Journal of Field Robotics

WILEY

RESEARCH ARTICLE OPEN ACCESS

A Cascaded Strategy With Embodied Artificial Intelligence: Forward Kinematics Solutions for CCRobot-S

Zhenliang Zheng[1,2] | Yongyuan Xu[1,3] | Xuchun He[2] | Tin Lun Lam[1,2] | Ning Ding[2]

[1]School of Science and Engineering, The Chinese University of Hong Kong (CUHK), Shenzhen, China | [2]Shenzhen Institute of Artificial Intelligence and Robotics for Society (AIRS), Shenzhen, China | [3]Institute of Control Systems, Hamburg University of Technology, Hamburg, Germany

Correspondence: Tin Lun Lam (tllam@cuhk.edu.cn) | Ning Ding (dingning@cuhk.edu.cn)

Received: 31 January 2025 | **Revised:** 20 October 2025 | **Accepted:** 24 November 2025

Funding: Guangdong S&T Program, Grant/Award Number: 2025B0909040003; Shenzhen Science and Technology Program, Grant/Award Number: GJHZ20240218114202004; Guangdong Provincial Leading Talent Program, Grant/Award Number: 2024TX08Z319; "Zhiguo" Action of Guangxi Science and Technology Program, Grant/Award Number: ZG2503980003; Longgang District Shenzhen's "Ten Action Plan" for Supporting Innovation Projects, Grant/Award Number: LGKCSDPT2024003.

Keywords: cable climbing robot | cascaded strategy | collaborative climbing robot squad | embodied artificial intelligence | forward kinematics | reconfigurable cable-driven parallel robots

ABSTRACT

This paper presents a novel cable-climbing mechanism: the Collaborative Climbing Robot Squad (CCRobot-S), a variant of Reconfigurable Cable-Driven Parallel Robots (R-CDPR), specifically designed for the inspection and maintenance of stay cables. The forward kinematics of the CCRobot-S robotic system, however, is inherently mathematically intractable. This research proposes a novel cascaded strategy with Embodied Artificial Intelligence (EAI) to effectively tackle the forward kinematics problem. In this proposed strategy, a lightweight deep learning-based model integrated with numerical method optimization supplants traditional methods, providing feedback on the poses of the flying platform to the control loop of the CCRobot-S robotic system. It provides an approximate solution as initial values through a deep neural network by learning from physical or simulated interactive experiences of CCRobot-S, and then transfers the suitable initial values with kinematic constraints or physical constraints that are near the real solution to the numerical method. This process achieves a stable and robust solution for the forward kinematics of CCRobot-S. This article includes the foundational kinematic analysis of CCRobot-S, the formulation of the CCRobot-S model, a comprehensive introduction and analysis of the cascaded strategy, including the dataset preparation, the training configuration, the solution inference, and the numerical method optimization. Comprehensive evaluations and experiments were undertaken to examine the proposed strategy. The results reveal and confirm that the deep-learning neural network implemented in the CCRobot-S robotic system is effective. Additionally, the proposed cascaded strategy achieves higher prediction accuracy than the standalone neural network approach under the condition of real-time execution (position error reduced from 1.43 ± 1.88 mm to 0.16 ± 0.73 mm in the X direction, from -2.72 ± 4.05 mm to -0.46 ± 0.85 mm in the Y direction, and from $2.32 \pm 2.51°$ to $-0.65 \pm 0.6°$ in the Θ orientation). The cascaded strategy also guarantees convergence in 100% of test cases (50/50) and demonstrates enhanced stability and robustness (1:1 mapping from the joint space to the task space)relative to the conventional Newton-Raphson algorithm's numerical method. These attributes are crucial and necessary for the CCRobot-S system to be effectively deployed in real-world applications.

1 | Introduction

Various cable-climbing robots have been developed to inspect and maintain bridge cables, which serve as the primary load-bearing components of cable-stayed bridges. These cables are susceptible to long-term damage caused by corrosion due to rainwater penetration, wear resulting from wind- and rain-induced vibrations, and external forces. The majority of these cable climbing robots, as documented in the literature, are focused on stay cable inspection with carrying detection (Mengqi et al. 2024; Cho et al. 2013a, 2013b, 2014, 2016; Xu et al. 2011; Nguyen et al. 2024, 2022; Tavakoli et al. 2012, 2011; Wang et al. 2021; Nguyen et al. 2024; Xu et al. 2025), with only a small amount of cable climbing robots used for cable maintenance tasks (Xu et al. 2021; Luo et al. 2007; Li et al. 2009; Yuan et al. 2010), since inspection tasks only need to move robots with low loads, while maintenance tasks always need to equip heavy payloads. Current cable-climbing robots can rarely directly equip such units due to their low load capacity. Meanwhile, most of the current cable climbing robots are constrained to a stay cable, and they can not cross over the stay cables, which leads to low working efficiency.

The category of Cable-Driven Parallel Robots (CDPRs) that possess the ability to dynamically alter cable attachment points is designated as Reconfigurable Cable-Driven Parallel Robots (R-CDPRs) (Piva et al. 2025; Trautwein et al. 2025; Wu et al. 2025). Reconfiguration is generally accomplished by permitting the cable attachment points on the base frame to move without restriction. Typically, R-CDPRs are composed of a mobile platform (end-effector) connected to a movable base via flexible cables. The lengths of these cables are adjusted through the use of winches, with the movable base facilitating platform motion control. Cables provide the additional benefit of being capable of being coiled, which is advantageous for both actuation and transport purposes. In contrast to conventional CDPRs (Xu et al. 2024; Gao et al. 2025; Wang et al. 2025), the bases of R-CDPRs are not fixed; the geometric structure can be reconfigured by adjusting the cable setup. This reconfigurability affords a greater range of flexible configuration options and enhances the robot's performance in several ways, such as enlarging workspaces, boosting system rigidity, enhancing payload capacity, or reducing cable tensions.

As such, we introduce a novel cable climbing mechanism: a Collaborative Climbing Robot Squad (CCRobot-S) based on R-CDPR for bridge cables inspection and maintenance tasks, as shown in Figure 1. This system allows for a reconfigurable kinematic morphology through its movable anchor bases and enables the capacity to cross over the stay cables for its flying platform. Compared to our earlier CCRobot series (Zheng et al. 2018a, 2018b, 2025; Zheng and Ding 2019; Ding et al. 2020; Zheng et al. 2022; Zhang et al. 2021; Zheng et al. 2021), the collaborative robot squad design liberates the dimensions and scales of the robot's reachable workspace. It moves the part of the robotic system that indeed needs to be moved, thereby enhancing the working efficiency and climbing agility. Additionally, this strategy utilizes controllable adhesion instead of friction to interact with the bridge cable surface for the flying platform, achieving force multiplication for more effective manipulation.

Although CCRobot-S significantly enhances the efficiency and heavy-duty capacity for cable inspection and maintenance tasks, the mobility of its base introduces redundancy, which in turn increases the complexity of motion control for the flying platform. Given the augmented number of reconfigurable Degrees of Freedom (DoFs), the key challenge in the motion control of the CCRobot-S is to ascertain a stable and accurate solution to the forward kinematics. In the motion control of CCRobot-S, forward kinematics is used to estimate the pose and velocity of the flying platform as feedback values in the control loop. The solution to the forward kinematics problem is essential for completing the control loop in the motion control system. Specifically, the forward kinematics determines whether the desired point within the workspace has been reached. This information is crucial for adjusting the robot's movements to ensure precise positioning and effective task execution.

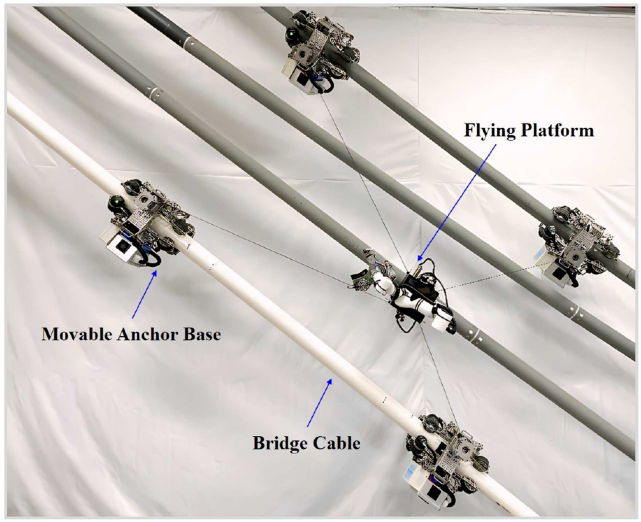

FIGURE 1 | The Collaborative Climbing Robot Squad (CCRobot-S) based on the reconfigurable cable-driven parallel mechanism. [Color figure can be viewed at wileyonlinelibrary.com]

For serial robots, it is straightforward to calculate the pose of the end-effector from given joint coordinates (Denavit and Hartenberg 1955), and the result is always unique, even if the robot has kinematically redundant articulated joints. For parallel robots, this problem can be complex, and reconfigurability makes the situation worse. Small errors in the kinematic parameters due to disturbances in the measurement and other uncertainties from the movable anchor bases can lead to scenarios where no solutions or an infinite number of solutions are found. Meanwhile, the kinematic redundancy arising from the base mobility leads to high-dimensional nonlinear characteristics in the kinematic formulations. Furthermore, the distinctive feature of the cable robot—the unidirectional nature of the forces applied by the cables on the end effector (Oh and Agrawal 2005)—requires that the tension distribution algorithm must be implemented to ensure the positivity of cable tension (Borgstrom et al. 2009), which will exert constraints on the solving of the forward kinematics problem. In practical applications, ill-suited pose feedback from the forward kinematics may lead to dangerous and devastating situations. For instance, the platform may rush or crash into an undesired position, the driving cable may tear apart abruptly, or the flying platform may drop suddenly. These incidents can cause severe damage to

the robot and pose significant safety risks. Consequently, achieving a stable and robust solution for the forward kinematics of CCRobot-S is a significant challenge.

1.1 | State of the Art

Extensive research has been conducted on the forward kinematics problem in the context of various parallel robot configurations. In spite of reported successful solutions, scholars persist in exploring diverse methodologies within this domain. Existing forward kinematic analysis methodologies can be broadly categorized into five approaches: 1) the algebraic approach, 2) the analytical approach, 3) the numerical method, 4) optimization techniques, and 5) neural network-based strategies.

The *algebraic approach* represents a kinematic mapping technique wherein constraints are reformulated into a univariate polynomial. Husty (Husty 1996) presented an algorithm for addressing general kinematics, utilizing kinematic mapping (Bottema and Roth 1990) to transmute kinematic equations into a projective space. This transformation offers valuable insights into the architecture of the underlying mathematical challenge. Within this projective space, the equations assume an algebraic form, culminating in a polynomial of degree 40. Regrettably, the algebraic approach appears to be unsuitable for real-time applications. Moreover, the manipulation of 40th-order polynomials necessitates highly specialized numerical methods, with numerical stability emerging as a critical concern.

The *analytical approach* seeks to derive a closed-form expression for solving the forward kinematics problem. While computationally efficient, this method is typically only applicable to simple parallel robots with low degrees of freedom, as it depends on specific, non-generic geometries (Bosscher et al. 2007; Yang et al. 2004; Ferraresi et al. 2004). For such specialized geometries, unique structural characteristics can be leveraged. Yang et al. (2004) proposed an intriguing method for the kinematic analysis of a planar robot with three degrees of freedom and four cables. By capitalizing on the over-constrained nature of this mechanism, a fourth-order polynomial was formulated, offering a fundamentally closed-form solution. Ferraresi et al. Ferraresi et al. (2004) detailed a closed-form solution for a suspended cable robot configured in a 6-3 setup, where the robot's geometry aligns with the Simplified Symmetric Manipulator (SSM) design, a well-recognized configuration in traditional parallel robotics. However, in practical applications, Reconfigurable Cable-Driven Parallel Robots (R-CDPRs) seldom exhibit such non-generic geometric attributes, given that their geometric configurations are dynamically reconfigurable through the continuous adjustment of cable attachment points.

The *numerical method* employs numerical techniques such as the Newton-Raphson or Levenberg-Marquardt algorithms, commencing with an initial guess, to iteratively solve the forward kinematics problem. It is evident that the majority of objectives under study necessitate numerical solutions due to the complexity of formulating analytical solutions for over-constrained nonlinear equations (Jeong et al. 1999; Miermeister et al. 2013; Pott 2010). However, starting with an unconstrained initial guess poses risks of slow convergence rates, trapping in local optimums, or incorrect solutions during the iterative search process. Meanwhile, iterative solvers have the drawback that their region of convergence is limited and difficult to predict. If multiple solutions exist, it is difficult, but possible (Merlet 2004) to find all of them, and how to choose the correct solution is still a tricky problem. Additionally, these iterative search methods treat the cables as rigid links, and the tension condition of the cables is not considered. This treatment can also incur a wrong solution. Non-convergences or incorrect solutions mean wrong feedback poses in the control loop, which may lead to dangerous and devastating situations. For instance, the platform may rush or crash into an undesired position, the driving cable may tear apart abruptly, or the flying platform may drop suddenly. These incidents can cause severe damage to the robot and pose significant safety risks.

Optimization represents an alternative methodology for ascertaining the solution to the forward kinematics problem, wherein the roots of the constraint equations must be approximated as accurately as feasible. These constraints are essentially distance equations, interpretable as the squared distance from an exact solution. Thus, summing up the errors from each constraint leads to a least-squares problem. In the study by Santos et al. Santos and Gouttefarde (2021), the forward kinematics is formulated as the problem of minimizing the error between the measured cable lengths and the cable lengths computed by the inverse kinematics, and QR decomposition is used to solve the linearized version of the nonlinear forward kinematics problem. Analogous to the numerical method, optimization techniques also suffer from the pitfall of requiring an initial guess, which can result in issues such as oscillating optimal solutions and entrapment in local optima.

The *neural network-based strategy* is a more exotic approach that employs neural networks for addressing forward kinematics. The Artificial Neural Network (ANN) (Sharkawy and Khairullah 2023; Geng and Haynes 1991; Schmidt et al. 2014; Ghasemi et al. 2010; Sancak et al. 2023; Hu et al. 2025; Tao et al. 2025) is designed as an approximation function to predict the pose of the end-effector given the cable lengths. The network is trained such that cable length data serves as the input, with the corresponding end-effector pose data as the output. Researchers have explored neural networks to solve the forward kinematics problem. Geng and Haynes Geng and Haynes (1991) made initial efforts to employ the standard Multi-Layer Perceptrons (MLP) with Back-Propagation (BP) and cerebellar model arithmetic computer, yet their effort was limited to proposing the use of neural networks to solve the forward kinematics problem, rather than actually solving the problem accurately. Ali Ghasemi et al. Ghasemi et al. (2010) also applied neural networks to solve the forward kinematics problem of cable robots. Although neural network models can offer approximations to the forward kinematics solution, the accuracy of the predicted pose remains suboptimal, and the scale of data processed can reach millions, thus demanding substantial computational resources.

An alternative approach to address this challenge involves employing an exteroceptive sensor to capture the pose of the end-effector. Examples of such sensors include motion-capture systems, laser position sensors, or multi-camera global positioning systems (Lytle et al. 2004). However, deploying these exteroceptive measurement systems in large-scale, unstructured field environments is often impractical. Consequently, utilizing

cable lengths and subsequently inferring the end-effector pose through forward kinematics presents a more viable solution.

An optimal solution for addressing the challenge of the forward kinematics problem should not only be easy to derive but should also offer benefits such as reduced computational resource consumption, high precision, robustness, rapid real-time execution, and potentially, hardware integration. Although the five categories of approaches aforementioned have been validated on fixed-base CDPR robots, they encounter fundamental limitations when applied to reconfigurable CDPR systems (R-CDPRs)—CCRobot-S.

- **Algebraic approach** suffers from exponential growth of polynomial degree with the number of movable anchor bases in the CCRobot-S system, making real-time implementation infeasible.
- **Analytical approach** is typically applicable only to simple parallel robots with low degrees of freedom, as it depends on specific, non-generic geometries. However, the geometry of CCRobot-S is time-varying and generic.
- **Numerical method** is computationally intensive and requires an initial guess within an a priori unknown convergence basin. As such, whether the algorithm converges largely depends on the selection of initial values. Movable anchor bases of CCRobot-S enlarge the search space and drastically reduce the success rate.
- **Optimization** inherits the same sensitivity to initial guesses and can oscillate between local minima induced by redundant reconfiguration parameters of CCRobot-S.
- **Neural network-based strategy** typically cannot provide an accurate solution for the forward kinematics problem with small datasets. However, large datasets can lead to non-real-time execution.

To the best of the authors' knowledge, no comprehensive approach has yet been proposed for solving the forward kinematics problem of R-CDPRs that can overcome the aforementioned drawbacks.

1.2 | Contribution

The advent of Embodied Artificial Intelligence (EAI) represents an evolution in the field of robotics. The integration of EAI in robotics has become increasingly important, particularly for field robots operating in dynamic and unstructured environments. This approach transcends traditional computational intelligence by incorporating a physical aspect that is vital for robots to operate effectively in real-world settings.

To achieve high pose predicted accuracy and enhance motion robustness, we propose a novel cascaded strategy to effectively tackle the forward kinematics problem by leveraging the principles of EAI. The core concept involves utilizing a deep learning-based neural network by learning from physical or simulated interactive experiences of CCRobot-S to provide approximate solutions as initial values for a numerical method, thereby ensuring rapid convergence and a unique solution mapping. The neural net is used to quickly provide outputs that are estimated for suitable inputs to the numerical method. Starting with the suitable initial values that are near the real solution provides a "head start" and enables fast convergence for the numerical method with a reduced number of iterations. As a result, achieving the solution of the forward kinematics in real-time is possible. Moreover, by collecting sample data through physical or simulated interactive experiences of CCRobot-S in the deep learning-based neural network, we can guarantee that the forward kinematics solutions are limited to a single outcome instead of having numerous solutions. This achieves a one-to-one mapping from the joint space to the task space.

The contribution of this paper is manifold and can be summarized as follows.

- A novel Collaborative Climbing Robot Squad (CCRobot-S) is presented, and it holds promise for practical applications. Moreover, data can be collected from physical interactive experiences with the stay cables using the CCRobot-S robotic system.
- A novel cascaded strategy is proposed to address the forward kinematics problem of the CCRobot-S robotic system. To the best of our knowledge, this marks the first application of a lightweight deep learning-based model augmented with a numerical method to the Reconfigurable Cable-Driven Parallel Robots (R-CDPR) architecture.
- In contrast to existing methods, the proposed approach employs a deep learning-based neural network to optimize the initial guess of the numerical method, yielding an approximation that is acceptably close to the true solution. Consequently, the proposed method effectively mitigates the risk of non-convergence and incorrect solutions, thereby enhancing the task robustness and reliability of the robotic system. These improvements, in turn, enhance confidence in deploying the system for field applications.
- The proposed cascaded strategy with EAI has been designed and implemented. Extensive evaluations and experiments were conducted to verify its effectiveness. The results demonstrate the feasibility of using the cascaded strategy to solve the forward kinematics problem.

1.3 | Structure of the Paper

The structure of the remaining sections in this paper is outlined as follows. Section 2 presents the problem formulation and provides the basics of kinematic analysis. Section 3 details the proposed cascaded strategy with EAI. Section 4 discusses the performance evaluations and experiments based on this approach. Finally, conclusions and future work are presented in Section 5.

2 | Problem Formulation

As shown in Figure 1, the innovative climbing system, CCRobot-S, has been developed and demonstrated. The CCRobot-S system comprises four cable-climbing robots that serve as movable anchor bases on the stay cables, along with a flying platform capable of accommodating a robotic manipulator, end effectors, specialized tools, or sensors for stay cable

inspection and maintenance. Each movable anchor base is connected to the flying platform via a DYNEEMA® cable.

Similar to how a spider creates a large tensile structure between fixed attachments (Kovac 2016), the CCRobot-S system forms a parallel-type, cable-driven manipulation structure. However, unlike traditional tensile structures with fixed attachments, the counterparts in the CCRobot-S system are movable, providing enhanced flexibility and adaptability.

Figure 2 illustrates the general model and configuration of the CCRobot-S system. In the notation, the superscripts l and g denote the local frame of the flying platform and the global frame, respectively.

The flying platform is a 3-DoF system (with a pose comprising translation and rotation) driven by four cables. The generalized coordinates are expressed as vectors:

$$\mathbf{k} = [\mathbf{p}^{\mathbf{T}}\mathbf{r}^{\mathbf{T}}]^{\mathbf{T}} \in \mathbb{R}^3, \tag{1}$$

where $\mathbf{p}^{\mathbf{T}} \in \mathbb{R}^2$ and $\mathbf{r}^{\mathbf{T}} \in SO(1)$ denote the position and orientation of the flying platform, respectively. The Cartesian translation of the movable anchor base relative to the global coordinate frame $\{\mathfrak{g}\}$ is given by:

$$^{g}\mathbf{A}_i = [A_{ix} \quad A_{iy}]^{\mathbf{T}} \quad (i = 1, 2, 3, 4), \tag{2}$$

and its orientation is correlated with the inclination angle of the stay cable $\boldsymbol{\alpha}$ or $\boldsymbol{\beta}$. The cable attachment location on the flying platform relative to the local coordinate frame $\{\mathfrak{l}\}$ is represented by:

$$^{l}\mathbf{B}_i = [B_{ix} \quad B_{iy}]^{\mathbf{T}} \quad (i = 1, 2, 3, 4). \tag{3}$$

Since the anchor base is movable and constrained along the stay cable (a one-dimensional reconfiguration), the reconfigurable points $^{g}\mathbf{A}_i$ can be modeled as:

$$^{g}\mathbf{A}_i = [A_{ix}(u_i) \quad A_{iy}(u_i)]^{\mathbf{T}} \quad (i = 1, 2, 3, 4), \tag{4}$$

where u_i is a variable; consequently, the reconfiguration path of the movable anchor base can be described by a polynomial function.

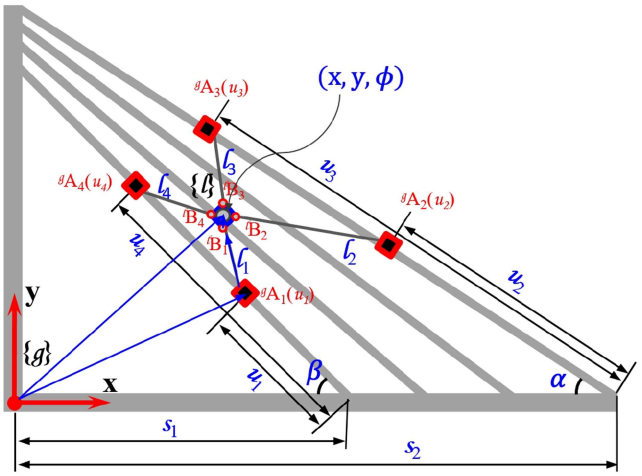

FIGURE 2 | The general model and configuration of the CCRobot-S system. [Color figure can be viewed at wileyonlinelibrary.com]

By applying a vector loop to the CCRobot-S model, as shown in Figure 2, the closure constraint can be expressed as:

$$\mathbf{BA}_i(u_i) + {}^{g}_{l}\mathbf{R}(\mathbf{r})\,{}^{l}\mathbf{B}_i + \mathbf{p} - {}^{g}\mathbf{A}_i(u_i) = 0;\ i = 1, 2, 3, 4, \tag{5}$$

where ${}^{g}_{l}\mathbf{R}(\mathbf{r})$ is the rotation matrix between coordinate frames $\{\mathbf{g}\}$ and $\{\mathbf{l}\}$.

For the motion of CCRobot-S, the position of the movable anchor base is changeable. With the introduction of the linear joints of the movable anchor bases, the system becomes kinematically redundant. The segment $\mathbf{BA}_i$ of the cable i can be expressed as

$$\mathbf{BA}_i(u_i) = {}^{g}\mathbf{A}_i(u_i) - {}^{g}_{l}\mathbf{R}(\mathbf{r})\,{}^{l}\mathbf{B}_i - \mathbf{p};\ i = 1, 2, 3, 4. \tag{6}$$

Furthermore, this can be rewritten as polynomial functions with variable u_i as

$$\mathbf{BA}_i(u_i) = \begin{bmatrix} A_{ix}(u_i) - x \\ A_{iy}(u_i) - y \end{bmatrix} - \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} B_{ix} \\ B_{iy} \end{bmatrix}, \tag{7}$$

where $\mathbf{p} = [x, y]^{\mathbf{T}}$, $\mathbf{r} = [\phi]$.

Meanwhile,

$$\begin{cases} A_{1x}(u_1) = s_1 - u_1\cos\beta \\ A_{2x}(u_2) = s_2 - u_2\cos\alpha \\ A_{3x}(u_3) = s_2 - u_3\cos\alpha \\ A_{4x}(u_4) = s_1 - u_4\cos\beta \end{cases}. \tag{8}$$

Substituting Equations (8) into (7), we receive four nonlinear coupled equations that form an under-determined system. As such, the mapping between the pose $\mathbf{k} = [\mathbf{p}^{\mathbf{T}}\mathbf{r}^{\mathbf{T}}]^{\mathbf{T}} \in \mathbb{R}^3$ of the flying platform in the operational space and the length of the cables $\mathbf{L} \in \mathbb{R}^4$ as well as the robot configurations $\mathbf{U} \in \mathbb{R}^6$ in the configuration space is identified, where the cables length parameters $\mathbf{L} = [l_1, l_2, l_3, l_4]^{\mathbf{T}}$, $l_i = \|AB_i\|$ $(i = 1, 2, 3, 4)$ ($\|\|$ is the Euclidian norm), the reconfigurable parameters $\mathbf{U} = [u_1, u_2, u_3, u_4, \alpha, \beta]^{\mathbf{T}}$, here, it needs to clarify that in our climbing scenario of the cable-stayed bridge, s_1 and s_2 are constants while α and β are variables since they are changing with different cables.

For the forward kinematics function, a mapping $\varphi^{\mathrm{FK}} : \mathbb{R}^{10} \to \mathbb{R}^3$ in the following form

$$\mathbf{k} = \varphi^{\mathrm{FK}}(\mathbf{q}), \tag{9}$$

is sought, where $\mathbf{k} = [x, y, \phi]^{\mathbf{T}} \in \mathbb{R}^3$ is the state of the task space (flying platform position and orientation), $\mathbf{q} = [l_1, l_2, l_3, l_4, u_1, u_2, u_3, u_4, \alpha, \beta]^{\mathbf{T}} \in \mathbb{R}^{10}$ is the combining of the joint state, including the cable length variables and the reconfiguration variables.

It must be underlined that computing this mapping is, in general, very complicated due to the high-dimensional nonlinear characteristics. Meanwhile, the forward kinematic transmission functions include reconfigurable variables, leading

to multi-variable under-determined equations. The computational complexity for solving the forward kinematics increases significantly with the number of reconfigurable cable attachment points. Specifically, the computational complexity is $O(d^r)$ where d is the discretization number and r is the number of reconfigurable cable attachment points.

3 | Cascaded Strategy With EAI

As discussed previously, the forward kinematics problem for the reconfigurable parallel-type cable-driven climbing robot is highly complex due to its reconfigurable coupling, nonlinearity, computational complexity, and the possibility of non-convergence. In this work, we propose a bi-level optimization framework that integrates lightweight deep learning with the numerical method, addressing the intractability of forward kinematics for Reconfigurable Cable-Driven Parallel Robots (R-CDPRs). This framework leverages Embodied Artificial Intelligence (EAI) to learn from physical/simulated interactions, providing a theoretically grounded approach to overcome the high-dimensional nonlinearity of traditional methods. This cascaded strategy is built on two key theoretical foundations:

Neural Network as a Learned Prior Approximation: The lightweight deep learning model acts as a data-driven prior that captures the nonlinear mapping, providing approximate solutions that serve as initial values for the numerical method. Trained on data generated by the robot's kinematic model or through physical interactions, the neural network ensures that these initial values satisfy the kinematic constraints or physical constraints of the system. This addresses the theoretical challenge of non-uniqueness and high-dimensional nonlinearity in traditional methods.

Numerical Method for Refinement: The numerical method refines the neural network's output within a locally convex region, leveraging the theoretical property that small perturbations from the neural network's initial guess ensure the objective function of the numerical method is strictly convex, where the Hessian of the objective function is positive definite under cable tension constraints.

To provide an intuitive understanding of this cascaded strategy, we conceptualize the forward kinematics problem as a minimum searching problem for an error function with several local minima. As illustrated in Figure 3, an example of an error surface is presented. To locate the global minimum, the deep learning Neural Network (NN) model shifts the initial value into a feasible region, rather than relying on a random guess. With this approximate initial value, we can then apply the numerical method to search for a more accurate solution, thereby ensuring convergence and robustness.

3.1 | Deep Learning-Based Neural Network

In this paper, we are dealing with the robotic forward kinematics problem, specifically predicting the position and orientation of the flying platform based on given cable lengths and robotic configurations. The task is fundamentally a regression problem where the model learns to map a set of input features to a set of continuous output values. Our objective is to establish

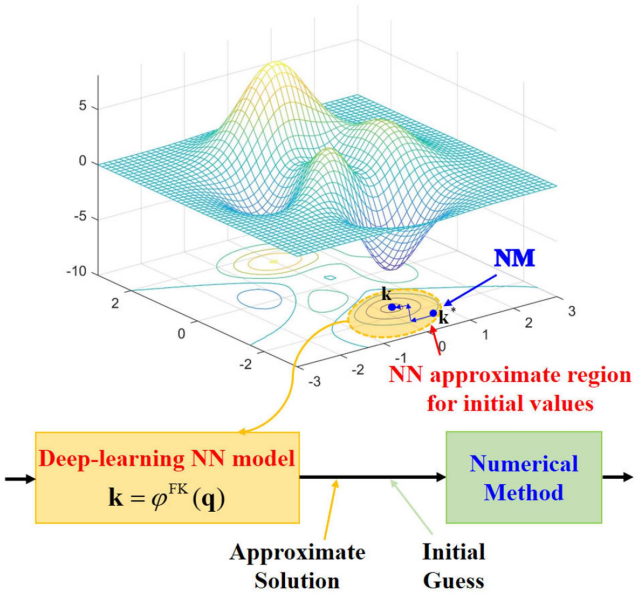

FIGURE 3 | Intuitive interpretation of the cascaded strategy for solving forward kinematics. Step one: the original data is fed into the lightweight deep learning Neural Network (NN) model. The output of this model serves as the initial value for the Numerical Method (NM) optimization. This initial value is strategically positioned within an approximate region that is close to the optimal solution. Step two: the numerical method is then employed to refine the solution, achieving a higher degree of stabiity and accuracy. [Color figure can be viewed at wileyonlinelibrary.com]

a mapping from the configuration space (cable lengths and reconfigurable parameters) to the task space (pose of the flying platform). Since the Multi-Layer Perceptron (MLP) architecture with Back-Propagation (BP) is well-suited for learning complex, nonlinear mappings between input and output spaces, we employed this architecture as the core of our deep learning-based approach. The deep learning-based neural network approach comprises three key components: dataset preparation, training configuration, and solution inference. Figure 4 shows the processes of the deep learning-based neural network in solving the forward kinematics of CCRobot-S.

3.1.1 | *Dataset Preparation*

Here, we employed two distinct methods for data collection. The first method involved learning from physical interactive experiences with the stay cables of the CCRobot-S robotic system. The second method was based on simulations derived from the inverse kinematics of the CCRobot-S system.

In this study, a portion of the data was obtained from real interactive experiments conducted with the stay cables of the CCRobot-S robotic system. We adopt encoders integrated into the wheels of the movable anchor bases to perceive and record the reconfigurable parameters $\mathbf{U} = [u_1, u_2, u_3, u_4, \alpha, \beta]^\mathbf{T}$, where α and β are the actual values of the inclination angles of the bridge cables, and we also use the encoders integrated into the winch of the movable anchor bases to perceive and measure the cables length parameters $\mathbf{L} = [l_1, l_2, l_3, l_4]^\mathbf{T}$. By recombining these data points, the input values for the training model are set as

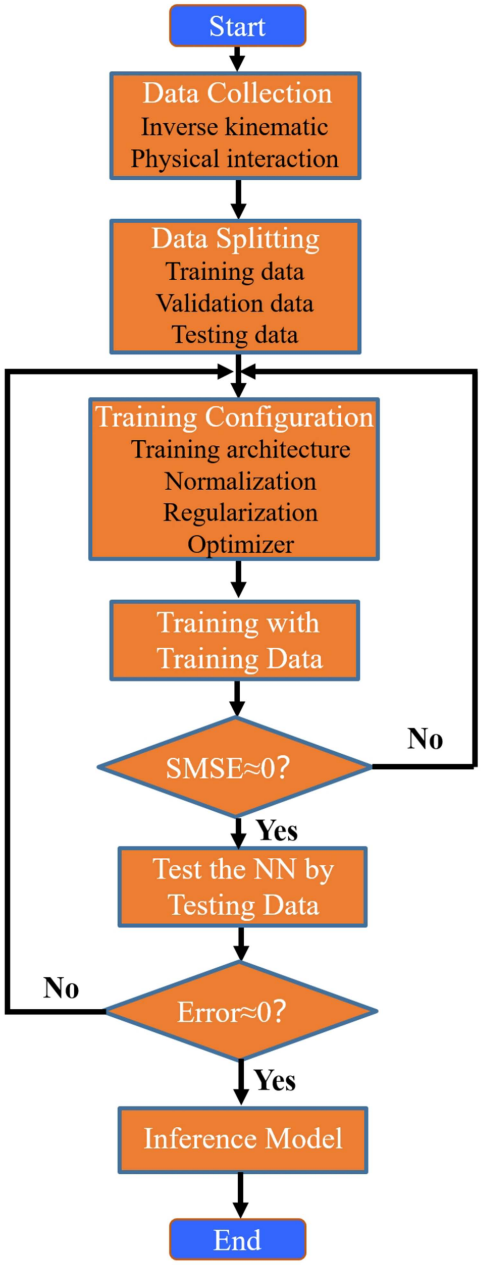

FIGURE 4 | The processes of the deep learning-based neural network in solving the forward kinematics of CCRobot-S. [Color figure can be viewed at wileyonlinelibrary.com]

$$\mathbf{q} = [l_1, l_2, l_3, l_4, u_1, u_2, u_3, u_4, \alpha, \beta]^{\mathbf{T}} \in \mathbb{R}^{10}, \tag{10}$$

in the configurational space, and the output of the forward kinematics model is set as

$$\mathbf{k} = [x, y, \phi]^{\mathbf{T}} \in \mathbb{R}^3 \tag{11}$$

in the operational space.

While the method of learning from physical interaction using the CCRobot-S robotic system offers an intuitive sense and ensures a one-to-one mapping from the joint space to the task space, the process of collecting thousands of data points through this method is labor-intensive and time-consuming.

Compared to solving forward kinematics, the transformation of inverse kinematics is straightforward and can be easily implemented based on the known configuration of CCRobot-S. Moreover, the solution for inverse kinematics is guaranteed to exist and be unique. Consequently, we employ inverse kinematics to prepare the remainder of the dataset in this study. First, we determine the robot configuration according to the adjustment of the reconfigurable parameters $u_1, u_2, u_3, u_4, \alpha, \beta$. Here, α and β are learned from the bridge model, while u_1, u_2, u_3, u_4 are discretely adjusted within the range of interest using iterative methods. This process allows us to explore all possible robot configurations. Next, we discretize the wrench-closure workspace for each robot configuration. Finally, each discrete point in the wrench-closure workspace is collected as part of the task space. Then, the discrete points were processed through the mathematical equation representing the inverse kinematics solution (Equation (7)), i.e., $\varphi^{\mathrm{IK}} : \{x, y, \phi, u_1, u_2, u_3, u_4, \alpha, \beta\} \rightarrow \{l_1, l_2, l_3, l_4\}$. By recombining these data points, the input values for the forward kinematics model are set as Equation (10).

To facilitate the study and accelerate the analysis of the training results, we selected a specific configuration to lower the data capacity to evaluate the deep learning neural network. A dataset of 360,000 samples was prepared, which allowed for efficient training and validation while maintaining sufficient representativeness for the evaluation. All of these data values are stored in a `.txt` file; the partial dataset is shown in Table 1. In the `.txt` file, the first ten elements of each row are designated as features (the first four elements represent the cable length, and the last six represent the reconfiguration parameters), while the last three elements are used as labels for the neural network.

3.1.2 | *Training Configuration*

In this study, we constructed a Multi-Layer Perceptron (MLP) to predict the pose status of the flying platform.

In deep learning, the number of hidden layers (depth) and the number of neurons per hidden layer (width) are core hyperparameters that determine a model's structure and performance. In this manuscript, we refer to the following principles for choosing the number of hidden layers and the number of neurons per hidden layer.

- **Match Task Complexity:** using 1–3 layers, 0–256 neurons per layer for simple tasks (sample size <10,000); 5–20 layers, 64–512 neurons per layer for moderately complex tasks (10,000 < sample size < 1,000,000); and 20–100 layers, 128-1024 neurons per layer for highly complex tasks (sample size > 1,000,000).
- **Domain Empirical Values:** fully connected layers commonly use powers of 2 (e.g., 32, 64, 128, 256) for more efficient memory allocation and data processing;
- **Shallow and Wide Models (few layers, many neurons per layer):** excelling at processing flat features (e.g., tabular data) and have low training difficulty. Based on these core principles, the architecture of the deep learning-based neural network was systematically determined through a grid search over network depth and width, aiming to balance prediction accuracy and computational efficiency. We evaluated networks with different numbers of hidden layers and varying neuron counts. After the empirically tuned

TABLE 1 | Partial dataset.

l_1 (cm)	l_2 (cm)	l_3 (cm)	l_4 (cm)	u_1 (cm)	u_2 (cm)	u_3 (cm)	u_4 (cm)	α (degree)	β (degree)	x (cm)	y (cm)	ϕ (degree)
84.2846	99.4276	34.3562	0.1402	0	74.0	164.1	88.3	33. 1°	12. 5°	−59.5	62.5	44. 5°
84.2671	99.4330	34.3736	0.1267	0	74.0	164.1	88.3	33. 1°	12. 5°	−59.5	62.5	45. 0°
83.5952	98.8240	34.3478	0.7260	0	74.0	164.1	88.3	33. 1°	12. 5°	−59.0	62.0	44. 0°
83.5775	98.8292	34.3646	0.7394	0	74.0	164.1	88.3	33. 1°	12. 5°	−59.0	62.0	44. 5°
83.5600	98.8345	34.3816	0.7534	0	74.0	164.1	88.3	33. 1°	12. 5°	−59.0	62.0	45. 0°
83.9498	98.9350	33.9881	0.6217	0	74.0	164.1	88.3	33. 1°	12. 5°	−59.0	62.5	44. 0°
83.9323	98.9403	34.0051	0.6175	0	74.0	164.1	88.3	33. 1°	12. 5°	−59.0	62.5	44. 5°
83.9148	98.9457	34.0223	0.6145	0	74.0	164.1	88.3	33. 1°	12. 5°	−59.0	62.5	45. 0°
84.2711	99.0593	33.6665	0.8292	0	74.0	164.1	88.3	33. 1°	12. 5°	−59.0	63.0	45. 0°
82.8881	98.2272	34.3711	1.4247	0	74.0	164.1	88.3	33. 1°	12. 5°	−58.5	61.5	44. 0°
82.8704	98.2322	34.3876	1.4400	0	74.0	164.1	88.3	33. 1°	12. 5°	−58.5	61.5	44. 5°
82.8529	98.2374	34.4042	1.4557	0	74.0	164.1	88.3	33. 1°	12. 5°	−58.5	61.5	45. 0°
83.2427	98.3363	34.0043	1.1786	0	74.0	164.1	88.3	33. 1°	12. 5°	−58.5	62.0	44. 0°
83.2252	98.3414	34.0209	1.1867	0	74.0	164.1	88.3	33. 1°	12. 5°	−58.5	62.0	44. 5°
83.2077	98.3468	34.0377	1.1955	0	74.0	164.1	88.3	33. 1°	12. 5°	−58.5	62.0	45. 0°

process, the final configuration consists of four hidden layers with 64, 128, 256, and 512 neurons, respectively.

As such, the structure of this neural network is illustrated in Figure 5. The MLP consists of six layers, including four hidden layers with varying numbers of neurons (64, 128, 256, and 512). The specific architecture is as follows:

- **Input layer:** Ten neurons (corresponding to the ten variables of the cable lengths and reconfigurable parameters $\{l_1, l_2, l_3, l_4, u_1, u_2, u_3, u_4, \alpha, \beta\}$).
- **Hidden layer:**

  Hidden Layer 1: 64 units
  Hidden Layer 2: 128 units
  Hidden Layer 3: 256 units
  Hidden Layer 4: 512 units

  The number of neurons in the hidden layers was determined based on the learning results. Each hidden layer is followed by a Sigmoid activation function to introduce nonlinearity into the model. The weights of the network are initialized using the Xavier initialization method, and the biases are initialized to a small constant value (0.1). This initialization strategy helps in maintaining the scale of gradients and signals as they propagate through the network, which is crucial for the effective training of deep networks.
- **Output layer:** Three neurons (corresponding to the three pose states of the flying platform $\{x, y, \phi\}$).

The quadratic cost function can be expressed as:

$$C = \frac{1}{2}\sum_{i=1}^{n}(k_i - \hat{k}_i)^2 + \lambda\sum_{i=1}^{n} w_i^2 \tag{12}$$

where k_i is the i-th feature of the sample, and $\hat{k}_i$ is the predicted value of the i-th feature of the sample. The activation function used is the Sigmoid function:

$$f(x) = \frac{1}{1 + e^{-\sigma \cdot x}}, \tag{13}$$

following the approach in seng Yee and Lim (1997). However, the use of the sigmoid activation function has been proven to be inefficient due to the gradient vanishing problem. Specifically, the gradient near the endpoints of the sigmoid function becomes very small, which can adversely affect the efficiency and effectiveness of the training process.

Regularization: To address this issue, we employ regularization techniques in the learning process. These techniques are known to mitigate the gradient vanishing problem and accelerate the training process. The second term in Equtaion (12) represents an L_2-norm regularization technique, which is used to prevent overfitting. This term is the sum of the squares of the magnitudes of the model's weights, where w_i is the i-th weight in the model, and λ is the regularization strength – a hyperparameter that controls the amount of regularization. L_2-norm regularization helps keep the weights small, leading to a simpler model that generalizes better to new, unseen data.

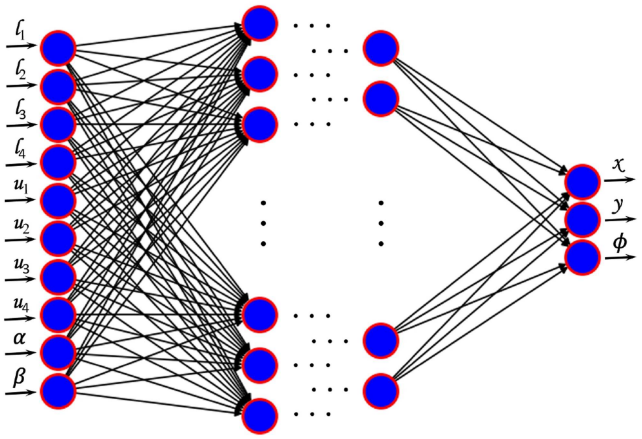

FIGURE 5 | The training configuration of the deep learning neural network. [Color figure can be viewed at wileyonlinelibrary.com]

Optimizer: Meanwhile, we select the Adam optimizer for training due to its fast convergence and adaptive learning rate properties. These features enable efficient optimization by dynamically adjusting the learning rate for each parameter based on the magnitude of the gradient, thereby improving both the speed and stability of the training process.

Normalization: In this case, the ten input features have significantly different value ranges, and some input features are much larger than others. During training, the neural network will primarily adjust the weights for these larger features while potentially ignoring the smaller ones. To mitigate this issue, batch normalization technology is adopted in this study. For a mini-batch: $\mathcal{B} = \{\mathbf{q}^1, \mathbf{q}^2, ..., \mathbf{q}^m\}$, we set

$$\begin{cases} \dfrac{1}{m}\sum_{n=1}^{m} q_i^n \rightarrow \mu_i^{\mathcal{B}} \\ \dfrac{1}{m}\sum_{n=1}^{m} \left(q_i^n - \mu_i^{\mathcal{B}}\right)^2 \rightarrow \left(\sigma_i^{\mathcal{B}}\right)^2 \\ \dfrac{q_i^n - \mu_i^{\mathcal{B}}}{\sqrt{\left(\sigma_i^{\mathcal{B}}\right)^2 + \varepsilon}} \rightarrow \hat{q}_i^n \end{cases} \tag{14}$$

where $\mathbf{q} = \{l_1, l_2, l_3, l_4, u_1, u_2, u_3, u_4, \alpha, \beta\}$ is the input sample, m is the size of the mini-batch, and k_i^n is the i-th feature of the n-th sample. $\mu_i^{\mathcal{B}}$ is the mean of the i-th feature in the mini-batch $\mathcal{B}$, $(\sigma_i^{\mathcal{B}})^2$ is the variance of the i-th feature in the mini-batch $\mathcal{B}$, and ε is a small constant to prevent division by zero. Applying scaling and shifting:

$$net^n \leftarrow W\hat{q}^n + w_0 \equiv BN_{W,w_0}(q^n) \tag{15}$$

where net^n is the output of the n-th sample after batch normalization, W and w_0 are learnable parameters, $\hat{x}^n$ is the normalized sample vector, and $\text{BN}_{W,w_0}(x^n)$ represents the function that applies batch normalization. As such, the inputs are linearly rescaled to have zero mean and unit variance on the training set. The outputs are centered to have a zero mean on the training set. This process standardizes the data in a mini-batch to improve the stability and efficiency of model training.

It is still a challenge to set good hyperparameter values in the neural network. Here, we use the broad strategy to find good hyperparameter values by individually adjusting each hyperparameter, gradually improving performance. Since we do not have a priori knowledge of which hyperparameter needs to be adjusted, we have to try different hyperparameter values. To further speed up experimentation, we strip the network down to the simplest network to do meaningful learning. In that case, we're getting feedback in a fraction of a second, rather than once every ten seconds or so. That means we can more quickly experiment with other choices of hyperparameter, or even conduct experiments trialling many different choices of hyperparameter nearly simultaneously. Once we've explored to find an improved value for one hyperparameter, then we move on to find a good value for another hyperparameter. Then experiment with a more complex architecture with different hidden layers and neurons. Then adjust the hyperparameter values again. And so on, at each stage evaluating performance using our held-out validation data, and using those evaluations to find better hyperparameters.

After fine-tuning, the hyperparameters used in our neural network are as follows:

- Batch size: 128.
- Learning rate: 0.0001.
- Weight decay: 1e-5.
- Epochs: 300.

3.1.3 | *Solution Inference*

In this part, we develop a solution inference to predict the real pose of the end effector in the CCRobot-S system. As illustrated in Figure 6, we have trained the neural network and obtained the deep learning model. When we input a vector $\mathbf{q}$ into the model, it outputs a predicted label $\hat{\mathbf{k}}^*$. However, this is not the value we desire to obtain since the normalization processing during training alters the scale of the output. As a result, $\hat{\mathbf{k}}^*$ does not directly represent the real pose of the flying platform in the CCRobot-S coordinate system. To obtain an intuitive result, we conduct a denormalization process for the output of the neural network model. This process is the reverse of Equation (14), that is,

$$k_i^* = \hat{k}_i^* * \sigma_i^{\mathcal{B}} + \mu_i^{\mathcal{B}}. \tag{16}$$

In that case, after processing the approximate function, we obtain a real predicted output $\mathbf{k}^*$ representing the pose of the end effector.

3.2 | Numerical Method Optimization

After obtaining an approximate solution, we employ numerical method optimization to refine the solution and obtain a more precise estimate of the flying platform's pose. The cable lengths corresponding to the desired pose of the flying platform are denoted as $\boldsymbol{l}_o$, while the cable lengths derived from the inverse kinematic model (Equation (7), i.e., $\varphi^{\text{IK}} : \{x, y, \phi, u_1, u_2, u_3, u_4, \alpha, \beta\} \rightarrow \{l_1, l_2, l_3, l_4\}$) for a given pose $\mathbf{k}$ are represented as $\boldsymbol{l}(\mathbf{k})$.

The numerical method can be formulated as an optimization task aimed at minimizing the error between $\boldsymbol{l}_o$ and $\boldsymbol{l}(\mathbf{k})$. The objective function is defined as $\varsigma(\mathbf{k}) = \frac{1}{2}\|\boldsymbol{l}(\mathbf{k}) - \boldsymbol{l}_o\|_2^2$, where $\|\cdot\|_2$ denotes the Euclidean norm. The general formulation of the optimization problem is then

$$\begin{aligned} \mathbf{K}^* = \arg\min_{\mathbf{k}} \frac{1}{2}\left\|\boldsymbol{l}(\mathbf{k}) - \boldsymbol{l}_o\right\|_2^2 \\ \text{s.t.} \quad \boldsymbol{l}(\mathbf{k}) = \mathbf{BA}(\mathbf{k}) \\ \mathbf{k}_0 = \mathbf{k}^* \\ \mathbf{k} \quad \in \mathcal{W} \end{aligned}, \tag{17}$$

here, $\mathbf{k}_0$ denotes the initial value of the flying platform's pose, obtained from the neural network. The vector $\mathbf{BA}(\mathbf{k})$ is defined as $[\mathbf{BA}_1(\mathbf{k}), \mathbf{BA}_2(\mathbf{k}), \mathbf{BA}_3(\mathbf{k}), \mathbf{BA}_4(\mathbf{k})]^T$. Each component $\mathbf{BA}_i(\mathbf{k})$ can be derived from Equation (7), that is, $\varphi^{\mathrm{IK}} : \{x, y, \phi, u_1, u_2, u_3, u_4, \alpha, \beta\} \rightarrow \{l_1, l_2, l_3, l_4\}$.

Since the initial value $\mathbf{k}_0$ is an acceptable approximation close to the real value, we assume that the numerical method optimization process is carried out under the same configuration of the CCRobot-S, the parameters $u_1, u_2, u_3, u_4, \alpha, \beta$ are held constant and are known. $\mathcal{W}$ represents the current wrench-closure workspace under this configuration (Gouttefarde and Gosselin 2006).

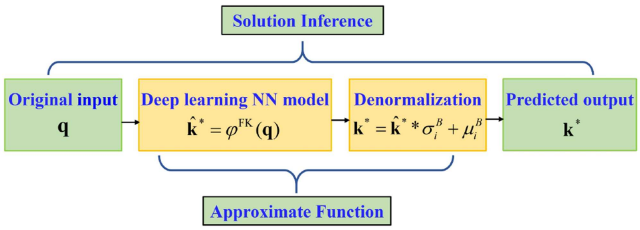

FIGURE 6 | Schematic diagram of the solution inference for the CCRobot-S system. [Color figure can be viewed at wileyonlinelibrary.com]

4 | Performance Evaluations and Experiments

In this section, we first evaluate the performance and effectiveness of the deep learning model applied to the R-CDPR system. We then compare the prediction accuracy of the cascaded strategy with that of the single neural network training approach. Furthermore, we present the robotic system implementation of CCRobot-S with the cascaded strategy. Finally, we also compare the convergence and robustness of the cascaded strategy with those of the Newton-Raphson algorithm-based numerical method.

4.1 | Evaluation of the Deep Learning NN

Here, we split the dataset values into three parts: training data, validation data, and testing data, with a ratio of 8:1:1. Specifically, 288,000 values are allocated for training, 36,000 values for validation, and 36,000 values for testing. The training results of the NN model are depicted in Figure 7. The image on the left of Figure 7 shows the model loss at each epoch for both the training data and the validation data. The minimum loss for the training data is almost close to 0, while the minimum loss for the validation data is near 0.05. The image on the right of Figure 7 shows the model accuracy trend for both the training data and the validation data, with the best accuracy values being 97.96% and 92.45%, respectively.

Given the prediction method, we can evaluate the quality of the neural network predictions in several ways. Perhaps the simplest is the squared error loss, where we compute the squared residual between the predicted value and the target value at each test point. This can be summarized by the Mean Squared Error (MSE), which is obtained by averaging over the test set. However, this quantity is sensitive to the overall scale of the target values. Therefore, it makes sense to normalize it by the variance of the target values in the test set to obtain the Standardized Mean Squared Error (SMSE). This normalization

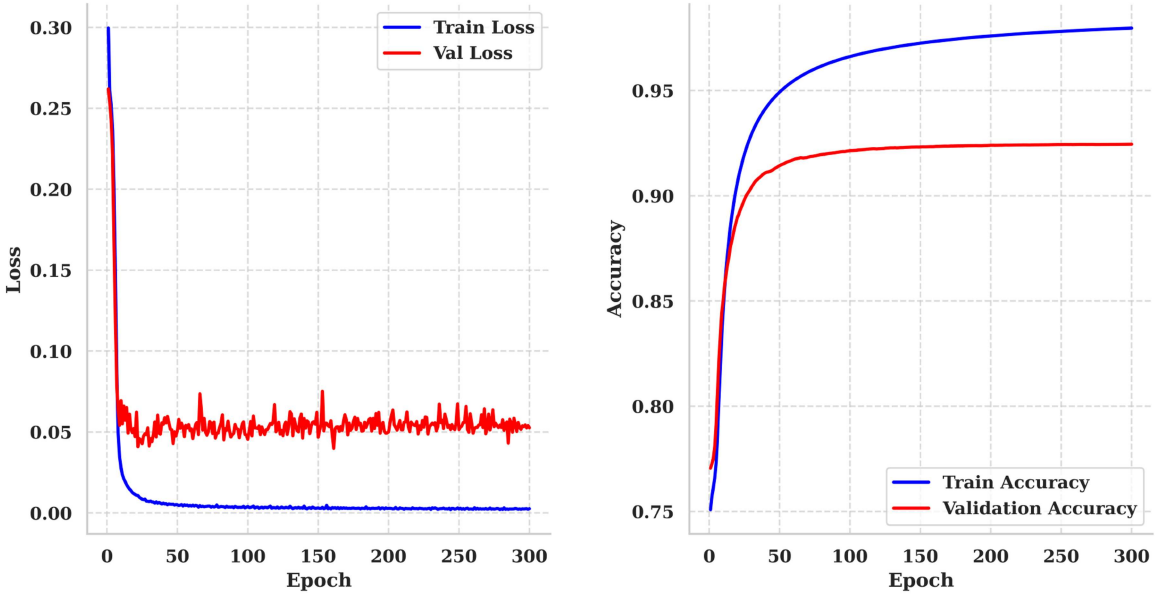

FIGURE 7 | Training and validation performance of the deep learning-based neural network: the model loss at each epoch and the model predicted accuracy trend. [Color figure can be viewed at wileyonlinelibrary.com]

TABLE 2 | Correlation evaluation of deep learning-based NN model.

Dataset	SMAE	SMSE	R^2
Training set	0.0315	0.0085	0.9915
Validation set	0.1199	0.0569	0.9431

ensures that the trivial method of guessing the mean of the training targets yields an SMSE of approximately 1.

Since we have normalized the training data, the SMSE value equals the MSE value in our case. Consequently, in this study, we focus solely on the SMSE values. Similarly, we also adopt the Standardized Mean Absolute Error (SMAE) to evaluate the model. The SMAE is calculated by dividing the Mean Absolute Error (MAE) by the standard deviation of the target values in the test set.

Meanwhile, the coefficient of determination R^2 score can provide more intuitive insights compared to metrics such as SMSE in regression analysis evaluations (Chicco et al. 2021). Therefore, the accuracy of the NN model in performing forward kinematics was also evaluated using the R^2 metric. Finally, the values of SMAE, SMSE, and R^2 are presented in Table 2.

According to Table 2, the R^2 values for the training and validation sets are 0.9915 and 0.9431, respectively. These values are close to 1, indicating a strong relationship between the input and output variables. The evaluation results demonstrate that the learning-based neural network method is effective in predicting forward solutions for the reconfigurable parallel-type cable-driven climbing robot squad.

It is important to highlight the limitations that exist in this method. Although we have set the training epoch as high as 300, the prediction accuracy for the validation data is 92.45%, which is not a desired value for precise position control of the flying platform. To address this issue, we have attempted to increase the complexity of the training model, such as by adding more hidden layers and increasing the number of neurons in the middle layers, in order to improve prediction accuracy. However, the inference time increases dramatically with an increase in model complexity. This is highly detrimental to real-time control of the CCRobot-S.

4.2 | Evaluation of the Prediction Accuracy

As aforementioned, the NN approximator had limitations in prediction accuracy under the condition of real-time execution. To address these limitations, we developed an additional numerical method optimization process to combine with the NN approximator as a cascaded control strategy, thereby improving prediction accuracy. To further clarify the performance of the cascaded strategy and evaluate its prediction accuracy, we present an example in which the four movable anchor bases are constrained to two stay cables, and the flying platform, driven and manipulated by the four parallel cables together, can move along the stay cable.

For the flying platform, the given trajectory in the task space is expressed by $\mathbf{k}(t) \in \mathbb{R}^3$, as shown in Equation (18). The time variable t ranges from 0 to 50 s. A linear motion is used for the trajectory, based on the task requirement that the flying platform travels along the stay cable.

TABLE 3 | Parameters for CCRobot-S model.

Parameters sheet			
Movable anchor base vector (cm)		Flying platform vector (cm)	
$\mathbf{u}_1$	0	${}^l\mathbf{B}_1$	$[-9.5\ \ -9.5]^T$
$\mathbf{u}_2$	74.0	${}^l\mathbf{B}_2$	$[9.5\ \ -9.5]^T$
$\mathbf{u}_3$	164.1	${}^l\mathbf{B}_3$	$[9.5\ \ 9.5]^T$
$\mathbf{u}_4$	88.3	${}^l\mathbf{B}_4$	$[-9.5\ \ 9.5]^T$
α	33. 1°	β	45°
s_1	12.5	s_2	23
Origin location of $\{\mathbf{g}\}$		$[0\ 0]^T$	

$$\mathbf{k}(t) = [2 - t,\ 0.8693t + 23.7107,\ 49°]^{\mathbf{T}}. \tag{18}$$

To simplify the study of the reconfiguration mechanisms and validate the effectiveness of the cascaded strategy, we selected a specific configuration by setting the reconfigurable parameters u_1, u_2, u_3, u_4, α, β. The specific parameter values are provided in Table 3.

In this process, we divided the entire desired trajectory into 50 points and derived the corresponding cable lengths using inverse kinematics (Equation (7)) for each point. We then used these cable lengths and the reconfigurable parameters to infer the predicted trajectory using both the deep learning-based neural network method alone and the cascaded strategy proposed in this study. Each moving iteration for the experiment was recorded.

Figure 8 presents the desired trajectory composed of 50 points (in blue), the predicted trajectory composed of 50 points derived from the deep learning-based neural network method alone (in black), and the predicted trajectory composed of 50 points derived from the cascaded strategy (in red). Here, the coordinate point represents the position $\mathbf{p}$ of the flying platform, and the orientation of the squares indicates the rotation $\mathbf{r}$ of the flying platform. It can be seen that the predicted trajectory derived from the cascaded strategy is closer to the desired trajectory than that derived from the deep learning-based neural network method alone. Additionally, the time required to implement the deep learning-based neural network method and the cascaded strategy was almost the same because the deep learning-based neural network provides an initial value that is close to the forward kinematic solution, which ensures quick convergence of the numerical method. As a result, only a small number of iterations are required to achieve the real solution during the numerical method optimization process.

To quantitatively analyze and compare the prediction accuracy, we statistically analyzed the errors within the 50 points, as shown in Figure 9. The average position errors of the flying platform in the X direction, Y direction, and Θ orientation predicted by the deep learning-based neural network method alone (green color) are 1.43 ± 1.88 mm, −2.72 ± 4.05 mm, and 2.32 ± 2.51°, respectively. In contrast, the average errors

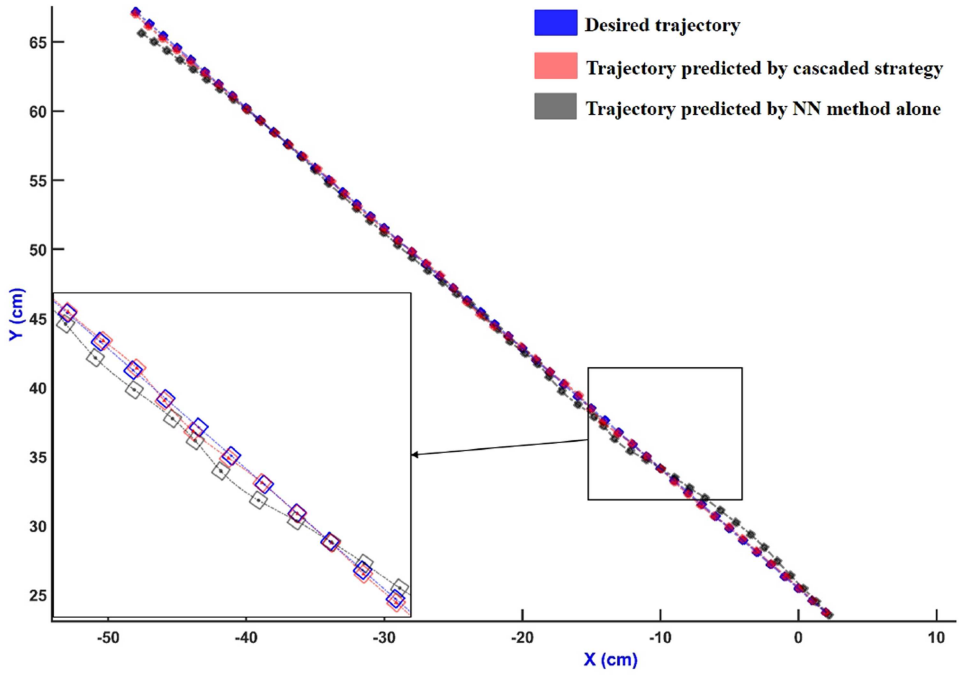

FIGURE 8 | The results of the predicted trajectories based on the deep learning-based neural network method alone (black color) and the proposed cascaded strategy (red color), and the comparison between them and the given trajectory (blue color). [Color figure can be viewed at wileyonlinelibrary.com]

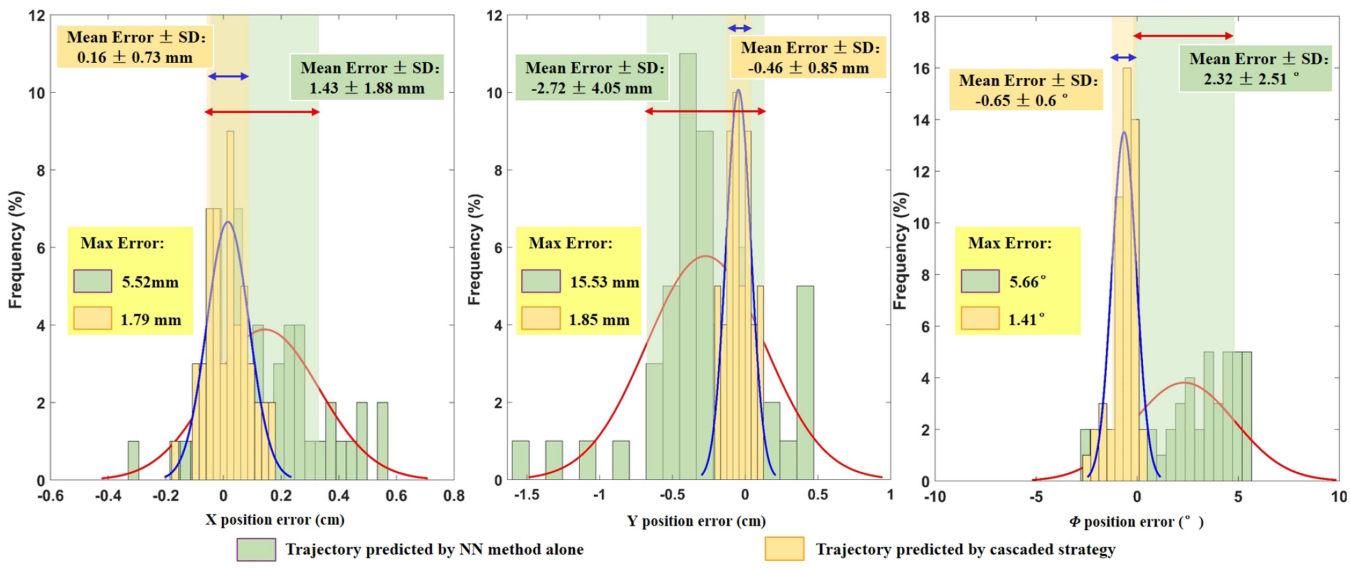

FIGURE 9 | Statistical pose errors of the flying platform in the X direction, Y direction, and Θ orientation, and the comparisons between the neural network method alone and the cascaded strategy. [Color figure can be viewed at wileyonlinelibrary.com]

predicted by the cascaded strategy (yellow color) were reduced to 0.16 ± 0.73 mm, -0.46 ± 0.85 mm, and $-0.65 \pm 0.6°$, respectively. The results clearly show that the cascaded strategy outperforms the neural network method alone in terms of prediction accuracy.

It is worth noting that when we control the system, we focus on the position $\mathbf{p}$ of the flying platform while omitting consideration of the rotation $\mathbf{r}$ due to its limited controllability and adjustability. Adjusting the orientation angle of the flying platform impacts the tension distribution significantly, leading to workspace contraction and a decrease in the maximum Tension Factor (TF) value, which is used to measure and evaluate the quality of the workspace of the flying platform. A higher TF contributes to the flying platform's stability, rigidity, and robustness. In practice, the desired range of orientation angles is typically narrow. To compensate for this disadvantage, we introduce a rotational joint on the end of the flying platform to control the yaw of the Euler angle of the flying platform in the mechanical design phase. With this DoF, the flying gripper can liberally adjust to the optimal angle to align with the inclination angle of the stay cable to be grasped without altering the tension distribution of the driving cables.

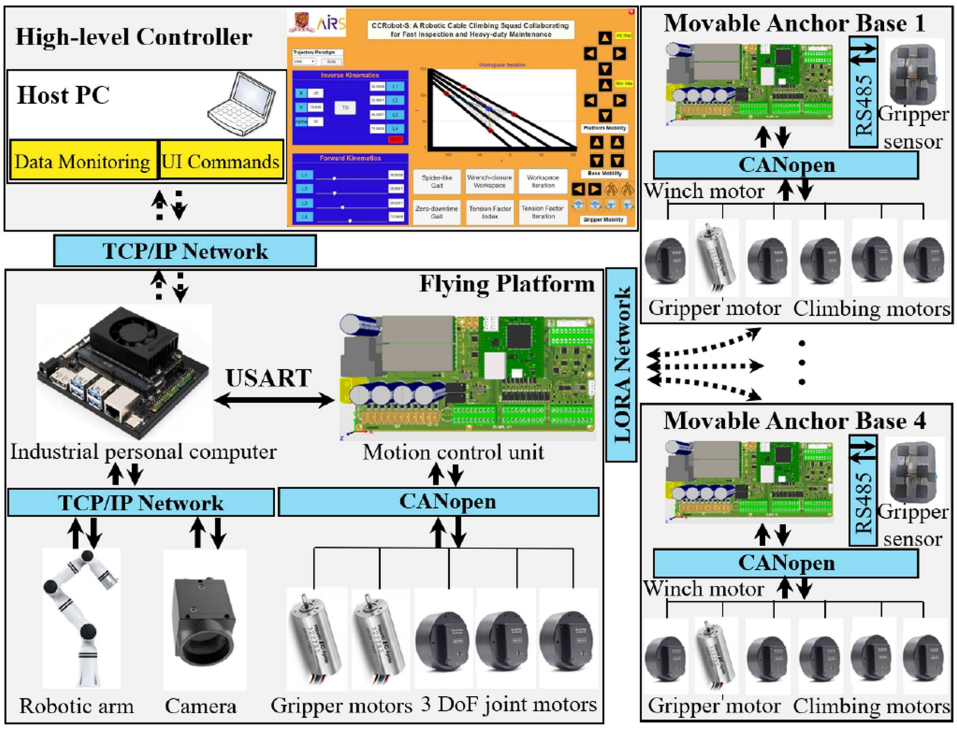

FIGURE 10 | The hardware architecture and electronic system of CCRobot-S. [Color figure can be viewed at wileyonlinelibrary.com]

4.3 | System Implementation

Figure 1 shows the CCRobot-S prototype attached to the bridge cables. The prototype comprises four cable-climbing robots serving as the movable anchor bases on the stay cables, along with a flying platform. Each movable anchor base is connected to the flying platform through a DYNEEMA® cable. Worm gearing was integrated into the transmission chain for the gripper to ensure safety during the experiments. Since the worm gearing is self-locking, the flying platform and the anchor bases could remain in their poses on the stay cables without energy consumption. In addition, brakes were used on the customized winches in case the flying platform fell due to a sudden power outage. Meanwhile, we prepared a multi-cable test platform for the experiments.

Before introducing the system implementation, the hardware architecture and electronic system of CCRobot-S that implements the prototype are presented as shown in Figure 10, which outlines the components and their connections within the CCRobot-S system. The architecture consists of two subsystems: a remote controller and an onboard controller that communicate with each other by Transmission Control Protocol/Internet Protocol (TCP/IP). The onboard controller is responsible for controlling five sub-robots: one flying platform and four movable anchor bases. The communication between the flying platform and the movable anchor bases is supported by a Long Range Wide Area Network (LoRaWAN), which facilitates robust one-to-many communication from a central host to multiple slaves simultaneously.

Based on the hardware architecture and electronic system of CCRobot-S, the flying platform can move continuously along multiple stay cables and retrieve images without the necessity to halt and adhere to the stay cable. Figure 11 illustrates the overall control system architecture. Forward kinematics is used to estimate the pose and velocity of the flying platform as feedback values in the control loop. In conventional methods, numerical schemes such as the Newton-Raphson or Levenberg-Marquardt algorithms are employed with an initial guess to solve the problem iteratively. In this architecture, a `.pth` file trained by the deep learning method is integrated into the control system, it offers an initial guess for the numerical method optimization. In addition, we tested the inference model 100 times on each of the CPU and GPU hardware. For one set of inputs, the minimum inference time of this model on CPU and on CUDA is 0.197 and 0.319 ms, respectively. The maximum inference time of this model on CPU and on CUDA is 0.560 and 0.670 ms, respectively. The average inference time of this model on CPU and on CUDA is 0.245 and 0.380 ms, respectively. For 100 sets of inputs, the minimum inference time of this model on CPU and on CUDA is 0.858 and 0.432 ms, respectively. The maximum inference time of this model on CPU and on CUDA is 1.693 and 0.931 ms, respectively. The average inference time of this model on CPU and on CUDA is 0.998 and 0.501 ms, respectively. This test result shows that the training model is suitable for real-time execution for the CCRobot-S system. As such, the deep learning model combined with numerical method optimization replaces the conventional numerical methods to provide feedback values to the Proportional-Derivative (PD) feedback loop, which implements the computation of the desired torque of the flying platform. Subsequently, based on the tension distribution control (Pott and Bruckmann 2013), we map the torque of the flying platform from the operational space to the desired cable lengths in the configurational space. It is important to note that the input data for the neural network model includes the encoded values $\{u_1, u_2, u_3, u_4\}$, the cable lengths $\{l_1, l_2, l_3, l_4\}$, and $\{\alpha, \beta\}$, which are determined by the initial configuration of the CCRobot-S system. The experimental task involved moving along the cables

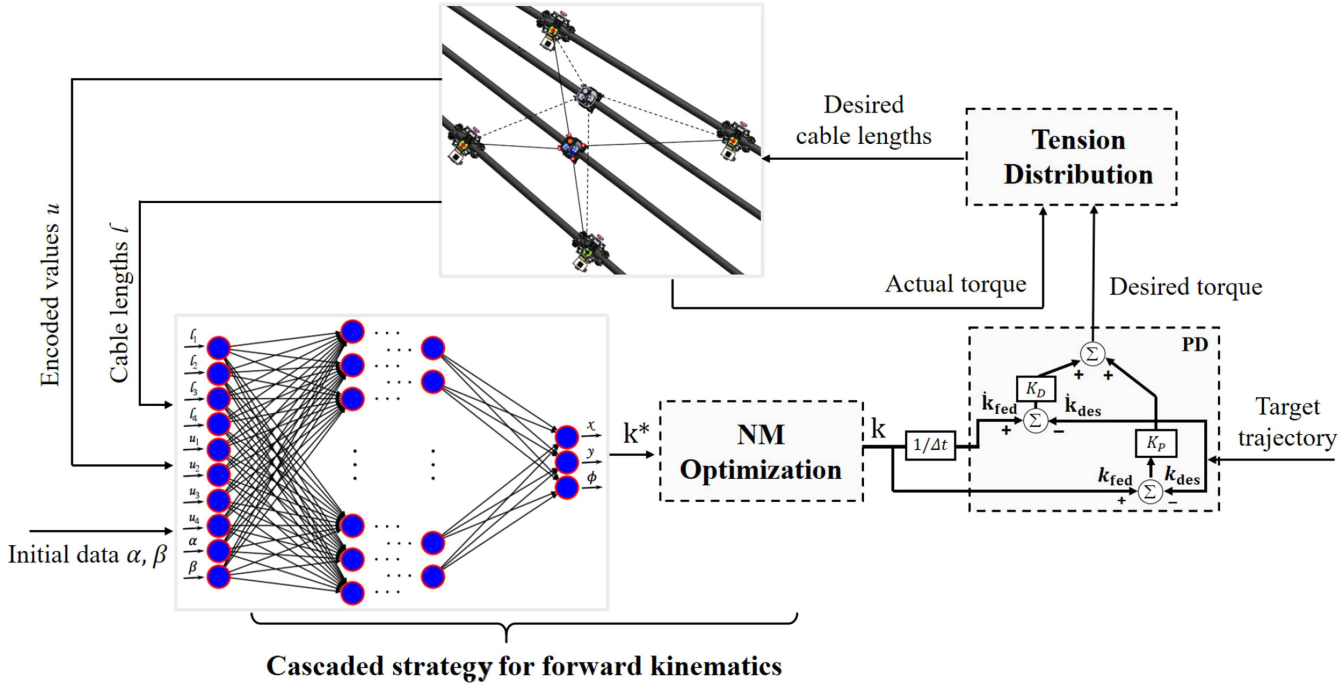

FIGURE 11 | The whole control system architecture of CCRobot-S, in which the forward kinematics solved by the cascaded strategy is used as feedback values in the control loop. [Color figure can be viewed at wileyonlinelibrary.com]

and crossing over the cables successively with a zigzag trajectory by the flying platform. The experimental result was recorded and shown in the Video S1. Overall, the system generally accomplished the cable surface inspection task. Consequently, the cascaded strategy for the CCRobot-S system has been verified to be feasible.

4.4 | Evaluation of the Convergence and Robustness

Convergence and robustness of the control algorithm are always vital for the CCRobot-S robotic system. Unfortunately, conventional approaches used to solve the forward kinematics, such as the Newton–Raphson method, cannot always guarantee convergence, which may lead to disastrous results. For example, the driving cable may tear apart abruptly, or the flying platform may drop suddenly due to ill-suited pose feedback. In this part, we tested the convergence of the cascaded strategy and the numerical method implemented by the Newton–Raphson algorithm. For the Newton–Raphson method, we chose the initial values randomly and then executed the algorithm repeatedly using 50 data points from the set of 36,000 validation data points. At least one execution of this algorithm cannot derive the right forward solution for the CCRobot-S robotic system, two adverse phenomena emerged during these tests stochastically:

- *Singularities:* If the initial value is not well-chosen, the Jacobian matrix $\mathbf{J}(\mathbf{k}, \mathbf{U})$ may become nearly singular during the iteration process (i.e., its determinant is very close to zero). This can cause the method to diverge or converge very slowly, and the flying platform cannot move further.
- *Multiple Solutions:* The Newton–Raphson method may converge to different roots depending on the initial value. As a result, the system may either fail to identify the correct root or inadvertently select an ill-suited one, and the flying platform may rush or crash to an undesired position.

For the proposed cascaded strategy, the initial values are provided by the deep learning-based neural network and then executed by the numerical algorithm with the same 50 data points from the set of 36,000 validation data values repeatedly. The test results show that every execution of this algorithm can derive the right forward solution for the CCRobot-S robotic system, and the phenomenon of the flying platform rushing or crashing to an undesired position does not happen.

Compared with the test results obtained using the Newton-Raphson algorithm and the proposed cascaded strategy, it is evident that the proposed cascaded strategy guarantees convergence and is more robust than the Newton-Raphson numerical method.

Finally, after the thorough evaluation and experimental analysis to assess the performance of our proposed cascaded strategy against existing methods, the comparisons of the methods, including the Newton-Raphson, Single NN, and Cascaded Strategy based on key performance metrics, are concluded in Table 4. As depicted in Table 4, the cascaded strategy demonstrated superior performance in terms of convergence rate, initial value source, solution uniqueness, and prediction accuracy.

5 | Conclusions and Further Works

This paper introduced and implemented a novel cascaded strategy with embodied artificial intelligence to solve the forward kinematics of the reconfigurable parallel-type cable-driven climbing robot squad. In this strategy, we adopted a deep learning-based neural network approach by learning from

TABLE 4 | Comparisons of the methods based on key performance metrics.

Method	Newton–Raphson	Single NN	Cascaded strategy
Initial value source	Random guess	Direct prediction	NN + kinematic constraints
Convergence rate	$\leq$ 98% (49/50 cases)	N/A	100% (50/50 cases)
Solution uniqueness	Multiple solutions	Unique solution (1:1 mapping)	Unique solution (1:1 mapping)
Prediction accuracy	equivalent to the cascaded strategy	1.43 $\pm$ 1.88 mm (X position error)	0.16$\pm$0.73 mm (X position error)

physical or simulated interactive experiences of CCRobot-S to derive an approximate solution, and then employed the numerical method to optimize the solution by treating the approximate solution as the initial values. During the implementation of the cascaded strategy, we developed the CCRobot-S kinematic model, dataset preparation, training configuration, solution inference, and numerical method optimization. Extensive evaluations and experiments have verified the feasibility and effectiveness of the proposed approach. The proposed cascaded strategy guarantees convergence and outperforms traditional methods in terms of prediction accuracy and robustness. Moving forward, we plan to deploy the CCRobot-S robotic system on actual cable-stayed bridges. By collecting dataset points through physical interactive experiences of CCRobot-S, we aim to expand the data capacity of the neural network. This will further enhance the system's performance. Based on this expanded dataset, we will conduct more evaluations to assess the effectiveness of the cascaded strategy.

Acknowledgments

Guangdong S&T Program (grant 2025B0909040003), Shenzhen Science and Technology Program (grant GJHZ20240218114202004), Guangdong Provincial Leading Talent Program (grant 2024TX08Z319), "Zhiguo" Action of Guangxi Science and Technology Program (grant ZG2503980003), and Longgang District Shenzhen's "Ten Action Plan" for Supporting Innovation Projects (grant LGKCSDPT2024003).

Data Availability Statement

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

Borgstrom, P. H., B. L. Jordan, G. S. Sukhatme, M. A. Batalin, and W. J. Kaiser. 2009. "Rapid Computation of Optimally Safe Tension Distributions for Parallel Cable-Driven Robots." *IEEE Transactions on Robotics* 25, no. 6: 1271–1281.

Bosscher, P., II R. L. Williams, L. S. Bryson, and D. Castro-Lacouture. 2007. "Cable-Suspended Robotic Contour Crafting System." *Automation in Construction* 17, no. 1: 45–55.

Bottema, O., and B. Roth. 1990. *Theoretical Kinematics*. Courier Corporation.

Chicco, D., M. J. Warrens, and G. Jurman. 2021. "The Coefficient of Determination R-Squared Is More Informative Than Smape, Mae, Mape, Mse and Rmse in Regression Analysis Evaluation." *Peerj computer science* 7: e623.

Cho, K. H., Y. H. Jin, H. M. Kim, and H. R. Choi. 2014. "Development of Novel Multifunctional Robotic Crawler for Inspection of Hanger Cables in Suspension Bridges." In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2673–2678. IEEE.

Cho, K. H., Y. H. Jin, H. M. Kim, H. Moon, J. C. Koo, and H. R. Choi. 2013a. "Caterpillar-Based Cable Climbing Robot for Inspection of Suspension Bridge Hanger Rope." In *2013 IEEE International Conference on Automation Science and Engineering (CASE)*, 1059–1062. IEEE.

Cho, K. H., Y. H. Jin, H. M. Kim, H. Moon, J. C. Koo, and H. R. Choi. 2016. "Multifunctional Robotic Crawler for Inspection of Suspension Bridge Hanger Cables: Mechanism Design and Performance Validation." *IEEE/ASME Transactions on Mechatronics* 22, no. 1: 236–246.

Cho, K. H., H. M. Kim, Y. H. Jin, et al. 2013b. "Inspection Robot for Hanger Cable of Suspension Bridge: Mechanism Design and Analysis." *IEEE/ASME Transactions on mechatronics* 18, no. 6: 1665–1674.

Denavit, J., and R. S. Hartenberg. 1955. A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices.

Ding, N., Z. Zheng, J. Song, Z. Sun, T. L. Lam, and H. Qian. 2020. "CCrobot-III: A Split-Type Wire-Driven Cable Climbing Robot for Cable-Stayed Bridge Inspection." In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 9308–9314. IEEE.

Ferraresi, C., M. Paoloni, S. Pastorelli, and F. Pescarmona. 2004. "A New 6-DOF Parallel Robotic Structure Actuated by Wires: The Wiro-6.3." *Journal of Robotic Systems* 21, no. 11: 581–595.

Gao, H., C. Chevallereau, and S. Caro. 2025. "Enhancing Safety in Collaborative Cable-Driven Parallel Robots: Contact Distinction and Management for Carrying Tasks." *IEEE Transactions on Automation Science and Engineering* 22: 18860–18874.

Geng, Z., and L. Haynes. 1991. "Neural Network Solution for the Forward Kinematics Problem of a Stewart Platform." In *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, 2650–2651. IEEE Computer Society.

Ghasemi, A., M. Eghtesad, and M. Farid. 2010. "Neural Network Solution for Forward Kinematics Problem of Cable Robots." *Journal of Intelligent & Robotic Systems* 60: 201–215.

Gouttefarde, M., and C. M. Gosselin. 2006. "Analysis of the Wrench-Closure Workspace of Planar Parallel Cable-Driven Mechanisms." *IEEE Transactions on Robotics* 22, no. 3: 434–445.

Hu, Y., F. Chen, B. Zhu, et al. 2025. "Forward Kinematics for Parallel Platforms Based on the Integration of Residual Networks With Self-Attention and Error Compensation." *IEEE Access* 13: 60200–60212.

Husty, M. L. 1996. "An Algorithm for Solving the Direct Kinematics of General Stewart–Gough Platforms." *Mechanism and Machine Theory* 31, no. 4: 365–379.

Jeong, J. W., S. H. Kim, and Y. K. Kwak. 1999. "Kinematics and Workspace Analysis of a Parallel Wire Mechanism for Measuring a Robot Pose." *Mechanism and Machine Theory* 34, no. 6: 825–841.

Kovac, M. 2016. "Learning From Nature How to Land Aerial Robots." *Science* 352, no. 6288: 895–896.

Li, J., X. Liu, S. Jiang, R. Li, and L. Ren. 2009. "Design of Continuous Climbing Pneumatic Cable Maintenance Robot." In *Mechatronics and Automation, 2009. ICMA 2009. International Conference on*, 4633–4637. IEEE.

Luo, J., S. Xie, Z. Gong, and T. Lu. 2007. "Development of Cable Maintenance Robot for Cable-Stayed Bridges." *Industrial Robot: An International Journal* 34, no. 4: 303–309.

Lytle, A. M., K. S. Saidi, R. V. Bostelman, W. C. Stone, and N. A. Scott. 2004. "Adapting a Teleoperated Device for Autonomous Control Using Three-Dimensional Positioning Sensors: Experiences With the Nist Robocrane." *Automation in construction* 13, no. 1: 101–118.

Mengqi, H., J. Li, F. Xu, and L. Hu. 2024. "Design and Implementation of a Novel Wheel-Based Cable Inspection robot." In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1310–1315. IEEE.

Merlet, J. -P. 2004. "Solving the Forward Kinematics of a Gough-Type Parallel Manipulator With Interval Analysis." *International Journal of robotics research* 23, no. 3: 221–235.

Miermeister, P., W. Kraus, and A. Pott. 2013. "Differential Kinematics for Calibration, System Investigation, and Force Based Forward Kinematics of Cable-Driven Parallel Robots." *Cable-Driven Parallel Robots* 12: 319–333.

Nguyen, S. T., K. T. La, and H. M. La. 2024. "Agile Robotic Inspection of Steel Structures: A Bicycle-Like Approach With Multisensor Integration." *Journal of Field Robotics* 41, no. 2: 396–419.

Nguyen, S. T., H. Nguyen, and S. T. Bui, et al. 2022. "An Agile Bicycle-Like Robot for Complex Steel Structure Inspection." In *2022 International Conference on Robotics and Automation (ICRA)*, 157–163. IEEE.

Oh, S. -R., and S. K. Agrawal. 2005. "Cable Suspended Planar Robots With Redundant Cables: Controllers With Positive Tensions." *IEEE Transactions on Robotics* 21, no. 3: 457–465.

Piva, G., D. Richiedei, and A. Trevisani. 2025. "Model Inversion for Trajectory Control of Reconfigurable Underactuated Cable-Driven Parallel Robots." *Nonlinear Dynamics* 113, no. 8: 8697–8712.

Pott, A. 2010. "An Algorithm for Real-Time Forward Kinematics of Cable-driven Parallel Robots." In *Advances in Robot Kinematics: Motion in Man and Machine: Motion in Man and Machine*, 529–538. Springer.

Pott, A., and T. Bruckmann. 2013. *Cable-Driven Parallel Robots*. Springer.

Sancak, C., M. Itik, and T. T. Nguyen. 2023. "Position Control of a Fully Constrained Planar Cable-Driven Parallel Robot With Unknown or Partially Known Dynamics." *IEEE/ASME Transactions on Mechatronics* 28: 1605–1615.

Santos, J. C., and M. Gouttefarde. 2021. "A Real-Time Capable Forward Kinematics Algorithm for Cable-Driven Parallel Robots Considering Pulley Kinematics." In *Advances in Robot Kinematics 2020*, 199–208. Springer.

Schmidt, V., B. Müller, and A. Pott. 2014. "Solving the Forward Kinematics of Cable-Driven Parallel Robots With Neural Networks and Interval Arithmetic." In *Computational Kinematics: Proceedings of the 6th International Workshop on Computational Kinematics (CK2013)*, 103–110. Springer.

Seng Yee, C., and K.-B. Lim. 1997. "Forward Kinematics Solution of Stewart Platform Using Neural Networks." *Neurocomputing* 16, no. 4: 333–349.

Sharkawy, A. -N., and S. S. Khairullah. 2023. "Forward and Inverse Kinematics Solution of a 3-DOF Articulated Robotic Manipulator Using Artificial Neural Network." *International Journal of Robotics and Control Systems* 3, no. 2: 330–353.

Tao, J., H. Zhou, and W. Fan. 2025. "A Hybrid Strategy for Forward Kinematics of the Stewart Platform Based on Dual Quaternion Neural Network and ARMA Time Series Prediction." In *Actuators*, 159. MDPI.

Tavakoli, M., G. Cabrita, R. Faria, L. Marques, and A. T. de Almeida. 2012. "Cooperative Multi-Agent Mapping of Three-Dimensional Structures for Pipeline Inspection Applications." *International Journal of Robotics Research* 31, no. 12: 1489–1503.

Tavakoli, M., L. Marques, et al. 2011. "3dclimber: Climbing and Manipulation Over 3d Structures." *Mechatronics* 21, no. 1: 48–62.

Trautwein, F., D. Dietrich, A. Pott, and A. Verl. 2025. "Strategy for Topological Reconfiguration of Cable-Driven Parallel Robots." *Journal of Mechanisms and Robotics* 17, no. 1: 010908.

Wang, R., J. Li, and Y. Li. 2025. "A Review on Design, Modeling and Control Technology of Cable-Driven Parallel Robots." *Robotics* 14, no. 9: 116.

Wang, Z., B. He, Y. Zhou, K. Liu, and C. Zhang. 2021. "Design and Implementation of a Cable Inspection Robot for Cable-Stayed Bridges." *Robotica* 39, no. 8: 1417–1433.

Wu, Q., B. Zi, B. Hu, and Y. Li. 2025. "Design, Kinematics and Stiffness Analysis of a Reconfigurable Cable-Driven Parallel Robot." *Chinese Journal of Mechanical Engineering* 38, no. 1: 98.

Xu, F., S. Dai, Q. Jiang, and X. Wang. 2021. "Developing a Climbing Robot for Repairing Cables of Cable-Stayed Bridges." *Automation in Construction* 129: 103807.

Xu, F., K. Ma, Y. Zhou, et al. 2025. "Design and Control Method of a Coupled Loading–Damping Mechanism of Cable-Detecting Robots for Large Bridges." *Journal of Field Robotics* 42, no. 5: 2005–2027.

Xu, F., X. Wang, and L. Wang. 2011. "Cable Inspection Robot for Cable-Stayed Bridges: Design, Analysis, and Application." *Journal of Field Robotics* 28, no. 3: 441–459.

Xu, J., B. -G. Kim, X. Feng, and K. -S. Park. 2024. "Online Motion Planning of Mobile Cable-Driven Parallel Robots for Autonomous Navigation in Uncertain Environments." *Complex & Intelligent Systems* 10, no. 1: 397–412.

Yang, G., S. H. Yeo, and C. B. Pham. 2004. "Kinematics and Singularity Analysis of a Planar Cable-Driven Parallel Manipulator." In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No. 04CH37566)*, 3835–3840. IEEE.

Yuan, J., X. Wu, Y. Kang, and A. Ben. 2010. "Research on Reconfigurable Robot Technology for Cable Maintenance of Cable-Stayed Bridges In-Service." In *2010 International Conference on Mechanic Automation and Control Engineering*, 1019–1022. IEEE.

Zhang, W., Z. Zheng, and X. Fu, et al. 2021. "CCRobot-IV-F: A Ducted-Fan-Driven Flying-Type Bridge-Stay-Cable Climbing Robot." In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 4184–4190. IEEE.

Zheng, Z., and N. Ding. 2019. "Design and Implementation of CCRobot-II: A Palm-Based Cable Climbing Robot for Cable-stayed Bridge Inspection." In *2019 International Conference on Robotics and Automation (ICRA)*, 9747–9753. IEEE.

Zheng, Z., N. Ding, and H. Chen, et al. 2022. "CCRobot-V: A Silkworm-Like Cooperative Cable-climbing Robotic System for Cable Inspection And Maintenance." In *2022 International Conference on Robotics and Automation (ICRA)*, 164–170. IEEE.

Zheng, Z., S. Hu, and N. Ding. 2018a. "A Biologically Inspired Cable Climbing Robot: CCRobot-Design And Implementation." In *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2354–2359. IEEE.

Zheng, Z., C. Wang, X. Hu, et al. 2025. "Developing a Climbing Robot for Stay Cable Maintenance With Security and Rescue Mechanisms." *Journal of Field Robotics* 42: 2532–2548.

Zheng, Z., X. Yuan, X. Yu, and N. Ding. 2018b. " Mechanical Design of a Climbing Robot for Inspection on a Cable-Stayed Bridge." In The World Congress on Intelligent Control and Automation (WCICA), 1680–1684. IEEE.

Zheng, Z., W. Zhang, X. Fu, et al. 2021. "CCRobot-IV: An Obstacle-Free Split-Type Quad-Ducted Propeller-Driven Bridge Stay Cable-Climbing Robot." *IEEE Robotics and Automation Letters* 7, no. 4: 11751–11758.

Supporting Information

Additional supporting information can be found online in the Supporting Information section.
Supplementary Information