{P_0}, R_0$, target pose $\mathbf{P_1}, R_1$, time duration T

Ensure: Generated Autonomous motion trajectory

Define initial pose and target pose:
  $\mathbf{Initial\ pose} = [R_0, P_0]$
  $\mathbf{Target\ pose} = [R_1, P_1]$
Compute the Euler angles from the rotation matrices R_0 and R_1:
  $\mathbf{R} = R_z \times R_y \times R_x$
  Solve for α, β, γ using the inverse calculation of Euler angles.
Set boundary conditions for position, velocity, and acceleration:
  $\mathbf{q(0)}, \dot{\mathbf{q}}(\mathbf{0}), \ddot{\mathbf{q}}(\mathbf{0}), \mathbf{q(T)}, \dot{\mathbf{q}}(\mathbf{T}), \ddot{\mathbf{q}}(\mathbf{T})$
Construct the time matrix $\mathbf{M}$ and boundary condition vector $\mathbf{B}$
Solve for polynomial coefficients $\mathbf{a}$ using:
  $\mathbf{a} = M^{-1} \cdot B$
Generate the trajectory using the polynomial coefficients over the time interval.

After independently planning the trajectories for pose, six sets of fifth-order polynomial coefficients can be obtained. Using the input time parameter t, the corresponding pose of the robotic arm's end-effector can be determined. Then we implement Trajectory Tracking at Low-Level.

2) Trajectory Tracking: Trajectory tracking refers to the process of achieving a planned trajectory of end-effector by controlling the joint angles of a robotic arm. In this process, the target pose is calculated using the results of polynomial interpolation, and the error between the current and target pose is evaluated. The joint angle increments are then computed using the pseudo-inverse Jacobian matrix to update the joint angles, which are subsequently applied to the actuators, thereby achieving real-time motion control. The process as demonstrated in Algorithm 5.

In robotic arm motion control, fixed-frequency control can enhancing control stability by ensuring that the system updates and executes actuator control commands at a constant and sufficiently small time step. The control system periodically sends new angle commands to each joints at a predetermined fixed frequency, thereby mitigating motion instability caused by irregular time intervals or delays. To implement fixed-frequency, we can ensure that the interval between the current angle command and the previous one aligns with the predetermined frequency by comparing the timestamps of when each command is sent. The specific process is as as demonstrated in Fig.3.

Algorithm 5 Real-Time Control Loop for Motion Execution

Require: Initial joint angles $\boldsymbol{\theta_0}$, polynomial coefficients $\mathbf{A}$, and total time T

Ensure: Real-time updated joint commands for desired motion

Initialize $t_{\text{running}} \leftarrow 0$
while $t_{\text{running}} < T$ **do**
  Update t_{running}
  Compute the Jacobian $\mathbf{J}(\boldsymbol{\theta})$ and pseudo-inverse $\mathbf{J}^+(\boldsymbol{\theta})$
  Calculate the current end-effector pose $\mathbf{T}_{\text{current}}$ using forward kinematics
  Compute the target pose $\mathbf{T}_{\text{target}}(t_{\text{running}})$ using polynomial interpolation: $\mathbf{T}_{\text{target}} = \mathbf{a} \cdot \mathbf{M}(t_{\text{running}})$
  Calculate the error $\mathbf{e} \leftarrow \mathbf{T}_{\text{target}} - \mathbf{T}_{\text{current}}$
  Compute joint increments $\Delta\boldsymbol{\theta} \leftarrow \mathbf{J}^+(\boldsymbol{\theta}) \cdot \mathbf{e}$
  Update joint angles $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$
  Apply the new joint angles to the actuators
end while

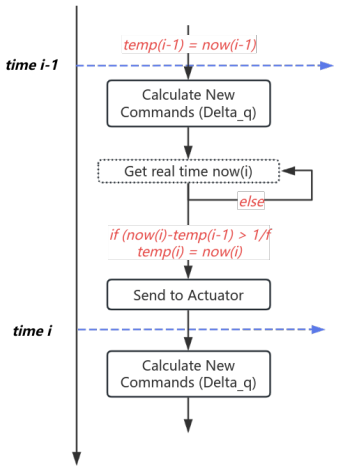

Fig. 3. The process controls command updates by managing time intervals. After each command is sent, the system records the current time and checks the difference from the previous time. If the time difference exceeds the preset interval (e.g., 1/f, where f is the update frequency), the time is updated and the control command is sent; otherwise, the system waits until the condition is met.

C. Foot Interface Control System

The Foot Interface controller is a parallel-serial hybrid mechanism composed of 8 springs and force sensors [4]. In the Low-Level control mode, dual-threading is employed to achieve the control of robotic arm motion by using the Foot

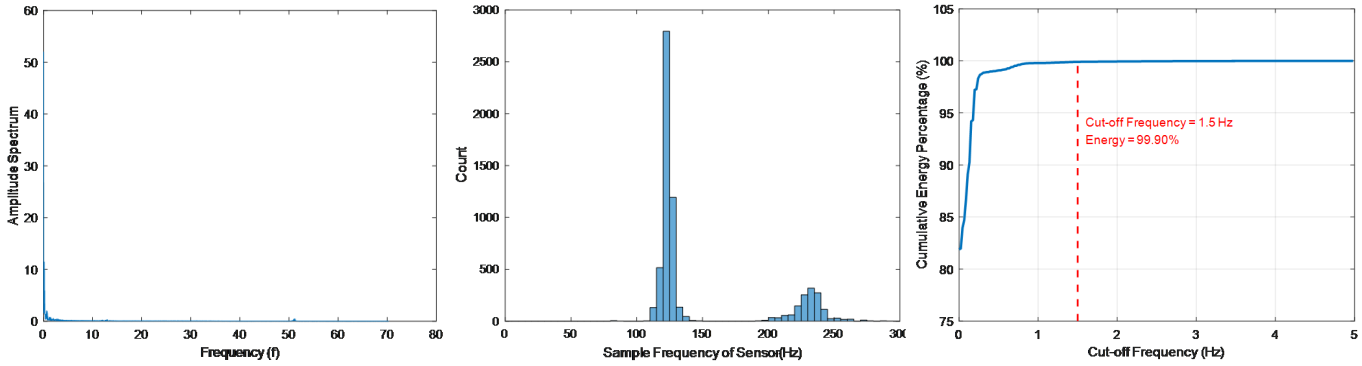

Fig. 4. (a) Amplitude Spectrum, most of the energy is concentrated in low frequencies, with a small amount of high-frequency noise. (b) Sensor sampling frequency distribution histogram, the average sampling frequency of the sensor signal is above 120Hz. (c) Schematic diagram of cumulative signal energy percentage changing with cutoff frequency.

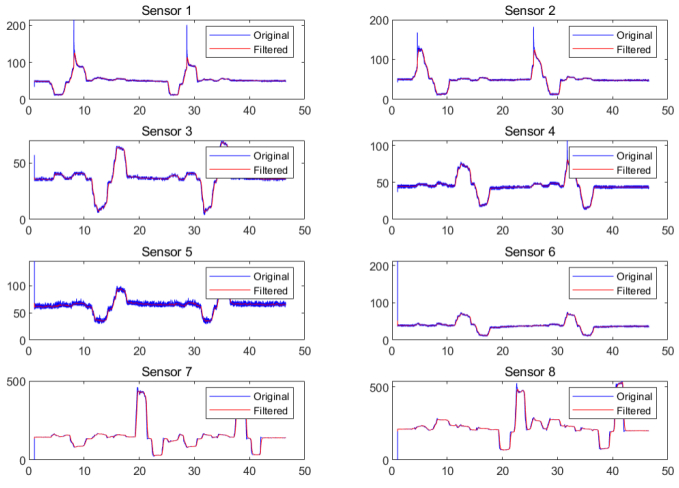

Fig. 5. Comparison of Original and Filtered Signals Across Eight Sensors, Each subplot corresponds to a specific sensor, showcasing the original signal (in blue) and the signal after applying a Low-pass filtering process (in red).

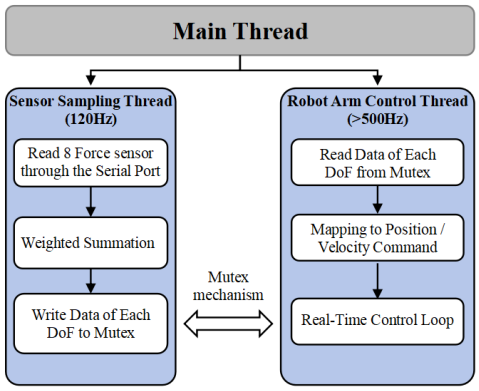

Fig. 6. Framework of dual-threaded control system. Sensor Sampling thread is responsible for acquiring data from the force sensors on the Foot Interface, and to compute changing data of each DoF. Robot Arm Control thread is responsible to control the motion of robot arm. The change data of Foot Interface are mapped to the Position or Velocity data of the end-effector of the robot arm, thereby controlling the real-time movement of the robot arm. The communication between two threads use by mutex mechanism.

Interface, with the implementation of Position and Velocity Mapping control strategies.

1) Data Processing of Force Sensors: Using FFT to analysis the data from the force sensor, as demonstrated in Fig.4(a). Since the sensor signals are direct current signals, a first-order Butterworth low-pass filter is employed for signal processing. As demonstrated in Fig.4(b), the average sampling frequency of the sensor signals exceeds 120Hz, hence the Butterworth filter's sampling frequency is set to 120Hz. Fig.4(c) demonstrates that the cumulative energy percentage of the signal increases with the cut-off frequency, and when the cut-off frequency reaches or exceeds 1.5Hz, the retained cumulative energy percentage surpasses 99.90%. Consequently, the cut-off frequency for the Butterworth filter is set at 1.5Hz. A comparison of the signal data before and after filtering as demonstrated in Fig.5.

The processed data, through weighted summation, can be used to derive the variations corresponding to each degree of freedom. These variations can be mapped to control command of position control and velocity control mode.

2) Dual-threaded Control System Development: The average sampling frequency of Foot interface is large than 120Hz, while the control frequency of the robotic arm under Low-Level control typically exceeds 500Hz. In a mixed system of high-frequency control and low-frequency sensor sampling, a single-threaded control system would restrict the control frequency of the robotic arm to the sampling rate of the Foot interface, significantly reducing the response speed and control accuracy of the robotic arm. Therefore, a dual-threaded control system is employed as the control framework, using mutex mechanism to realize data communication between the two threads. Thread 1 is used to sample sensor data for mapping; Thread 2 is used to control robotic arm movement in real-time. The framework of dual-threaded control system as demonstrated in Fig.6.

3) Position Mapping Control Mode: In the Position mapping control mode, the variations in the each DoF of the Foot Interface are mapped to the target position of the robotic arm's end-effector. The control system first calculates the error between the current position of the end-effector and the target position. Then, using the pseudo-inverse Jacobian matrix to compute the joint angle increments required to correct this error, and subsequently applies the updated joint angles to the actuators, thereby achieving precise motion control of the robotic arm. The process as demonstrated in Algorithm 6.

Algorithm 6 Position Mapping Control Mode

Require: Setting End effector position along 3-axis $\mathbf{v_x}, \mathbf{v_y}, \mathbf{v_z}$, Control Frequency f, Mapping coefficient $\boldsymbol{\gamma}$.

while New sensor position update **do**
  New End effector position $[\mathbf{x}, \mathbf{y}, \mathbf{z}]$ = $\boldsymbol{\gamma}$·New sensor position.
  Compute current forward kinematics $\mathbf{T}_{\text{current}}$
  while $\mathbf{T}_{\text{current}} \neq [\mathbf{x}, \mathbf{y}, \mathbf{z}]^T$ **do**
    Update current joint angles $\boldsymbol{\theta}_{\text{current}}$
    Compute current forward kinematics $\mathbf{T}_{\text{current}}$
    Compute FK of target pose $\mathbf{T}_{\text{target}} \leftarrow \mathbf{T}_{\text{current}} + [\mathbf{v_x}, \mathbf{v_y}, \mathbf{v_z}]^T \cdot \frac{1}{f}$
    Calculate error $\mathbf{e} \leftarrow \mathbf{T}_{\text{target}} - \mathbf{T}_{\text{current}}$
    Compute the Jacobian $\mathbf{J}(\boldsymbol{\theta})$ and pseudo-inverse $\mathbf{J}^{+}(\boldsymbol{\theta})$
    Calculate joint increments $\Delta\boldsymbol{\theta} \leftarrow \mathbf{J} \cdot \mathbf{e}$
    Update joint commands $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$
    Apply the new joint angles to the actuators
    if new sensor position detected **then**
      Break loop and reinitialize control loop
    end if
  end while
end while

4) Velocity Mapping Control Mode: In the Velocity mapping control mode, the variations in the each DoF of the Foot interface are mapped to the linear velocity of the robot end-effector along the x,y,z-axes. The control process is the same as the Position mapping method. The process as demonstrated in Algorithm 7.

Algorithm 7 Velocity Mapping Control Mode

Require: Mapping End effector velocity along 3-axis $\mathbf{v_x}, \mathbf{v_y}, \mathbf{v_z}$, Control Frequency f, Mapping coefficient $\boldsymbol{\gamma}$.

while New sensor position update **do**
  New End effector velocity $\mathbf{v_x}, \mathbf{v_y}, \mathbf{v_z}$ = $\boldsymbol{\gamma}$·New sensor position.
  while true **do**
    Update current joint angles $\boldsymbol{\theta}_{\text{current}}$
    Compute current forward kinematics $\mathbf{T}_{\text{current}}$
    Compute FK of target pose $\mathbf{T}_{\text{target}} \leftarrow \mathbf{T}_{\text{current}} + [\mathbf{v_x}, \mathbf{v_y}, \mathbf{v_z}]^T \cdot \frac{1}{f}$
    Calculate error $\mathbf{e} \leftarrow \mathbf{T}_{\text{target}} - \mathbf{T}_{\text{current}}$
    Compute the Jacobian $\mathbf{J}(\boldsymbol{\theta})$ and pseudo-inverse $\mathbf{J}^{+}(\boldsymbol{\theta})$
    Calculate joint increments $\Delta\boldsymbol{\theta} \leftarrow \mathbf{J} \cdot \mathbf{e}$
    Update joint commands $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$
    Apply the new joint angles to the actuators
    if new sensor position detected **then**
      Break loop and reinitialize control loop
    end if
  end while
end while

III. Results

A. Control Performance of the Robot under Low-Level Control

1) Control Performance Evaluation of Point-to-point Motion: Experiment 1 evaluates the point-to-point motion control performance of the robotic arm's end-effector under Low-Level control. The experiment is structured into six distinct trials, with each trial commencing from a predefined Home Pose ($[\mathbf{0.45}, \mathbf{0}, \mathbf{0.43}, \frac{\pi}{\mathbf{2}}, \mathbf{0}, \frac{\pi}{\mathbf{2}}]$) and proceeding to a specified Target Pose ($[\mathbf{x}, \mathbf{y}, \mathbf{z}, \boldsymbol{\gamma}, \boldsymbol{\beta}, \boldsymbol{\alpha}]$). In each trial, the motion duration from the home pose to the target pose was standardized to 10 seconds. This 10-second interval serves as the temporal parameter for autonomous motion planning. This experiment evaluates the robotic arm's control performance under different target pose under Low-level control. When target pose is ($[\mathbf{0.6}, -\mathbf{0.2}, \mathbf{0.8}, \frac{\pi}{\mathbf{2}}, \frac{\pi}{\mathbf{3}}, \frac{\pi}{\mathbf{3}}]$), the experiment result as demonstrated in Fig.7. The mean tracking error along the X-axis is -0.0001; along the Y-axis, the mean tracking error is 0.0006; and along the Z-axis, the mean tracking error is 0.0005. For the rotational trajectories, the mean tracking errors are 0.0016, -0.0007, and 0.0010, respectively.

2) Performance Evaluation of Fixed-frequency Control: Experiment 2 evaluates the fixed-frequency control under Low-Level control across different response frequencies ($200Hz, 500Hz, 800Hz, 1000Hz$). The experimental results as demonstrated in Fig.8. The distribution of the actual actuator frequencies for the robotic arm is observed to be concentrated around the corresponding control frequencies under different control frequency settings.

B. Control Performance of Foot Interface Control System

Experiment 3 evaluates the response performance and control accuracy of the Foot Interface control system under different strategy. The experimental setup for assessing the performance of the Foot Interface control system includes a Foot Interface and a Kinova Gen3 robotic arm. The Foot Interface is connected to a computer via an Arduino port, the communication baud rate is set to 460800. During experiment, the operator randomly manipulates the Foot Interface in both Position and Velocity Mode to control the motion of the robotic arm.

1) Position Mapping Mode: As demonstrated in Fig.9, it illustrates the positional changes of the robotic arm's end-effector and the Foot Interface along each axis. Overall, the trajectories of both are closely aligned, indicating that the control system accurately maps the positions along each axis. In subfigure 1, a noticeable time lag is observed around the 10-second mark; In subfigure 2, a similar time lag is observed

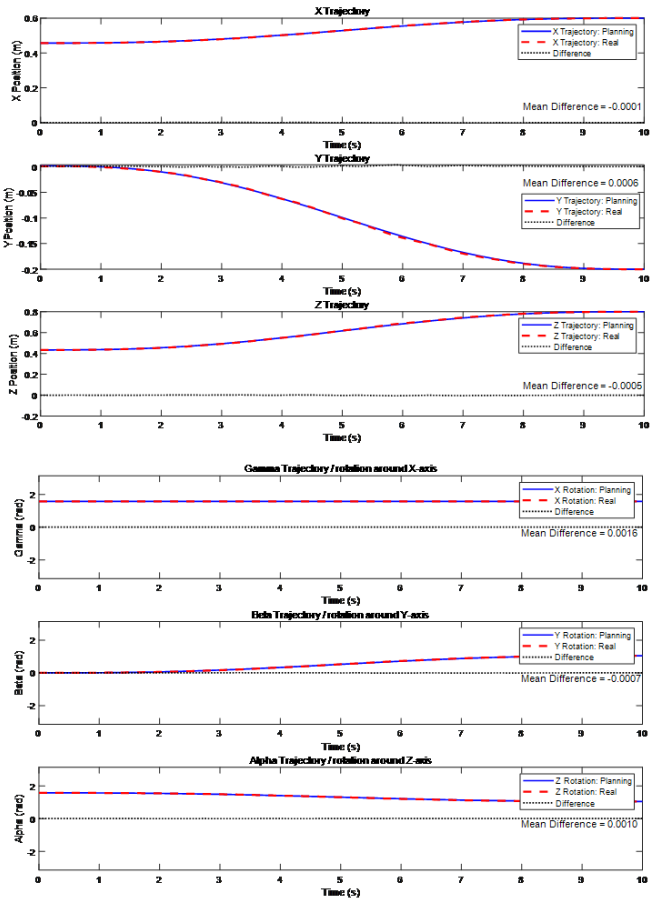

Fig. 7. The trajectory tracking performance of the robotic arm's end-effector is presented, showing the tracking results for three positional coordinates (X, Y, Z) and three rotational angles (Gamma, Beta, Alpha). In each subplot, the blue solid line represents the planned trajectory, the red dashed line represents the actual movement trajectory, and the black dotted line indicates the difference between the these.

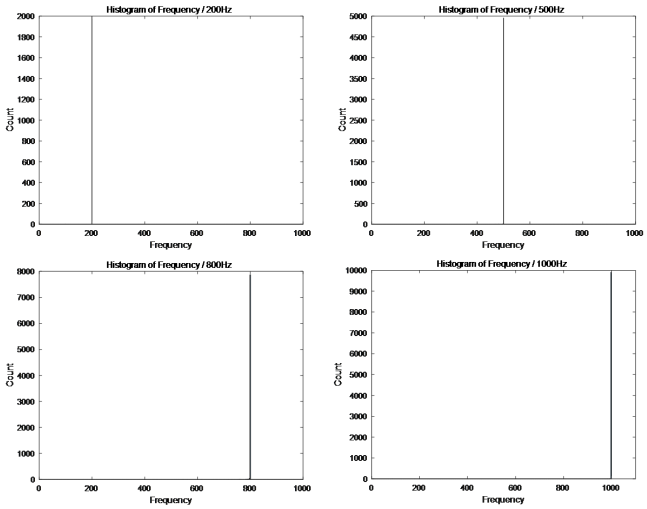

Fig. 8. Histograms of the actual drive frequency distribution of the robotic arm under different control frequencies. The vertical axis represents the count, indicating the number of occurrences of each drive frequency within the respective control frequency setting during movement.

around the 20-second mark; And in subfigure 3, a time lag is

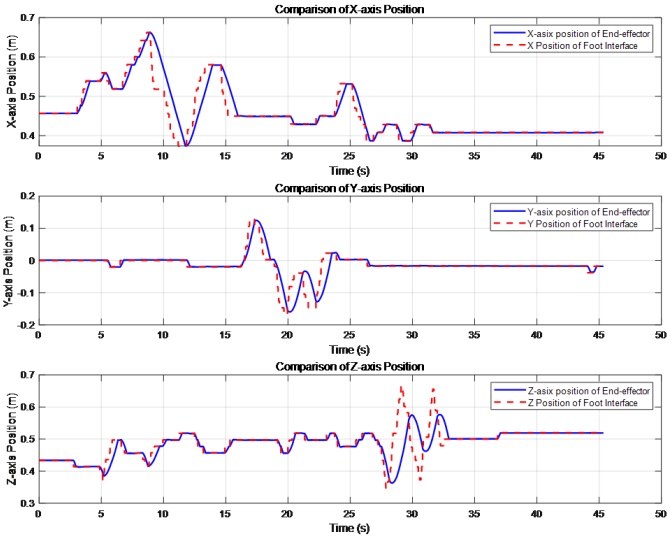

Fig. 9. The figure illustrates the position variations of the robotic arm's end-effector and the Foot Interface along the X, Y, and Z axes over time. The blue solid line represents the actual trajectory of the robotic arm's end-effector, while the red dashed line depicts the position data mapped from the Foot Interface.

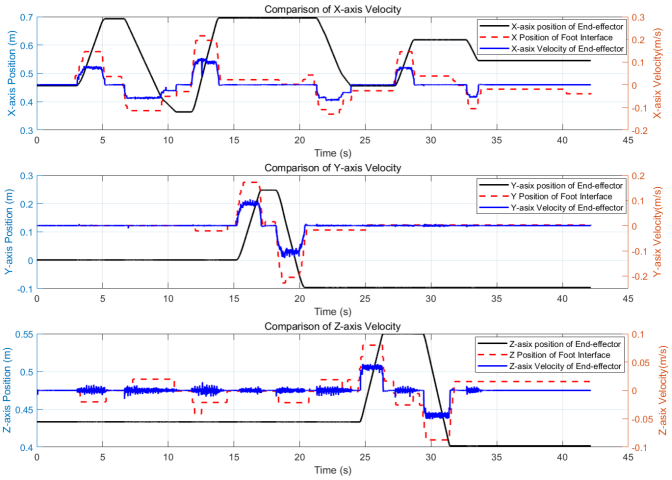

Fig. 10. The figure illustrates the position variations of the robotic arm's end-effector and the Foot Interface along the X, Y, and Z axes over time. The blue solid line represents the actual trajectory of the robotic arm's end-effector, while the red dashed line depicts the position data mapped from the Foot Interface.

observed around the 30-second mark. It can be inferred that these time lags coincide with periods of significant positional changes.

2) Velocity Mapping Mode: As demonstrated in Fig.10, it illustrates the variation process of the robotic arm's end-effector velocity corresponding to the positional changes of the Foot Interface along each axis. And it also presents the curves of the end-effector's positional changes over time along each axis. Overall, the trajectories are closely aligned, indicating that the control system accurately maps the velocity along each axis. However, in subfigure 3, it can be observed that the velocity mapped along the Z-axis exhibits significant noise.

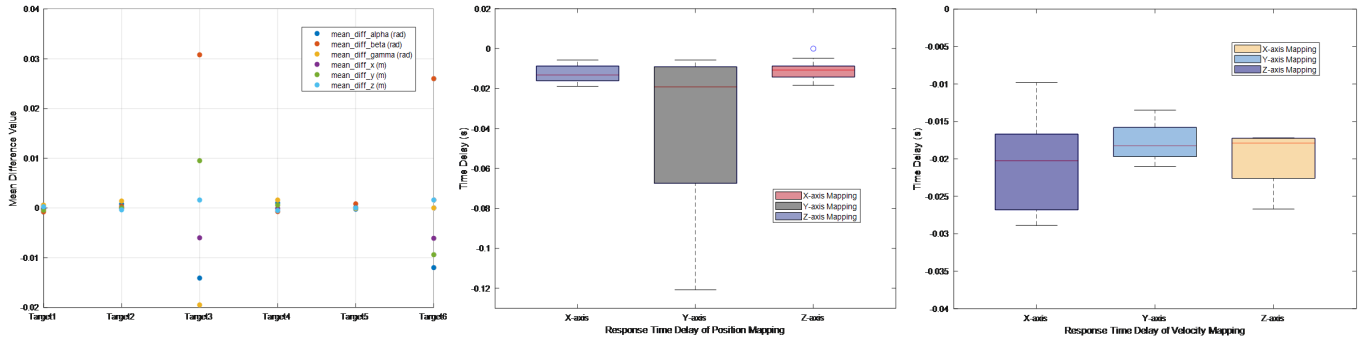

Fig. 11. (a) Mean error of position and orientation during trials with different Target. (b) Box plot of time delay in Position mapping mode. (c) Box plot of time delay in Velocity mapping mode.

IV. Discussion

This article first introduces the design and evaluation of control strategies for autonomous surgical robots operating under the Low-Level Control mode. Subsequently, two control strategies for implementing teleoperation in surgical robotic systems using the Foot Interface are introduced and systematically evaluated. Experimental studies assessed the control accuracy and response speed of the end-effector under the respective control strategies, demonstrating the feasibility of using the Foot Interface in teleoperation surgical robotic systems.

In Experiment 2, we compared the mean difference between the desired and actual positions of the end effector across six different target trials. The results indicate that the average error remained within 0.002 units, suggesting high consistency between the planned and actual motion trajectories, and thus, strong motion accuracy. However, the significant deviation in the β angle for Target 3 and 6 points to potential challenges in controlling specific orientations, particularly around the Y-axis. This deviation suggests the need for further refinement in control algorithms and system calibration.

In Experiment 3, we observed that the time delay in the axis data mapping process of the Foot Interface under the Position Mapping mode is approximately 0.015 seconds, which as demonstrated in Fig.11(b). The time delays in the X-axis and Z-axis mappings exhibit small interquartile ranges and minimal deviations from the mean, indicating that the delay is relatively stable. However, the significant fluctuations in the Y-axis delay suggest potential control issues in this direction, and further optimization or system adjustments might be necessary to improve the robotic arm's response performance along the Y-axis. In the Velocity Mapping mode, the time delay observed is approximately 0.018 seconds, as demonstrated in Fig.11(c). And the time delay of data mapping in the each axis directions is relatively stable. The time delay in both control strategies suggests that the Foot Interface is likely to provide adequate real-time performance and response speed for controlling a robotic arm.

In future work, the degree of freedom used to control the movement of the robot arm will be further increased to achieve real-time control of each angle in orientation. To further prove the feasibility of using the Foot Interface as an HMI for the teleoperation surgery robot system and the feasibility of acting on the microsurgery field.

Acknowledgments

Thanks to Dr.Verdel Dorian and Dr.Yanpei Huang for their guidance in the article.

References

[1] S. Avgousti, E. G. Christoforou, A. S. Panayides, et al., "Medical telerobotic systems: current status and future trends," *Biomedical Engineering Online*, vol. 15, pp. 1–44, 2016.

[2] J. Bodner, H. Wykypiel, G. Wetscher, et al., "First experiences with the da Vinci operating robot in thoracic surgery," *European Journal of Cardio-thoracic Surgery*, vol. 25, no. 5, pp. 844–851, 2004.

[3] F. Despinoy, A. Sánchez, N. Zemiti, et al., "Comparative assessment of a novel optical human-machine interface for laparoscopic telesurgery," in *Proc. 5th Int. Conf. Information Processing in Computer-Assisted Interventions (IPCAI)*, Fukuoka, Japan, Jun. 2014, pp. 21–30.

[4] Y. Huang, E. Burdet, L. Cao, et al., "A subject-specific four-degree-of-freedom foot interface to control a surgical robot," *IEEE/ASME Trans. Mechatronics*, vol. 25, no. 2, pp. 951–963, 2020.

[5] C. J. Payne, K. Vyas, D. Bautista-Salinas, et al., "Shared-control robots," *Neurosurgical Robotics*, pp. 63–79, 2021.

[6] "Discover our gen3 robots," Kinova. [Online]. Available: https://www.kinovarobotics.com/product/gen3-robots. Accessed: Feb. 22, 2024.

[7] L. S. Mattos, D. G. Caldwell, G. Peretti, et al., "Microsurgery robots: addressing the needs of high-precision surgical interventions," *Swiss Medical Weekly*, vol. 146, no. 4344, pp. w14375, 2016.

[8] J. Denavit and R. S. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," *J. Appl. Mech.*, vol. 22, no. 2, pp. 215–221, 1955.

[9] A. Aristidou and J. Lasenby, "Inverse kinematics: a review of existing techniques and introduction of a new fast iterative solver," *The Visual Computer*, vol. 25, no. 5-7, pp. 347–357, 2009.

[10] R. Fletcher, *Practical Methods of Optimization*, 2nd ed. Chichester, U.K.: John Wiley Sons, 2000.

[11] H. Das, J. E. Slotine, and T. B. Sheridan, "Inverse kinematic algorithms for redundant systems," in *Proc. IEEE Int. Conf. Robotics and Automation*, Philadelphia, PA, USA, 1988, pp. 43–48.

[12] J. Carpentier, G. Saurel, G. Buondonno, et al., "The Pinocchio C++ library: A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives," in *Proc. IEEE/SICE Int. Symp. System Integration (SII)*, Paris, France, 2019, pp. 614–619.

[13] Y. Guan, K. Yokoi, O. Stasse, et al., "On robotic trajectory planning using polynomial interpolations," in *Proc. IEEE Int. Conf. Robotics and Biomimetics (ROBIO)*, Shenyang, China, 2005, pp. 111–116.