Scilight

Journal of Artificial Intelligence for Automation
https://www.sciltp.com/journals/jaia

Article

Closed-Loop Co-Design of Motors, Motions, and Feedback Control for Robotic Manipulators

Jui-Te Lin [1], Zehui Lu [2] and Yebin Wang [3,*]

[1] Berkshire Grey Inc., Bedford, MA 01730, USA
[2] Lucid Motors, Newark, CA 94560, USA
[3] Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA 02139, USA
* Correspondence: yebinwang@ieee.org

How To Cite: Lin, J.-T.; Lu, Z.; Wang, Y. Closed-Loop Co-Design of Motors, Motions, and Feedback Control for Robotic Manipulators. *Journal of Artificial Intelligence for Automation* **2026**, *1*(2), 8. https://doi.org/10.53941/jaia.2026.100008

Received: 8 February 2026
Revised: 24 March 2026
Accepted: 7 April 2026
Published: 2 June 2026

Abstract: The co-design paradigm claims substantial advantages to hardware and control system design by addressing multidisciplinary challenges within a unified framework. Established co-design frameworks for robot manipulators have predominantly focused on two components: motor/arm design and trajectory optimization, which inadequately address real-world disturbances and model uncertainties and thus render suboptimal design and closed-loop system performance. This paper proposes a closed-loop co-design (CLCD) framework to jointly determine motors, motions, and a feedback controller, where the controller comprises a reinforcement learning (RL)-based compensator and a classic proportional-derivative controller for trajectory tracking. Simulation is performed to validate (1) the effectiveness of the proposed CLCD framework to attenuate the sim-2-real gap, (2) the viability of incorporating an RL-based controller into the CLCD for flexible and efficient synthesis of control policy, and (3) the scalability of the CLCD by applying it to perform co-design for 12 and 120 tasks.

Keywords: industrial robots; optimization and optimal control; reinforcement learning; methods and tools for robot system design

1. Introduction

Co-design concept, jointly optimizing both hardware and software components [1], has emerged as an important research direction across various domains, to name a few, mechatronics [2], aerospace [3], robotics [4–6], etc. Its inherently multidisciplinary nature often leads to large-scale optimization problems involving hundreds of thousands of variables. The computational co-design method provides a powerful framework for optimal design of hardware and software for higher performance [7]. Computational co-design is largely carried out in sequential, nested, and simultaneous manner [8,9]. Nonetheless, it presents a spectrum of challenges ranging from the intricacies inherent in design problems that span multiple disciplines to the complexity of large-scale nonlinear programming. One effective strategy for managing the complexity involves decomposing a complex problem into multiple more tractable subproblems. Furthermore, employing efficient gradient computation methods becomes essential when solving large-scale optimization problems [10]. In this work, we propose a unified computational framework for the optimization of both hardware (motor design) and software (motion planning and feedback control policy), demonstrated on an industrial robot.

1.1. Robot Co-Design

Robot co-design has demonstrated superior performance across a range of robotic systems such as manipulators [6,11], robotic hands [12], legged robots [7,13], and cylindrical robots [14]. Most existing work focus on optimizing the plant parameters and robot motions, overlooking the uncertainties arising from the deployment of the robots in real-world environments. Some recent work try to incorporate uncertainties in the co-design process

to close the sim-2-real gap. For example, work [15] utilizes genetic algorithms to explore the design space and deep reinforcement learning (RL) to optimize control policies for adapting to diverse environments. Chen et al. [12] explore a unified RL approach, integrating both hardware and computational policies into a joint policy. A policy iteration method for H_∞ co-design with mismatched uncertainties was proposed, and the performance improvement was demonstrated in [16]. However, the emphasis of H_∞ control on worst-case performance can lead to conservative designs and reduced system efficiency [17]. Bravo-Palacios et al. [18] introduced feedback controllers into co-design for robustness to disturbances in the nominal trajectory. Their work simultaneously optimizes robot morphology, trajectory, and feedback controller in a single optimization problem, resulting in an exceedingly large design space.

1.2. Trajectory Tracking

Numerous trajectory tracking control methods have been proposed for precise positioning in industrial automation [19, 20]. Among them, the most prominent ones are Proportional-Integral-Derivative (PID) controller [21,22] and the Linear Quadratic Regulator (LQR) [23,24]. Neural network-based control methods have been studied to enhance system robustness under disturbances [25,26]. As robots are often subject to constraints arising from joint limits, task-specific constraints, and motor specifications, model predictive control (MPC) has gained attention [27,28]. Furthermore, hybrid methods that combine RL with classic controllers have been examined and have demonstrated promising performance [29]. This approach provides a way to self-adjust and re-optimize the controller over time based on operational data, particularly as the robot's accuracy decreases due to hardware degradation.

1.3. Contributions

We propose a three-step co-design framework to perform closed-loop co-design (CLCD) of motor, motion (trajectory), and feedback control policy of an n degree-of-freedom (DOF) robotic manipulator. This is different from [30], which performs motor design and trajectory optimization (TO) only.

To enhance the learning efficiency as well as the robustness to disturbances, we parameterize the feedback controller as a combination of an RL-based reference compensator and a classical model-based controller. We further develop an efficient method for updating and transferring the actor and critic parameters in the RL module throughout the co-design loop. Additionally, unlike the work [30], which evaluates motor designs based solely on open-loop optimal trajectories, we use closed-loop trajectories that compensate for unknown disturbances, improving accuracy and numerical stability in motor design evaluation. Simulation studies for co-design across 12 and 120 tasks with varying initial and final states and loads demonstrate the scalability of the proposed CLCD framework, as well as the superior trajectory-tracking performance of the resulting closed-loop system compared with a traditional open-loop co-design (OLCD) approach.

2. System Modeling and Problem Formulation

This paper investigates robot co-design problem for an n-DOF robotic manipulator, each joint of which is driven by a surface-mount permanent magnet synchronous motor (SPMSM). The manipulator shall perform a total number of K tasks $\mathbf{\Gamma} \triangleq \{\Gamma_1, \Gamma_2, \cdots, \Gamma_K\}$ repetitively. Each task in $\mathbf{\Gamma}$ is defined by a desired task time t_f, an initial state $(\boldsymbol{\theta}(0), \dot{\boldsymbol{\theta}}(0))$, a desired terminal state $(\boldsymbol{\theta}(t_\mathrm{f}), \dot{\boldsymbol{\theta}}(t_\mathrm{f}))$, and a payload mass, where $\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}$ are the vectors representing the angular positions and velocities of all joints, respectively. The notation used throughout the paper is summarized in Table 1.

2.1. Motor-Robot System Modeling

The multidisciplinary modeling of the motor-robot system in [30, 31] is briefly introduced below to be self-contained. At its core is to establish the analytical mappings from motor design to the motor-robot dynamic model and cost functions.

Table 1. Table of notation.

Symbol	Description
n	Number of degrees of freedom (DOF) of the manipulator
K	Total number of tasks in the task set
$\boldsymbol{\Gamma}$	Set of manipulation tasks
t_{f}	Task final time
$\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \ddot{\boldsymbol{\theta}}$	Joint position, velocity, and acceleration vectors
$\boldsymbol{x}$	System state
$\boldsymbol{x}_r$	Robot state
$\boldsymbol{u}$	Motor voltage input vector
$i_{\mathrm{d}}, i_{\mathrm{q}}$	d-axis and q-axis currents of the SPMSM
$u_{\mathrm{d}}, u_{\mathrm{q}}$	d-axis and q-axis voltages of the SPMSM
ω	Rotor angular speed
τ	Motor torque
$\boldsymbol{\tau}_{\mathrm{L}}$	Joint load torque vector
P	Number of pole pairs
R	Resistance
$L_{\mathrm{d}}, L_{\mathrm{q}}$	d-axis and q-axis inductances
Φ_{m}	Magnet flux
J	Rotor inertia
Z	Gear ratio
η	Efficiency function
$\boldsymbol{\beta}_i$	Design vector of the ith motor
$\boldsymbol{\beta}$	Set of motor design vector
h_{m}	Magnet height
r_{so}	Outer radius of stator
r_{ro}	Outer radius of rotor
w_{tooth}	Tooth width
l	Axial core length
h_{sy}	Stator yoke height
b_0	Slot opening
$\boldsymbol{M}$	Manipulator inertia matrix
$\boldsymbol{C}$	Coriolis/centrifugal matrix
$\boldsymbol{G}$	Gravity vector
J_{cd}	Co-design objective
J_{TO}	Trajectory optimization objective
$\boldsymbol{Q}$	Positive definite weighting matrix
N	Number of discretization steps in each task
$\boldsymbol{\theta}_{\mathrm{ref}}, \dot{\boldsymbol{\theta}}_{\mathrm{ref}}$	Reference joint position and velocity trajectories
$\hat{\boldsymbol{x}}_r$	Reference input to the model-based closed-loop controller
$\boldsymbol{e}$	Joint position tracking error
$\dot{\boldsymbol{e}}$	Joint velocity tracking error
$\boldsymbol{e}_x$	Tracking error in reduced robot state
$\boldsymbol{\pi}$	RL-based compensator policy
$\boldsymbol{p}_\pi$	Parameters of the RL policy
$\boldsymbol{s}$	RL state
$\boldsymbol{a}$	RL action
R_π	RL reward
w_1, w_2, w_3	Positive weights in the RL reward function
$\boldsymbol{\tau}_{\mathrm{ff}}$	Feedforward torque command
$\boldsymbol{\tau}_{\mathrm{pd}}$	PD feedback torque command
q_i	Torque-speed probability density for the ith motor
$\boldsymbol{q}$	Vector of torque-speed probability densities for all motors
$\boldsymbol{g}$	Motor design constraints
ϵ	Convergence tolerance for the co-design loop
err	Motor design update error

2.1.1. Spmsm Design Parameterization and Dynamics

Motor design parameterization is illustrated by Figure 1. For the motor at the ith joint, its design variables are aggregated below:

$$\boldsymbol{\beta}_i \triangleq \begin{bmatrix} h_{\mathrm{m},i} & r_{\mathrm{so},i} & r_{\mathrm{ro},i} & w_{\mathrm{tooth},i} & l_i & h_{\mathrm{sy},i} & b_{0,i} \end{bmatrix} \in \mathbb{R}^7,$$

where $h_{\mathrm{m},i}$ is the height of magnet, $r_{\mathrm{so},i}$ is the outer radius of stator, $r_{\mathrm{ro},i}$ is the outer radius of rotor, $w_{\mathrm{tooth},i}$ is the width of tooth , l_i is the axial length of core, $h_{\mathrm{sy},i}$ is the stator yoke, $b_{0,i}$ is the slot opening and i denotes the i-th motor.

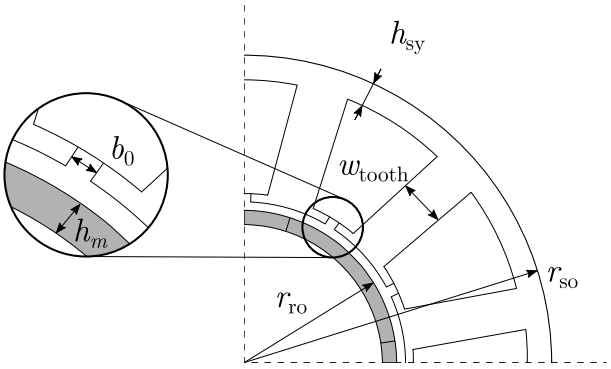

Figure 1. The SPMSM's geometry in the cross-sectional view [30].

The SPMSM dynamics can be modeled as follows [30]:

$$\dot{i}_{\mathrm{d}} = -Ri_{\mathrm{d}}/L_{\mathrm{d}} + P\omega i_{\mathrm{q}} + u_{\mathrm{d}}/L_{\mathrm{d}}, \tag{1a}$$

$$\dot{i}_{\mathrm{q}} = -Ri_{\mathrm{q}}/L_{\mathrm{q}} - P\omega(i_{\mathrm{d}} + \Phi_{\mathrm{m}}/L_{\mathrm{q}}) + u_{\mathrm{q}}/L_{\mathrm{q}}, \tag{1b}$$

$$\dot{\omega} = (1.5P\Phi_{\mathrm{m}} i_{\mathrm{q}} - \tau_{\mathrm{L}}/Z)/J, \tag{1c}$$

where i_{d} and i_{q} represent the currents in d-axis and q-axis, respectively; ω is the rotor speed; u_{d} and u_{q} are the voltage applied to d-axis and q-axis, respectively; $\tau \triangleq 1.5P\Phi_{\mathrm{m}} i_{\mathrm{q}}$ is the torque generated by the motor; τ_{L} is the load torque applied on the rotor shaft. P denotes the number of pole pairs, R the resistance, L_{d} and L_{q} the inductance in d-axis and q-axis, respectively, Φ_{m} the magnetic flux, J is the rotor inertia, and Z is the gearbox ratio. For SPMSM, $L_{\mathrm{d}} = L_{\mathrm{q}}$.

Remark 1. *Assume an ideal harmonics drive with a gear ratio Z. Then, we have $\omega = Z\dot{\theta}$ where θ is the joint angle.*

Remark 2. *Work [30, 31] show that for an SPMSM with design $\boldsymbol{\beta}_i$, both the model parameters $R, L_{\mathrm{d}}, L_{\mathrm{q}}, \Phi_{\mathrm{m}}, J$ and efficiency $\eta(\tau, \omega, \boldsymbol{\beta}_i)$ can be represented as analytical functions of design variables $\boldsymbol{\beta}_i$.*

Denote $\boldsymbol{\beta} \triangleq \begin{bmatrix} \boldsymbol{\beta}_1^\top & \cdots & \boldsymbol{\beta}_n^\top \end{bmatrix}^\top \in \mathbb{R}^{7n}$ as the design parameters of n motors for the manipulator. Boldface current and voltage variables (e.g., $\boldsymbol{i}_{\mathrm{d}}, \boldsymbol{u}_{\mathrm{d}}, \boldsymbol{\tau}_{\mathrm{L}}, \boldsymbol{\eta}$) represent vectors corresponding to all n motors of the manipulator.

2.1.2. Robot Manipulator Dynamics

The n-DOF industrial manipulator dynamics can be modeled as follows:

$$\boldsymbol{M}(\boldsymbol{\theta}, \boldsymbol{\beta})\ddot{\boldsymbol{\theta}} + \boldsymbol{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \boldsymbol{\beta})\dot{\boldsymbol{\theta}} + \boldsymbol{G}(\boldsymbol{\theta}, \boldsymbol{\beta}) = \boldsymbol{\tau}_{\mathrm{L}}, \tag{2}$$

where $\ddot{\boldsymbol{\theta}} \in \mathbb{R}^n$ is the vector of angular accelerations; $\boldsymbol{\tau}_{\mathrm{L}} \in \mathbb{R}^n$ is the vector of torques applied on joints; $\boldsymbol{M}, \boldsymbol{G}$ are the inertia matrix and gravitation forces, respectively; and $\boldsymbol{C}$ is related to the Coriolis force.

Remark 3. *Matrices $\boldsymbol{M}, \boldsymbol{C}, \boldsymbol{G}$ in (2) are analytical functions of motor design $\boldsymbol{\beta}$. Considering Remark 2, the dynamical models of both motor and robot (1) and (2) can be implemented using CasADi [32] to efficiently compute the gradients using automatic symbolic differentiation for co-design optimization.*

2.2. Robot Co-Design Problems

Denote $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{i}_{\mathrm{d}}^\top & \boldsymbol{i}_{\mathrm{q}}^\top & \boldsymbol{\theta}^\top & \dot{\boldsymbol{\theta}}^\top \end{bmatrix}^\top$ as the system state; $\boldsymbol{u} = \begin{bmatrix} \boldsymbol{u}_{\mathrm{d}}^\top & \boldsymbol{u}_{\mathrm{q}}^\top \end{bmatrix}^\top$ the control input. The robot co-design can be formulated as a single objective optimization problem, e.g.,

Problem 1. *Given the motor-robot dynamic model* (1) *and* (2) *of an* n*-DOF manipulator and the task set* $\mathbf{\Gamma}$*, determine the optimal motor design* $\boldsymbol{\beta}^*$ *and optimal control* $\boldsymbol{u}^*$ *which minimize the co-design objective function* $J_{\mathrm{cd}}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\beta}, \mathbf{\Gamma})$.

Problem 1 is challenging to solve because the representation of function J_{cd} is contingent on the control system architecture and controller parameterization. A multitude of works [6, 7, 13] assume open-loop optimal control system architecture and thus the controller is parameterized by reference trajectories $\boldsymbol{x}_{\mathrm{ref}}, \boldsymbol{u}_{\mathrm{ref}}$. The resultant co-design problem is tackled by solving two subproblems: design problem for $\boldsymbol{\beta}$ and TO problem for $\boldsymbol{x}^*, \boldsymbol{u}^*$. We argue that this treatment renders unsatisfactory system performance because the control system architecture and parameterization are unrealistic. Specifically, the prevailing control system architecture of industrial manipulators is way more sophisticated and hierarchical, involving TO, feedforward controller, and feedback control, etc., where each module addresses distinctive needs. For example, the feedback control module is to attenuate uncertainties; without it, the robustness of the resultant control system can, at best, be accounted by TO, which was not invented for this purpose.

This paper investigates the following closed-loop robot co-design problem by incorporating feedback controller, which admits a parameterized representation $\boldsymbol{\pi}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \boldsymbol{e}, \dot{\boldsymbol{e}}, \boldsymbol{p}_{\boldsymbol{\pi}})$, where $\boldsymbol{e}$ and $\dot{\boldsymbol{e}}$ are the tracking error of the angular positions of joints and the time derivative, respectively, and $\boldsymbol{p}_{\boldsymbol{\pi}}$ is the vector of the feedback control policy parameters.

Problem 2. *Given an* n*-DOF manipulator, the motor-robot dynamic model* (1) *and* (2)*, and task set* $\mathbf{\Gamma}$*, determine the optimal motor design* $\boldsymbol{\beta}^*$*, optimal joint trajectories* $(\boldsymbol{\theta}^*(t), \dot{\boldsymbol{\theta}}^*(t))$ *and control* $\boldsymbol{u}^*(t)$*, as well as the optimal feedback control policy* $\boldsymbol{\pi}^*(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \boldsymbol{e}, \dot{\boldsymbol{e}}, \boldsymbol{p}_{\boldsymbol{\pi}})$ *to minimize* $J_{\mathrm{cd}}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \boldsymbol{e}, \dot{\boldsymbol{e}}, \boldsymbol{p}_{\boldsymbol{\pi}}, \boldsymbol{\beta}, \mathbf{\Gamma})$.

Remark 4. *Problem 2 comprises three coupled sub-problems: TO, feedback control synthesis, and motor design optimization. It remains challenging because* $J_{\mathrm{cd}}(\cdot)$ *is not well-defined without selecting the control system architecture.*

3. Closed-Loop Co-Design Methodology

Aiming to solve Problem 2, this section presents the CLCD framework, as illustrated in Figure 2, where the control system architecture integrates a hierarchical model-based controller and a learning-based controller. A sequential and iterative workflow is adopted to address the multidisciplinary optimization problem by solving TO, feedback control policy synthesis, and motor design separately within each iteration of the co-design process. The new design obtained from motor design optimization is fed back to update the motor-robot dynamic model for both TO and control policy synthesis.

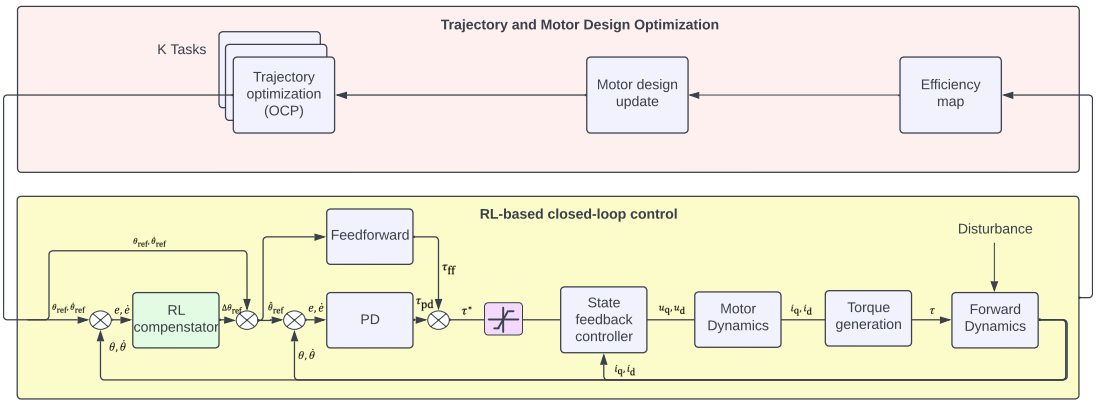

Figure 2. Overview of the proposed CLCD framework. The framework iteratively integrates TO, RL-augmented feedback control, and motor design optimization. A classical model-based feedback controller works with an RL-based reference compensator to mitigate disturbances and modeling uncertainties encountered in real-world operation.

In the TO stage, an ideal motor–robot model is employed, in which motor dynamics are explicitly included and the robot inertia matrices are updated according to the current motor design $\boldsymbol{\beta}$, while uncertainties such as friction are neglected. With motor dynamics incorporated, the control inputs are the voltages supplied to the motors. The TO stage computes $\boldsymbol{u}^*$ and $\boldsymbol{\theta}^*, \dot{\boldsymbol{\theta}}^*$ over N time steps for each task. The optimal trajectories $\boldsymbol{\theta}^*, \dot{\boldsymbol{\theta}}^*$ serve as reference trajectories $\boldsymbol{\theta}_{\mathrm{ref}}, \dot{\boldsymbol{\theta}}_{\mathrm{ref}}$ for the downstreaming closed-loop feedback controller.

Next, a feedback control policy is synthesized using a non-ideal motor–robot model that includes disturbances. The resulting policy, $\boldsymbol{\pi}^*(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \boldsymbol{e}, \dot{\boldsymbol{e}}, \boldsymbol{p}_{\boldsymbol{\pi}})$, acts as a reference compensator that augments the model-based controller to mitigate disturbances during trajectory tracking. Then the motor current controller will track the desired torque $\boldsymbol{\tau}^*$. Using the closed-loop operation trajectories of actual torque $\boldsymbol{\tau}$ and angular velocity $\dot{\boldsymbol{\theta}}$ (or equivalently $\boldsymbol{\omega} = Z\dot{\boldsymbol{\theta}}$), an analytical function of motor efficiency [31] is adopted to evaluate the motor design $\boldsymbol{\beta}$ given the task set $\boldsymbol{\Gamma}$.

The motor design optimization stage then updates the motor parameters by maximizing motor efficiency during operation, given the closed-loop trajectories under unknown disturbances. The complete CLCD procedure is summarized in Algorithm 1.

Algorithm 1: Proposed CLCD Algorithm

Inputs: Task set $\boldsymbol{\Gamma}$, Initial motor design $\boldsymbol{\beta}_{\text{prev}}$
while $||err||_1 > \epsilon$ **do**
  $\boldsymbol{\theta}_{\text{ref}}, \dot{\boldsymbol{\theta}}_{\text{ref}} = \texttt{TrajOpt}(\boldsymbol{\Gamma}, \boldsymbol{\beta}_{\text{prev}})$
  $\boldsymbol{\tau}, \boldsymbol{\omega}, \boldsymbol{\pi} = \texttt{RL}(\boldsymbol{\Gamma}, \boldsymbol{\beta}_{\text{prev}}, \boldsymbol{\theta}_{\text{ref}}, \dot{\boldsymbol{\theta}}_{\text{ref}})$
  $\boldsymbol{\beta}^* = \texttt{MotorDesignOpt}(\boldsymbol{\tau}, \boldsymbol{\omega}, \boldsymbol{\beta}_{\text{prev}})$
  $err = \boldsymbol{\beta}^* - \boldsymbol{\beta}_{\text{prev}}$, $\boldsymbol{\beta}_{\text{prev}} = \boldsymbol{\beta}^*$
$\boldsymbol{\theta}^*_{\text{ref}}, \dot{\boldsymbol{\theta}}^*_{\text{ref}} = \texttt{TrajOpt}(\boldsymbol{\Gamma}, \boldsymbol{\beta}^*)$
$\boldsymbol{\tau}^*, \boldsymbol{\omega}^*, \boldsymbol{\pi}^* = \texttt{RL}(\boldsymbol{\Gamma}, \boldsymbol{\beta}^*, \boldsymbol{\theta}^*_{\text{ref}}, \dot{\boldsymbol{\theta}}^*_{\text{ref}})$
Return: $\boldsymbol{\beta}^*, \boldsymbol{\theta}^*, \dot{\boldsymbol{\theta}}^*, \boldsymbol{\pi}^*$
Note: $\epsilon = 1\,\text{mm}$

3.1. Control System Architecture

A straightforward selection of control system architecture would be having an RL-based controller to interact with the motor-robot hardware directly, completely eliminating classical model-based controllers such as PID or LQR. However, in our empirical study, such an approach results in inferior positioning accuracy and significantly longer training time. This can be interpreted from two perspectives. First, the motor-robot dynamic model is stiff, and thus the system performance is sensitive to control policy parameters, which renders the training algorithm difficult to converge. Second, with the bandwidth of the motor dynamics being kHz, the control output is also in kHz, and thus the training involves more data per episode and incurs a higher computation burden. For these reasons, this paper adopts a hybrid control architecture that combines the adaptability of RL with the reliability of classical controllers. Specifically, the RL policy is used as a trajectory reference compensator that adjusts reference trajectories to mitigate errors caused by disturbances or model uncertainties, while a classical controller is retained to ensure training efficiency and high-precision trajectory tracking.

3.2. Trajectory Optimization

The initial step of the CLCD involves planning the reference trajectory by solving a continuous-time optimal control problem (OCP), which is commonly transformed into a non-linear optimization problem (NLP) by performing discretization over time [30, 33, 34]. Given a task Γ_k, an optimizer, e.g., IPOPT (Interior Point Optimizer) [35], seeks optimal control inputs $\boldsymbol{u}^*$ so that the final state is reached while minimizing the following energy-related cost function:

$$J_{\text{TO}} = \int_{t_0}^{t_f} \boldsymbol{u}^\top \boldsymbol{Q} \boldsymbol{u} \, \mathrm{d}t, \tag{3}$$

where $\boldsymbol{Q}$ is a positive definite diagonal matrix. To perform the TO for all K tasks, one way is to formulate and solve a single NLP problem by considering all control inputs as decision variables. This approach however results in an excessive number of variables, rendering the problem intractable for a large K. An alternative is to solve each task Γ_k individually for all tasks and proceed to the next step. This decomposition maintains tractability while keeping the optimization problem dimensionality fixed.The corresponding step is denoted as $\texttt{TrajOpt}(\cdot)$ in Algorithm 1. Note that, at this stage, we assume no disturbances in the motor-robot model (1) and (2). The optimal state trajectories $\boldsymbol{\theta}^*$ and $\dot{\boldsymbol{\theta}}^*$ serve as reference inputs $\boldsymbol{\theta}_{\text{ref}}$ and $\dot{\boldsymbol{\theta}}_{\text{ref}}$ for the trajectory tracking step.

3.3. Rl-Based Compensator for Trajectory Tracking

This section presents the formulation, parameterization, and training of the RL-based feedback compensator to reject the uncertainties omitted during TO. The function $\texttt{RL}(\cdot)$ in Algorithm 1 performs policy training using reference trajectories $(\boldsymbol{\theta}_{\text{ref}}, \dot{\boldsymbol{\theta}}_{\text{ref}})$ generated by the TO stage and measurements from the simulated environment.

3.3.1. RL Problem Formulation and Policy Parameterization

In order to mitigate the uncertainties that are neglected during TO, naturally one would update the controllers at all levels and the reference trajectories. Next, we argue that with the model-based state feedback controller being the low-level motor control as in Figure 2, the RL policy only needs to commentate at the manipulator level.

Remark 5. *The closed-loop motor-robot system dynamics* (1) *and* (2) *can be rewritten as*

$$\begin{aligned}\dot{\boldsymbol{i}} &= \boldsymbol{f}_i(\boldsymbol{i}, \dot{\boldsymbol{\theta}}, \boldsymbol{u}),\\ \dot{\boldsymbol{x}}_r &= \boldsymbol{f}_r(\boldsymbol{x}_r, \boldsymbol{\tau}_{\mathrm{L}}, \boldsymbol{d}),\end{aligned}$$

where $\boldsymbol{x}_r = \begin{bmatrix}\boldsymbol{\theta}^\top & \dot{\boldsymbol{\theta}}^\top\end{bmatrix}^\top$ *and the current dynamics are not subject to disturbances. Since the state-feedback controller* $\boldsymbol{u}(\boldsymbol{i}, \boldsymbol{e}_i)$*, with* $\boldsymbol{e}_i = \boldsymbol{i} - \boldsymbol{i}^*(\boldsymbol{\tau}_{\mathrm{L}})$*, can render accurately torque (equivalently current) tracking with bandwidth at kHz, which is orders of magnitude faster than that of the robot motion dynamics, one can assume* $\boldsymbol{e}_i \approx 0$ *(ignore the current dynamics) and focus on the manipulator dynamics.*

From Figure 2, the closed-loop robot system dynamics (2) result from the model-based controller (without the RL module) can be simplified as

$$\dot{\boldsymbol{x}}_r = \boldsymbol{f}_{\boldsymbol{r}}(\boldsymbol{x}_r, \boldsymbol{\tau}_{\mathrm{L}}(\boldsymbol{x}_r - \hat{\boldsymbol{x}}_r, \hat{\boldsymbol{x}}_r), \boldsymbol{d}), \tag{4}$$

where $\hat{\boldsymbol{x}}_r$ is the reference input to the model-based control system and $\boldsymbol{d}$ represents bounded disturbances such as unmodeled joint friction. Denote $\boldsymbol{e}_x = \boldsymbol{x}_r - \boldsymbol{x}_r^*$ the tracking error, whose dynamics follow

$$\dot{\boldsymbol{e}}_x = \boldsymbol{f}_{\boldsymbol{r}}(\boldsymbol{e}_x + \boldsymbol{x}_r^*, \boldsymbol{\tau}_{\mathrm{L}}(\boldsymbol{e}_x + \boldsymbol{x}_r^* - \hat{\boldsymbol{x}}_r, \hat{\boldsymbol{x}}_r), \boldsymbol{d}) - \dot{\boldsymbol{x}}_r^*, \tag{5}$$

where $\boldsymbol{\tau}_{\mathrm{L}} = \boldsymbol{\tau}_{\mathrm{ff}}(\hat{\boldsymbol{x}}_r) + \boldsymbol{\tau}_{\mathrm{pd}}(\boldsymbol{e}_x + \boldsymbol{x}_r^* - \hat{\boldsymbol{x}}_r)$ is the summation of torque commands from model-based controllers. With $\boldsymbol{d} = 0$, there is no need to include the RL module. Without model mismatches in the TO and model-based control design, the closed-loop system with $\hat{\boldsymbol{x}}_r = \boldsymbol{x}_r^*$ gives satisfactory performance. When $\boldsymbol{d} \neq 0$, neither $\boldsymbol{x}_r^*$ nor the controllers are optimal.

We employ RL to synthesize a control policy for the tracking error dynamics (5). There are multiple options to set up the RL problem. First, if $\hat{\boldsymbol{x}}_r$ is treated as the control input, the RL objective is to synthesize reference trajectories $\hat{\boldsymbol{\theta}}_r$ such that the tracking error exhibits improved transient performance and zero steady-state error under disturbances. Consequently, the RL policy admits the parameterization $\hat{\boldsymbol{x}}_r = \hat{\boldsymbol{\pi}}(\boldsymbol{x}_r, \boldsymbol{e}_x, \boldsymbol{p}_{\boldsymbol{\pi}})$. Alternatively, one can rewrite (5) as follows

$$\dot{\boldsymbol{e}}_x = f_r(\boldsymbol{e}_x + \boldsymbol{x}_r^*, \boldsymbol{\tau}_{\mathrm{L}}(\boldsymbol{e}_x + \Delta\boldsymbol{x}_r, \boldsymbol{x}_r^* - \Delta\boldsymbol{x}_r), \boldsymbol{d}) - \dot{\boldsymbol{x}}_r^*,$$

where $\Delta\boldsymbol{x}_r = \boldsymbol{x}_r^* - \hat{\boldsymbol{x}}_r$ is treated as the virtual control input. Then the RL objective is to synthesize the correction $\Delta\boldsymbol{x}_r$ of the reference trajectories from the TO.

We finally adopt the following reduced parameterization:

$$\boldsymbol{s} \triangleq \begin{bmatrix}\boldsymbol{\theta}^\top & \dot{\boldsymbol{\theta}}^\top & \boldsymbol{e}^\top & \dot{\boldsymbol{e}}^\top\end{bmatrix}^\top \in \mathbb{R}^{4n}, \quad \boldsymbol{a} \triangleq \Delta\boldsymbol{\theta}_{\mathrm{ref}} \in \mathbb{R}^n, \tag{6}$$

where $\boldsymbol{e} \triangleq \boldsymbol{\theta} - \boldsymbol{\theta}_{\mathrm{ref}}$ and $\dot{\boldsymbol{e}} \triangleq \dot{\boldsymbol{\theta}} - \dot{\boldsymbol{\theta}}_{\mathrm{ref}}$ denote joint position and velocity tracking errors, respectively, and the policy is

$$\boldsymbol{a} \triangleq \Delta\boldsymbol{\theta}_{\mathrm{ref}} = \boldsymbol{\pi}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \boldsymbol{e}, \dot{\boldsymbol{e}}, \boldsymbol{p}_{\boldsymbol{\pi}}). \tag{7}$$

The reward function is designed to maximize tracking accuracy (via error penalties) while maintaining physical stability (via action penalties), effectively allowing the policy to compensate for unmodeled uncertainties and avoid destabilizing the system:

$$R_\pi = -\left(w_1\|\boldsymbol{e}\|_2 + w_2\|\dot{\boldsymbol{e}}\|_2 + w_3\|\boldsymbol{a}\|_2\right), \tag{8}$$

where $w_1, w_2, w_3 > 0$ are weighting coefficients. Decreasing the action penalization weight w_3 makes the RL controller more responsive to errors due to uncertainties, at the cost of more aggressive control actions, increased control effort, and potentially more jittery robot behavior. Therefore, the RL problem is formulated as

$$\boldsymbol{\pi}^* = \arg\max_{\boldsymbol{\pi}} \ \mathbb{E}_{\boldsymbol{s},\boldsymbol{a}}\left[R_\pi\right]. \tag{9}$$

3.3.2. Environment

The RL environment is constructed in simulation using the motor and robot dynamics described in Section 2.1.1 and Section 2.1.2. The environment receives the reference trajectory ($\boldsymbol{\theta}^*_{\mathrm{ref}}$,$\dot{\boldsymbol{\theta}}^*_{\mathrm{ref}}$) from the TO stage and the corrective action $\boldsymbol{a}$ from the RL policy, producing a corrected reference ($\hat{\boldsymbol{\theta}}_{\mathrm{ref}}$, $\dot{\hat{\boldsymbol{\theta}}}_{\mathrm{ref}}$). This corrected reference is used by both the feedforward controller and the PD controller, which together generate the desired torque command $\boldsymbol{\tau}^*$.

The state-feedback controller ensures accurate torque tracking. In particular, motor dynamics are simulated at 2.5 kHz, while robot dynamics are simulated at 25 Hz. The use of dual simulation frequencies is motivated by the inherent separation in time scales between actuator-level and system-level dynamics in robotic systems. Consistent with prior work on actuator-aware modeling and multi-rate simulation [36–38], we integrate motor dynamics at 2.5 kHz to capture fast transient behavior, while updating robot dynamics at 25 Hz, which is sufficient for task-level motion and improves computational efficiency during training. The voltage commands (for a given motor) are given by

$$u_{\mathrm{d}} = L_{\mathrm{d}}(-P\omega i_{\mathrm{q}} + Ri^*_{\mathrm{d}}/L_{\mathrm{d}} - K_{\mathrm{d}}(i_{\mathrm{d}} - i^*_{\mathrm{d}})),$$
$$u_{\mathrm{q}} = L_{\mathrm{d}}(-P\omega(i_{\mathrm{d}} + \Phi_{\mathrm{m}}/L_{\mathrm{d}}) + Ri^*_{\mathrm{q}}/L_{\mathrm{d}} - K_{\mathrm{q}}(i_{\mathrm{q}} - i^*_{\mathrm{q}})),$$

where $i^*_{\mathrm{q}}, i^*_{\mathrm{d}}$ can be calculated for any torque command $\boldsymbol{\tau}^*$ ([31], Algorithm 1).

3.3.3. Training Algorithm and Training Efficiency

In the CLCD framework, iterative motor design updates can substantially alter system dynamics, making full policy retraining at each iteration computationally impractical. It entails an algorithm that not only results in high tracking accuracy but also achieves fast convergence.

We evaluated several representative algorithms, including PPO (proximity policy optimization), DDPG (deep deterministic policy gradient), TD3 (twin-delayed deep deterministic policy gradient), and SAC (soft actor-critic). SAC achieved the best performance in training time, cumulative reward, and tracking accuracy, and is therefore adopted in this work. In addition, transfer learning across co-design iterations is essential, as the motor design evolves during optimization. SAC is particularly well suited for this setting due to its off-policy nature, which allows previously collected data and learned network parameters to be effectively reused. This enables efficient warm-starting of both the policy and value networks, significantly reducing training time in subsequent iterations [39]. Accordingly, we initialize the networks using parameters from the previous iteration. Furthermore, as an off-policy actor–critic method, SAC improves sample efficiency through experience replay and employs a maximum-entropy objective to promote stable learning [40]. The stochastic policy and soft Q-function are parameterized using neural networks, with a target Q-function introduced to further enhance training stability. To balance the computational cost of control synthesis and motor design optimization, the number of training episodes per iteration is limited to the number of tasks, enabling efficient policy adaptation throughout the co-design process.

3.4. Motor Design Optimization

Finally, `MotorDesignOpt`$(\cdot)$ function in Algorithm 1, updates motor design $\boldsymbol{\beta}$ to maximize operational efficiency $\boldsymbol{\eta}(\tau,\omega;\boldsymbol{\beta}) = [\eta_1(\tau,\omega,\boldsymbol{\beta}_1),\cdots,\eta_n(\tau,\omega,\boldsymbol{\beta}_n)]^\top$ of n motors while accomplishing all tasks. Take the ith motor as an example. The requirement of all tasks on the ith motor is characterized by the operation data $(\boldsymbol{\tau}_i(t),\boldsymbol{\omega}_i(t))$ associated with previous design $\boldsymbol{\beta}_{\mathrm{prev},i}$, where $\boldsymbol{\tau}_i = \{\tau_{i,1}(t),\cdots,\tau_{i,K}(t)\}$ with $\tau_{i,k}(t)$ denoting the torque trajectory of the ith motor performing the kth task. Given the operational data, one can construct a probability density function $q_i(\tau,\omega,\boldsymbol{\beta}_{\mathrm{prev},i})$ over the torque-speed plane for all motors ($1 \le i \le n$) and have a vector of known distributions given by previous design $\boldsymbol{\beta}_{\mathrm{prev}}$: $\boldsymbol{q}(\tau,\omega;\boldsymbol{\beta}_{\mathrm{prev}}) = [q_1,\cdots,q_n]^\top$. The design optimization solves the following problem:

$$\min_{\boldsymbol{\beta}} \quad \int_0^{\tau_{\max}}\int_0^{\omega_{\max}} \boldsymbol{q}^\top(\tau,\omega;\boldsymbol{\beta}_{\mathrm{prev}})(1-\boldsymbol{\eta}(\tau,\omega;\boldsymbol{\beta}))\,\mathrm{d}\omega\mathrm{d}\tau \tag{10a}$$

$$\text{s.t.} \quad \boldsymbol{g}_{\mathrm{d}}(\boldsymbol{\beta}) \le \mathbf{0}, \tag{10b}$$

where $\boldsymbol{g}_{\mathrm{d}}(\boldsymbol{\beta})$ represents all the design constraints. An example of $\boldsymbol{g}_{\mathrm{d}}(\boldsymbol{\beta})$ is shown as the first 6 rows of Constraints in Table 2. It is noteworthy that $\boldsymbol{\eta}(\tau,\omega;\boldsymbol{\beta})$ is a vector of analytical functions derived during modeling.

Table 2. The co-design problem statement ($n = 6, N = 100$).

	Description	Lower	Variable	Upper	Optimization Quantity	
					Trajectory	Motor Design
Optimization variables $\boldsymbol{\beta}, \boldsymbol{u}(t)$	Axial length of core (mm)	20	l	100		n
	Outer radius of rotor (mm)	10	r_{ro}	100		n
	Outer radius of stator (mm)	10	r_{so}	100		n
	Height of magnet (mm)	1	h_{m}	5		n
	Stator yoke (mm)	5	h_{sy}	10		n
	Width of tooth (mm)	5	w_{tooth}	20		n
	Slot opening (rad)	1	b_0	10		n
	Voltage d axis (V)	-100	u_{d}	100	$n \times N$	
	Voltage q axis (V)	-100	u_{q}	100	$n \times N$	
	Total				$2 \times n \times N$	$7 \times n$
Constraints	Slot height	0	h_{ss}			n
	mass of stator and rotor (kg)	0	$m_{\mathrm{stator}} + m_{\mathrm{rotor}}$	3		$2 \times n$
	D Wire (mm)	0.6	D_{wire}			n
	Tooth width bound (mm)	0	$\frac{w_{\mathrm{tooth}}}{2(R_{\mathrm{ro}}+\delta)} + \frac{b_0}{2(R_{\mathrm{ro}}+\delta)}$	$\frac{\pi}{Q}$		$2 \times n$
	Magnetic fluxes in tooth (T)	0	$\frac{k_{\mathrm{p}}\phi_1}{w_{\mathrm{tooth}}L}$	1.5		$2 \times n$
	Magnetic fluxes in stator yoke (T)	0	$\frac{1}{\sqrt{3}}\frac{k_{\mathrm{p}}\phi_1}{h_{\mathrm{sy}}L}$	1.5		$2 \times n$
	Joint angle	-2π	$\theta_{1,4,6}$	2π	$2 \times 3 \times N$	
	Joint angle	-0.6π	$\theta_{2,3,5}$	0.6π	$2 \times 3 \times N$	
	Joint velocity	-2π	$\dot{\theta}$	2π	$2 \times n \times N$	
	current d axis (A)	-3	$i_{\mathrm{d},k}$	0	$2 \times n \times N$	$2 \times n$
	current q axis (A)	-3	$i_{\mathrm{q},k}$	3	$2 \times n \times N$	$2 \times n$
	Voltage d axis (V)	-100	$u_{\mathrm{d},k}$	100	$2 \times n \times N$	$2 \times n$
	Voltage q axis (V)	-100	$u_{\mathrm{q},k}$	100	$2 \times n \times N$	$2 \times n$
	Total				$2\times5\times n\times N+12\times N$	$18 \times n$

This design optimization problem can be solved numerically, and the resultant motor design is forwarded to the TO step as the next iteration. To assess the convergence of motor design parameters, we define the exit criterion as $||err||_1 \leq \epsilon$, where $err = \boldsymbol{\beta} - \boldsymbol{\beta}_{\mathrm{prev}}$ and ϵ denotes the tolerance. If the criterion is not met, the co-design loop continues. Once exiting the co-design loop, the optimized motor design is used to perform the TO and RL training phase again to fully complete the co-design algorithm.

4. Simulation Result

In this section, we apply both the proposed CLCD and a baseline OLCD method to a 6-DOF robot manipulator ($n = 6$) and compare the trajectory tracking performance of the resultant closed-loop control systems as well as the consequent motor designs. The co-design problem statement is detailed in Table 2, which provides a comprehensive list of optimization variables and constraints, along with their respective quantities. Main differences between the CLCD and OLCD paradigms are: (1) the latter only performs the TO and motor design optimization during design, and (2) during real-time trajectory tracking control, the latter excludes the RL block from Figure 2. For the simulation setup, we chose two different total numbers of tasks ($K = 12$ and $K = 120$) to validate the scalability of the proposed framework. Each task was generated by randomly selecting feasible initial and final states, with payloads ranging from 0 to 4 kg. The TO is conducted with a final time of 4 s and a time step of 0.04 s, which yields the total number of time steps $N = 100$. A pre-defined gear ratio of 50 is applied to all six motors. For the RL problem, the dimensions of the action, denoted by $\boldsymbol{a} \in \mathbb{R}^6$, and the state, denoted by $\boldsymbol{s} \in \mathbb{R}^{24}$, are determined as described in Section 3.3. The convergence criterion is set to $||err||_1 \leq 1$ mm. The simulation is performed on a PC with Intel® Core™ i7-8700 CPU @ 3.20 GHz. In addition, the PyTorch library and OpenAI Gym [41] are used for the RL training.

The convergence of the co-design algorithm is depicted in Figure 3. The result shows that despite the order of magnitude increase in the number of tasks, the number of co-design iterations remains stable with similar convergence behavior. In addition, the total computation time is close to linear growth as the number of tasks increases (4.2 hrs for $K = 12$ and 54.9 hrs for $K = 120$). This is expected because the computation burden primarily stems from performing the TO for all tasks. Computation efficiency can be further improved by implementing the

trajectory and motor dynamics in JAX for GPU-parallelized computation. Furthermore, the number of optimization steps provides a tunable trade-off between solution high-fidelity and computational overhead.

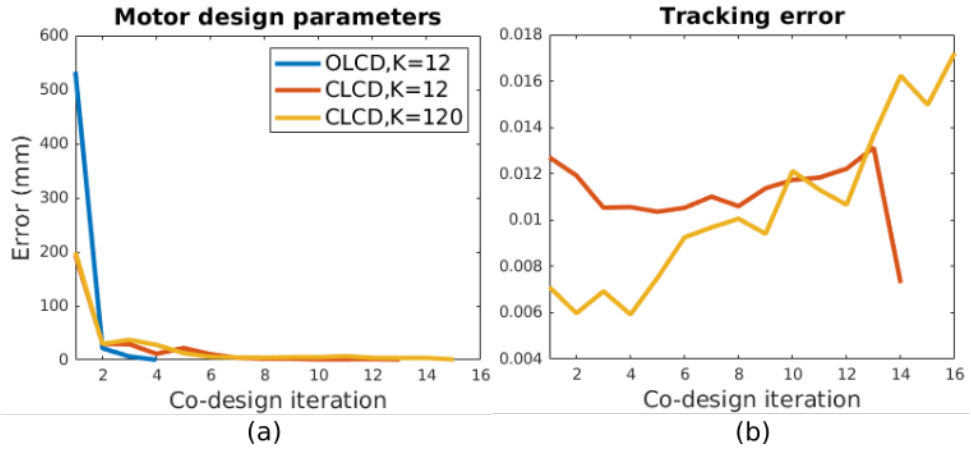

Figure 3. (**a**) shows the convergence of $||err||_1$ from the proposed CLCD framework. (**b**) presents the trajectory tracking error from the first to the last co-design iteration.

4.1. Trajectory Tracking

This section assesses the scalability and the tracking performance of the CLCD approach. Joint space trajectories are optimized through the TO and then treated as reference trajectories for closed-loop tracking control. The actor and critic networks for training both feature two fully-connected hidden layers with 256 nodes each. employing a consistent learning rate of 10^{-4}. The decay factor γ and temperature α are set to 0.99 and 0.02, respectively. While we adopted the standard default decay parameters for our experiments, some hyperparameters can be sensitive and require careful tuning for specific problems. Therefore, we conducted a parameter sweep to select the learning rate and temperature, ensuring both stable convergence and an appropriate convergence rate as the motor hardware parameters evolve across iterations. In addition, the PD controller is tuned to ensure system stability while achieving a stiff and responsive behavior. In each episode, the time step is 0.04 s, and the final time ($t_\mathrm{f} = 4$ s) is set to match that of the TO. Figure 3 illustrates the tracking error of the closed-loop control system result from the CLCD. Remarkably, the final RMS tracking error for 12 and 120 tasks are within the same order of magnitude. With 120 tasks, the RMS error increases as the design parameters converge. This behavior is expected since the controllers are not fine-tuned for a specific task but rather adapted to a broader range of tasks.

The trajectory tracking performance of the closed-control systems result from the CLCD and the OLCD is compared. The gains of the PD controller are meticulously tuned and set to identical values for both the CLCD and the OLCD approaches. The simulation results of both CLCD and OLCD for 12 tasks are showcased in Figure 4a,b, demonstrating the tracking performance of θ_1 and θ_6, respectively. Furthermore, Figure 4c,d illustrate the tracking performance of CLCD for 12 and 120 tasks, respectively. The results distinctly indicate the superior performance of the CLCD, exhibiting a consistent order of magnitude improvement in RMS tracking error compared to the OLCD, as shown in Table 3. The error magnitude for 120 tasks is notably higher, particularly when $t < 2$ s. This behavior may stem from the fundamental tradeoff between optimality and robustness of a static feedback-control policy when applied to a larger volume of tasks. Despite this, the overall performance of the CLCD surpasses that of the OLCD, underscoring the versatility and effectiveness of the proposed approach. Consequently, the CLCD can adeptly manage disturbances by adjusting the reference joint trajectory. Moreover, the current framework can accommodate model uncertainties and other types of disturbances, enabling simulation in various customized environments.

Table 3. Tracking Error (Unit: rad).

	Mean (μ_{θ_1})	**Std (σ_{θ_1})**	**Mean (μ_{θ_6})**	**Std (σ_{θ_6})**
CLCD (K = 12)	**0.0003**	0.0089	**0.0022**	0.0043
OLCD (K = 12)	0.0031	0.0222	0.0634	0.0458
CLCD (K = 120)	0.0096	0.0144	0.0078	0.0138

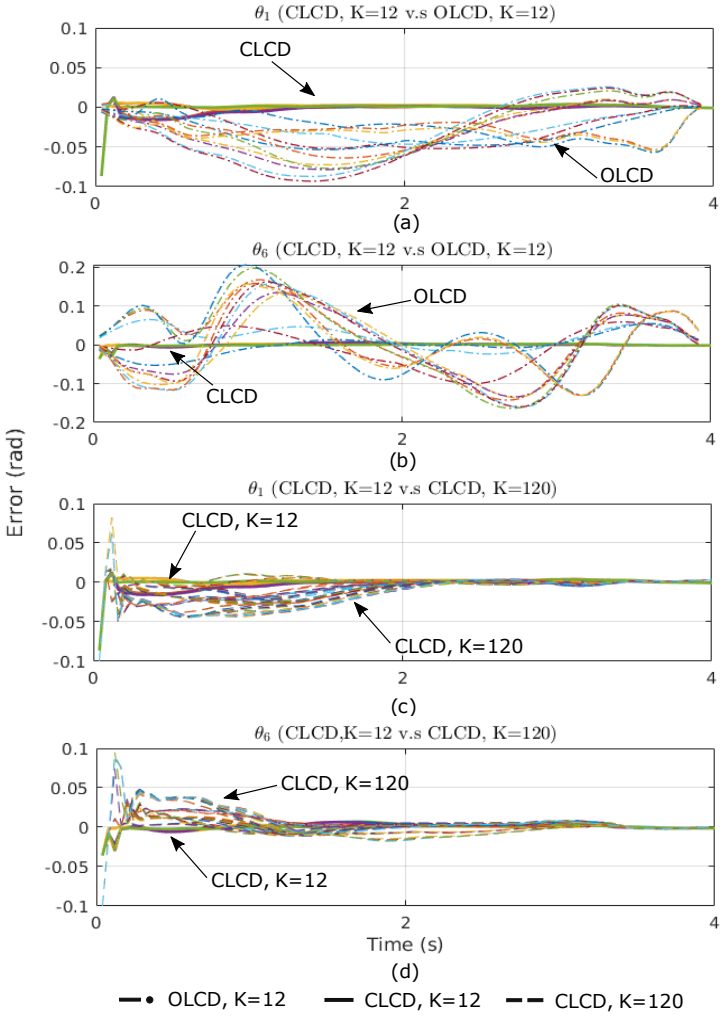

Figure 4. (**a**) and (**b**) display the tracking errors for θ_1 and θ_6 with all 12 trajectories under both CLCD and OLCD, with each trajectory represented by a distinct color, (**c**) and (**d**) illustrate the tracking errors for θ_1 and θ_6, respectively. Furthermore, both θ_1 and θ_6 with the number of trajectories, $K = 12$ and $K = 120$, are presented to demonstrate the difference in the resultant tracking accuracy.

4.2. Motor Design

The optimized motor parameters for both CLCD ($K = 12, 120$) and OLCD ($K = 12$) approaches are compared using a spider plot as shown in Figure 5. The scale in the spider plot is anchored on the lower and upper bounds of each design parameter. The comparison highlights significant disparities in motor designs from the two methods, particularly in the axial length of the core (l) across motors 1 to 6. A longer axial length of the core corresponds to increased magnetic flux, resulting in higher motor torque. This aligns with expectations, as the CLCD process accounts for disturbances during the design phase. Consequently, to compensate for friction in the motor and gearhead, a higher torque requirement becomes necessary. Other key design parameters, such as the stator outer radius (r_{so}) and tooth width (w_{tooth}), are reduced to compensate for the increased axial length l, thereby limiting the growth in resistance and inertia due to the increased material volume. Moreover, the CLCD with 120 tasks exhibits a slight increase in l compared to 12 tasks, which is also expected due to the variation of the tasks, consisting of different initial and final states and loads. This increase reflects the adaptability of the CLCD approach to account for a wider range of task scenarios and optimize the motor design accordingly. Other observable differences in the design parameters are considered minor compared to the axial length of the core (l). However, variations in any design parameters can impact the motor's dynamic performance and generalizability across hundreds of tasks in an industrial setting.

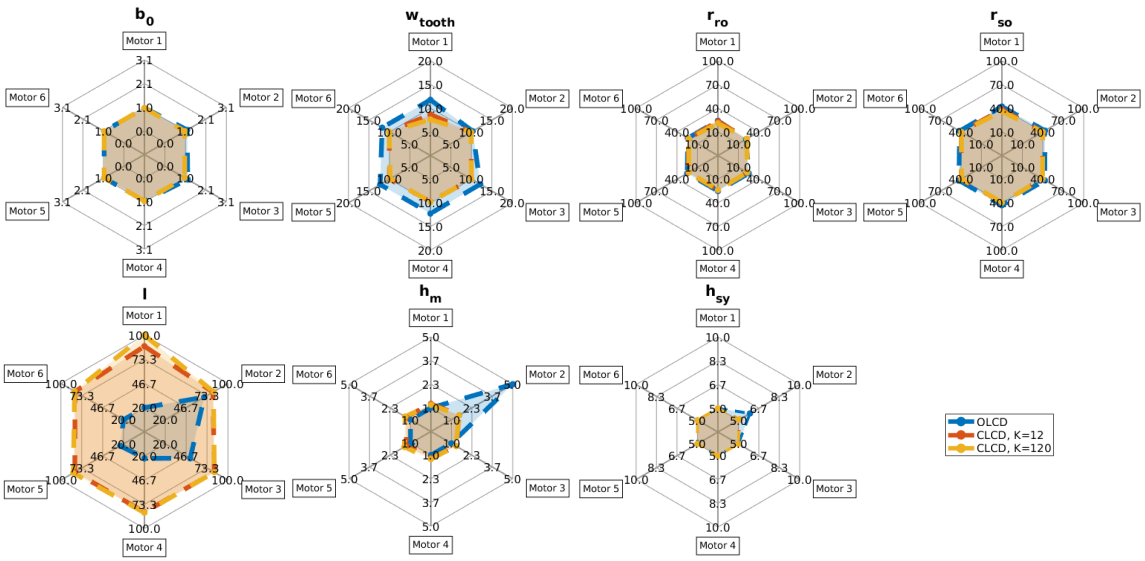

Figure 5. The 7 motor design parameters are optimized, and the spider plot showcases the difference in motor design parameters when the co-design algorithm is used with and without the closed-loop component.

5. Conclusions and Future Work

A novel CLCD framework is proposed, comprising three pivotal steps: the TO, feedback control policy learning through the integration of an RL-based compensator with classic PD control, and motor design optimization. The results emphasize the essential nature of incorporating a closed-loop system into the co-design framework, as disturbances can significantly influence system performance and motor design. This consideration becomes critical for a comprehensive and effective co-design strategy. The framework demonstrates the utility of the RL-based compensator in enhancing tracking error resilience when encountering disturbances. It also establishes the feasibility of efficiently training RL policies within each co-design iteration. In addition, we address high-dimensional optimization by decomposing the co-design problem into tractable subproblems and performing each trajectory optimization independently, thereby maintaining tractability and convergence quality, as demonstrated in the experiments.

The proposed three-step framework is modular, enabling easy integration of alternative state-of-the-art controllers, such as model predictive control (MPC) or learning-based approaches. Future work will delve into exploring the robustness of the controller across a broader spectrum of tasks and incorporating model uncertainties. Another interesting direction is to extend the co-design framework to multi-agent robotic systems and bimanual manipulators for more complex and dexterous tasks beyond a single-arm setting. These enhancements aim to further elevate the adaptability and reliability of the proposed co-design framework.

Author Contributions

J.-T.L.: conceptualization, methodology, software, data curation, validation; Z.L.: conceptualization, methodology, software; Y.W.: conceptualization, methodology, supervision. All authors have read and agreed to the published version of the manuscript.

Data Availability Statement

Not applicable.

Conflicts of Interest

The authors declare no conflict of interest.

Use of AI and AI-Assisted Technologies

During the preparation of this work, the authors used Google Gemini for the purpose of language refinement and grammatical editing. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article.

References

1. De Michell, G.; Gupta, R.K. Hardware/Software Co-Design. *Proc. IEEE* **1997**, *85*, 349–365.
2. Jiang, Y.; Wang, Y.; Bortoff, S.A.; et al. Optimal Codesign of Nonlinear Control Systems Based on a Modified Policy Iteration Method. *IEEE Trans. Neural Netw. Learn. Syst.* **2015**, *26*, 409–414.
3. Hale, A.L.; Dahl, W.; Lisowski, J. Optimal Simultaneous Structural and Control Design of Maneuvering Flexible Spacecraft. *J. Guid. Control Dyn.* **1985**, *8*, 86–93.
4. Ravichandran, T.; Wang, D.; Heppler, G. Simultaneous Plant-Controller Design Optimization of a Two-Link Planar Manipulator. *Mechatronics* **2006**, *16*, 233–242.
5. Park, J.H.; Asada, H. Concurrent Design Optimization of Mechanical Structure and Control for High Speed Robots. In Proceedings of the 1993 American Control Conference, San Francisco, CA, USA, 2–4 June 1993.
6. Pettersson, M.; Olvander, J. Drive Train Optimization for Industrial Robots. *IEEE Trans. Robot.* **2009**, *25*, 1419–1424.
7. Ha, S.; Coros, S.; Alspach, A.; et al. Computational Co-Optimization of Design Parameters and Motion Trajectories for Robotic Systems. *Int. J. Robot. Res.* **2018**, *37*, 1521–1536.
8. Herber, D.R.; Allison, J.T. Nested and Simultaneous Solution Strategies for General Combined Plant and Control Design Problems. *J. Mech. Des.* **2019**, *141*, 011402.
9. Garcia-Sanz, M. Control Co-Design: An Engineering Game Changer. *Adv. Control Appl.* **2019**, *1*, e18.
10. Hwang, J.T. A Modular Approach to Large-Scale Design Optimization of Aerospace Systems. Ph.D. Thesis, University of Michigan, Ann Arbor, MI, USA, 2015.
11. Toussaint, M.; Ha, J.S.; Oguz, O.S. Co-Optimizing Robot, Environment, and Tool Design via Joint Manipulation Planning. In Proceedings of the 2021 IEEE International Conference on Robotics and Automation, Xi'an, China, 30 May–5 June 2021.
12. Chen, T.; He, Z.; Ciocarlie, M. Co-Designing Hardware and Control for Robot Hands. *Sci. Robot.* **2021**, *6*, eabg2133.
13. Dinev, T.; Mastalli, C.; Ivan, V.; et al. A Versatile Co-Design Approach for Dynamic Legged Robots. In Proceedings of the 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Kyoto, Japan, 23–27 October 2022; pp. 10343–10349.
14. Baykal, C.; Alterovitz, R. Asymptotically Optimal Design of Piecewise Cylindrical Robots using Motion Planning. In Proceedings of the Robotics: Science and Systems, Cambridge, MA, USA, 12–16 July 2017.
15. Bhatia, J.; Jackson, H.; Tian, Y.; et al. Evolution Gym: A Large-Scale Benchmark for Evolving Soft Robots. In Proceedings of the 35th Conference on Neural Information Processing Systems (NeurIPS 2021), virtual, 6–14 December 2021; pp. 2201–2214.
16. Fan, Q.Y.; Wang, D.; Xu, B. H_∞ Codesign for Uncertain Nonlinear Control Systems Based on Policy Iteration Method. *IEEE Trans. Cybern.* **2021**, *52*, 10101–10110.
17. Zhang, W.; Huang, Y.; Xie, L. Infinite Horizon Stochastic H_2/H_∞ Control for Discrete-Time Systems with State and Disturbance Dependent Noise. *Automatica* **2008**, *44*, 2306–2316.
18. Bravo-Palacios, G.; Grandesso, G.; Prete, A.D.; et al. Robust Co-Design: Coupling Morphology and Feedback Design through Stochastic Programming. *J. Dyn. Syst. Meas. Control* **2022**, *144*, 021007.
19. Benosman, M.; Le Vey, G. Control of Flexible Manipulators: A Survey. *Robotica* **2004**, *22*, 533–545.
20. Yuan, M.; Manzie, C.; Good, M.; et al. A Review of Industrial Tracking Control Algorithms. *Control Eng. Pract.* **2020**, *102*, 104536.
21. Visioli, A.; Legnani, G. On the Trajectory Tracking Control of Industrial SCARA Robot Manipulators. *IEEE Trans. Ind. Electron.* **2002**, *49*, 224–232.
22. Cervantes, I.; Alvarez-Ramirez, J. On the PID Tracking Control of Robot Manipulators. *Syst. Control Lett.* **2001**, *42*, 37–46.
23. Lin, F. An Optimal Control Approach to Robust Control of Robot Manipulators. *IEEE Trans. Robot. Autom.* **1998**, *14*, 69–77.
24. Green, A.; Sasiadek, J.Z. Dynamics and Trajectory Tracking Control of a Two-Link Robot Manipulator. *J. Vib. Control* **2004**, *10*, 1415–1440.
25. Wai, R.J. Tracking Control Based on Neural Network Strategy for Robot Manipulator. *Neurocomputing* **2003**, *51*, 425–445.
26. Jiang, Z.H.; Ishida, T. A Neural Network Controller for Trajectory Control of Industrial Robot Manipulators. *J. Comput.* **2008**, *3*, 1–8.

27. Dai, L.; Yu, Y.; Zhai, D.H.; et al. Robust Model Predictive Tracking Control for Robot Manipulators with Disturbances. *IEEE Trans. Ind. Electron.* **2020**, *68*, 4288–4297.
28. Boscariol, P.; Gasparetto, A.; Zanotto, V. Model Predictive Control of a Flexible Links Mechanism. *J. Intell. Robot. Syst.* **2010**, *58*, 125–147.
29. Pane, Y.P.; Nageshrao, S.P.; Kober, J.; et al. Reinforcement Learning Based Compensation Methods for Robot Manipulators. *Eng. Appl. Artif. Intell.* **2019**, *78*, 236–247.
30. Stein, A.; Wang, Y.; Sakamoto, Y.; et al. Application-Oriented Co-Design of Motors and Motions for a 6DOF Robot Manipulator. In Proceedings of the 2025 IEEE International Conference on Automation Science and Engineering (CASE), Los Angeles, CA, USA, 17–21 August 2025; pp. 3318–3324.
31. Lu, Z.; Wang, Y. A Differentiable Dynamic Modeling Approach to Integrated Motion Planning and Actuator Physical Design for Mobile Manipulators. *J. Field Robot.* **2025**, *42*, 37–64.
32. Andersson, J.A.E.; Gillis, J.; Horn, G.; et al. CasADi: A Software Framework for Nonlinear Optimization and Optimal Control. *Math. Program. Comput.* **2019**, *11*, 1–36 .
33. Betts, J.T. *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*; SIAM: Philadelphia, PA, USA, 2010.
34. Zhao, Y.; Wang, Y.; Zhou, M.C.; et al. Energy-Optimal Collision-Free Motion Planning for Multiaxis Motion Systems: An Alternating Quadratic Programming Approach. *IEEE Trans. Autom. Sci. Eng.* **2019**, *16*, 327–338.
35. Wächter, A.; Biegler, L.T. On the Implementation of an Interior-Point Filter Line-Search Algorithm for Large-Scale Nonlinear Programming. *Math. Program.* **2006**, *106*, 25–57.
36. Tan, J.; Zhang, T.; Coumans, E.; et al. Sim-to-Real: Learning Agile Locomotion for Quadruped Robots. *arXiv* **2018**, arXiv:1804.10332.
37. Yu, N.; Zou, W.; Sun, Y. Passivity Guaranteed Stiffness Control with Multiple Frequency Band Specifications for a Cable-Driven Series Elastic Actuator. *Mech. Syst. Signal Process.* **2019**, *117*, 709–722.
38. Bicego, D.; Mazzetto, J.; Carli, R.; et al. Nonlinear Model Predictive Control with Enhanced Actuator Model for Multi-Rotor Aerial Vehicles with Generic Designs. *J. Intell. Robot. Syst.* **2020**, *100*, 1213–1247.
39. Torrey, L.; Shavlik, J. Transfer Learning. In *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*; IGI Global: Hershey, PA, USA, 2010; pp. 242–264.
40. Haarnoja, T.; Zhou, A.; Hartikainen, K.; et al. Soft Actor-Critic Algorithms and Applications. *arXiv* **2018**, arXiv:1812.05905.
41. Brockman, G.; Cheung, V.; Pettersson, L.; et al. Openai Gym. *arXiv* **2016**, arXiv:1606.01540.