frontiers | Frontiers in Robotics and AI

TYPE Original Research
PUBLISHED 12 September 2025
DOI 10.3389/frobt.2025.1627688

OPEN ACCESS

EDITED BY
Hani J. Marcus,
University College London, United Kingdom

REVIEWED BY
Dler Salih Hasan,
Salahaddin University, Iraq
Azamat Yeshmukhametov,
Nazarbayev University, Kazakhstan

*CORRESPONDENCE
Dirk Oberschmidt,
dirk.oberschmidt@tu-berlin.de

RECEIVED 13 May 2025
ACCEPTED 21 July 2025
PUBLISHED 12 September 2025

CITATION
Fritsch S and Oberschmidt D (2025) A projection-based inverse kinematic model for extensible continuum robots and hyper-redundant robots with an elbow joint. *Front. Robot. AI* 12:1627688. doi: 10.3389/frobt.2025.1627688

A projection-based inverse kinematic model for extensible continuum robots and hyper-redundant robots with an elbow joint

Sven Fritsch and Dirk Oberschmidt*

Department of Mechanical Engineering and Transport Systems, Technical University, Berlin, Germany

Inverse kinematics is a core problem in robotics, involving the use of kinematic equations to calculate the joint configurations required to achieve a target pose. This study introduces a novel inverse kinematic model (IKM) for extensible (i.e., length-adjustable) continuum robots (CRs) and hyper-redundant robots (HRRs) featuring an elbow joint. This IKM numerically solves a set of equations representing geometric constraints (abbreviated as NSGC). NSGC can handle target poses $\mathbf{X}_t = [x_t, y_t, z_t, \psi_t]$ in 3D space, which are projected onto a 2D plane and solved numerically. NSGC is capable of real-time operation and accounts for elbow joint limits. Extensive simulations and empirical tests confirm the reliability, performance, and practical applicability of NSGC.

KEYWORDS

continuum robots, hyper-redundant robots, inverse kinematics, optimization algorithm, robot control

1 Introduction

In recent years, the design and modeling of continuum robots (CRs) and hyper-redundant robots (HRRs) have garnered significant attention. CRs comprise multiple backbones routed in parallel and attached to a common end disk. Typically, CRs also incorporate spacer disks to maintain the backbone configuration and prevent buckling during actuation. On the other hand, HRRs consist of multiple rigid elements that can rotate relative to one another, enabling highly flexible and dexterous movements. Bending of the HRR is achieved through tendons routed through each segment. These tendons are connected to actuators that apply tension, inducing the desired curvature. Additional tendons control instruments located at the HRR's end-effector. The inherent miniaturization potential of CRs and HRRs makes them well-suited for applications in minimally invasive surgery and endoscopy, where precision and maneuverability are essential.

1.1 A literature review of IKMs for CRs and HRRs

The most commonly used approach to modeling CRs and HRRs is the constant curvature model. CRs and HRRs with multiple segments can be modeled using the piecewise constant curvature model (Burgner-Kahrs et al., 2015). Using the Jacobian

matrix to establish differential kinematics is the standard approach for CR/HRR inverse kinematic models (IKMs) (Whitney, 1972) and was also employed by Wolf et al. (2005), Garg et al. (2014), Guardiani et al. (2022), Lee et al. (2014), Li et al. (2017), Rone and Ben-Tzvi (2014), Simaan et al. (2009), Jones and Walker (2006), and Yeshmukhametov et al. (2019).

The generalized coordinates $\mathbf{q}$ are updated by the multiplication of the pseudo-inverse of the Jacobian (Chirikjian and Burdick, 1994) by the difference in parameterized end-effector coordinates (Klein and Huang, 1983). Alternative approaches comprise the augmented (Seraji, 1989) and extended (John, 1983) Jacobian.

Bajo et al. (2012) applied a Jacobian-based IKM to their IREP platform, which consists of two independent, two-segment CRs. They computed the Jacobian matrices for each segment individually.

Giorelli et al. (2013) proposed an alternative approach tailored to non-constant curvature CRs, where the static model is governed by nonlinear differential equations, making exact solutions challenging. As analytical solutions are unavailable, the elements of the Jacobian matrix cannot be computed directly. Instead, a feed-forward neural network is trained to learn the IKM for a non-constant curvature cable-driven manipulator. Similar neural network-based methods were presented by Lai et al. (2019) and Thuruthel et al. (2016).

Wei et al. (2023) presented an IKM for a two-segment HRR under the constant curvature assumption. They simplified the complex nonlinear system by reducing it to a single nonlinear equation, resulting in faster solutions.

Xu et al. (2018) adopted an iterative approach to compute the IKM utilizing a least squares method to determine the joint angles from specified cable lengths. Their model incorporated multilevel mappings that describe the relationships between motors and cables, cables and joints, and joints and the robot's end-effector.

Almanzor et al. (2023) introduced a method that combines a local inverse kinematics formulation in image space with a deep convolutional neural network trained on synthetic data. A 2D hand-drawn image of the desired shape is used as the input, and the network predicts motor commands to drive the robot toward that shape. Closed-loop control is achieved using visual feedback from a webcam.

Wild et al. (2023) developed a novel piecewise dual quaternion algorithm to model multi-section CRs, aiming to improve the traditional pseudo-inverse Jacobian method. The piecewise dual quaternion approach offers reduced computational complexity, faster convergence, and smoother backbone configurations, especially as the number of robot sections increases.

Kim and Wi (2025) introduced a tendon-driven discrete CR using ball–socket joints. Between one and three robot units connected in series were tested by applying proximal tendon tension, while distal tension was gradually increased to induce bending. The resulting bending curves were interpolated using third-to-fifth-order polynomials.

Li et al. (2025) proposed a novel multilevel motion control method for HRRs based on joint angle–cable force cooperative optimization. It comprises a finite-time optimization approach using macro–micro decomposition to solve inverse kinematics. The high-degree-of-freedom (DoF) robot is divided into smaller manipulators and solved using a dual neural network model.

Using the space vector method, the manipulator's kinematic model is developed to dynamically determine its endpoint position (Wang et al., 2024). The workspace is then generated through the Monte Carlo method. The original search approach is enhanced by introducing an angle-decoupling mechanism between adjacent links to calculate each joint's rotation angles.

Zhan et al. (2024) introduced a general approach for solving the real-time optimized IKM of redundant robots, while strictly enforcing hard limits in both joint and Cartesian spaces that cannot be violated. Instead of quadratic programming, the method employs constrained linear programming to address the IKM problem. Hard constraints—including joint limits, velocity, and acceleration bounds—are explicitly managed as inequality constraints.

Sheng et al. (2024) presented an IKM method that combines the end-following approach with a segmented, Jacobian-based iterative solver for CRs with redundant DoF and a moving base. The end-following method is first used to generate a smooth, constraint-compliant initial joint configuration by having joints sequentially follow the target points along a planned path. This initial guess is then refined using Jacobian-based inverse kinematics applied to robot segments, thus improving accuracy while reducing the computation time.

The following subsection provides a brief overview of the Jacobian-based IKM.

```
Initialize tol ← 0.01 and q ← (q)_0
while ‖X_t − X(q)‖ > tol do
  Compute J(q)
  J(q)^+ ← (J(q))^+
  ΔX ← X_t − X(q)
  q ← q + k · J(q)^+ · ΔX
end while
```

Algorithm 1. Jacobian-based IKM.

1.2 Jacobian-based IKM

The state-of-the-art IKMs for CRs and HRRs are based on the Jacobian matrix $\mathbf{J}$. $\mathbf{q}$ denotes the generalized (joint) coordinates, and $\mathbf{X}(\mathbf{q})$ denotes the end-effector pose corresponding to $\mathbf{q}$. $(\mathbf{q})_0$ represents the initial guess of $\mathbf{q}$. $\mathbf{X}_t$ denotes the target pose. $\mathbf{J}^+$ represents the Moore–Penrose inverse of the Jacobian as shown in Algorithm 1.

It should be noted that, in general, calculating the orientation difference requires careful handling beyond the simple subtraction of the target and achieved end-effector orientations.

The Jacobian matrix, as defined in Equation 1, is commonly used in the kinematics and dynamics of robotic systems. It relates variations in joint space to corresponding changes in configuration space.

$$\mathbf{J}(\mathbf{q}) = \begin{bmatrix} \frac{\partial X_1}{\partial q_1} & \cdots & \frac{\partial X_1}{\partial q_{n_j}} \\ \vdots & \ddots & \vdots \\ \frac{\partial X_m}{\partial q_1} & \cdots & \frac{\partial X_m}{\partial q_{n_j}} \end{bmatrix}. \tag{1}$$

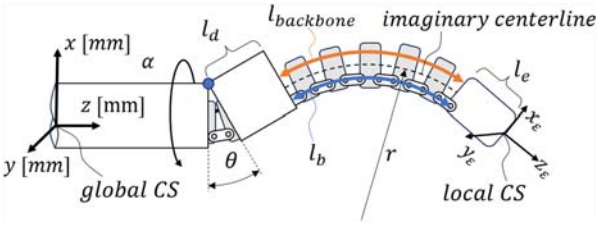

FIGURE 1
General CR/HRR structure suitable for NSGC: a proximal part that can be rotated about its main axis about angle α is followed by an elbow joint with deflection angle θ, a distal part with length l_d, a length-adjustable bending part characterized by length l_b and Δl_b, and the end-effector with length l_e; CS stands for coordinate system.

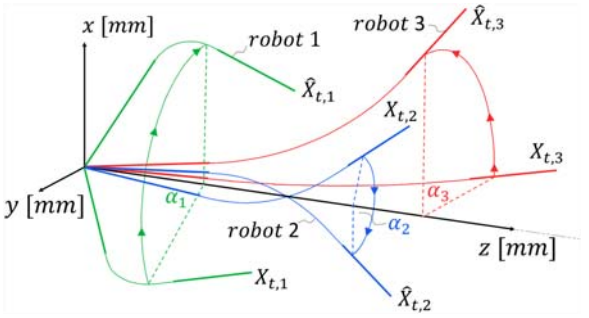

FIGURE 2
Projection of target poses $\mathbf{X}_t$ defined in 3D space projected onto the global x–z-plane.

The forward kinematic model of the robot describes the transformation matrices from the inertial frame $\mathcal{I}$ to the end-effector frame $\mathcal{E}$, as provided in Equation 2.

$$\mathbf{T}_{\mathcal{I}\mathcal{E}}(\mathbf{q}) = \mathbf{T}_{\mathcal{I}0} \cdot \left(\prod_{k=1}^{m} \mathbf{T}_{k-1,k}(\mathbf{q}_k) \right) \cdot \mathbf{T}_{m\mathcal{E}}. \tag{2}$$

The Moore–Penrose inverse is computed using Equation 3.

$$\mathbf{J}^{+} = \left(\mathbf{J}^T\mathbf{J}\right)^{-1}\mathbf{J}^T. \tag{3}$$

Due to the iterative nature of the algorithm, the error converges against a predefined threshold that is considered acceptable by the user. The gain k increases the step size per update and influences the convergence speed.

This study introduces a novel IKM that departs from the traditional Jacobian-based approach. Instead, it relies on the numerical solution of a set of equations representing geometric constraints, referred to hereafter as NSGC. These equations are formulated using the unknown parameters of the arc representing the curvature of the bent CR/HRR. Solving this system enables the calculation of the robot's length and bending parameters, which are subsequently converted into generalized coordinates $\mathbf{q}$.

2 NSGC

Given a target pose $\mathbf{X}_t = [x_t, y_t, z_t, \psi_t]$ in 3D space, the objective of the NSGC algorithm is to compute the generalized coordinates $\mathbf{q}$ defined by $\mathbf{q} = [l_b, \Delta l_b, \alpha, \theta]$. For CRs and HRRs, l_b is defined as the length of a tendon/backbone, while Δl_b is the length difference of the antagonistically actuated tendon/backbone. As the proposed HRR only features one backbone, l_b defines the sum of the lengths of the robotic elements and the connectors between the distal end of the distal part and the proximal end of the end-effector, while $\Delta l_b = l_b - l_{backbone}$ defines the length difference between the backbone and l_b, as illustrated in Figure 1. The proposed NSGC method can be applied to CRs and HRRs with an elbow joint and adjustable length capabilities.

2.1 Projection from 3D space onto the 2D plane

The target pose $\mathbf{X}_t = [x_t, y_t, z_t, \psi_t]$ is given in 3D space. x_t, y_t and z_t are the position coordinates defined in the global CS, while ψ_t describes the end-effector orientation about the y_ε-axis expressed in the local end-effector CS. The target pose must be projected onto a 2D plane, i.e., the global x–z-plane with $\text{proj}(\mathbf{X}_t) = \widehat{\mathbf{X}}_t = [\hat{x}_t, z_t, \psi_t]$. The projection $\hat{x}_t$ is computed using Equation 4 and can be either positive or negative.

$$\hat{x}_t = \pm\sqrt{x_t^2 + y_t^2}. \tag{4}$$

As shown in Figure 2, the angle α, about which $\mathbf{X}_t$ is rotated, is defined by Equation 5.

$$\alpha = \text{atan2}\left(y_t, x_t\right). \tag{5}$$

The projection does not affect ψ_t because ψ_t is defined as the rotation about the y_ε-axis in the local end-effector frame. In other words, ψ_t represents the same local orientation in both the original and projected poses. Similarly, the z-axis also remains unchanged by the projection, i.e., the z-value of the original target pose $\mathbf{X}_t$ in 3D is identical to the z-value of the projected target pose $\widehat{\mathbf{X}}_t$ in 2D.

The projection of different robot configurations onto the global x–z-plane is displayed in Figure 2. $\mathbf{X}_t$ denotes the original target pose, as defined by the user, and $\widehat{\mathbf{X}}_t$ denotes the target pose projected onto the global x–z-plane. The angle of rotation of the pose is denoted by α. For example, the $\mathbf{X}_{t,1}$ pose is rotated about $\alpha_1 = -60°$; the $\mathbf{X}_{t,2}$ pose is rotated about $\alpha_2 = -155°$; and the $\mathbf{X}_{t,3}$ pose is rotated about $\alpha_3 = 90°$. *A priori*, it is unknown whether the projection $\hat{x}_t$ should be positive (e.g., $\mathbf{X}_{t,1}$ or $\mathbf{X}_{t,3}$) or negative ($\mathbf{X}_{t,2}$). A detailed treatment of the projection ambiguity is provided in Section 2.1.

2.2 Robot configurations in 2D

Given a CR or HRR with an elbow joint and a length-adjustable bending part, the robot can achieve target poses $\widehat{\mathbf{X}}_t = [\hat{x}_t, z_t, \psi_t]$ in a 2D plane, where $\hat{x}_t$ and z_t denote the end-effector position and ψ_t denotes the end-effector orientation. The elbow joint is

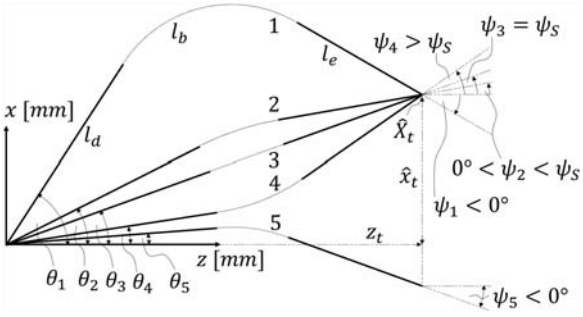

FIGURE 3
Different cases: robot 1: downward bending with $\psi_1 < 0°$ and $\hat{x}_t > 0$; robot 2: downward bending with $0° < \psi_2 < \psi_S$ and $\hat{x}_t > 0$; robot 3: straight configuration with $\psi_3 = \psi_S$ and $\hat{x}_t > 0$; robot 4: upward bending with $\psi_4 > \psi_S$ and $\hat{x}_t > 0$; robot 5: downward bending with $\psi_5 < 0°$ and $\hat{x}_t < 0$.

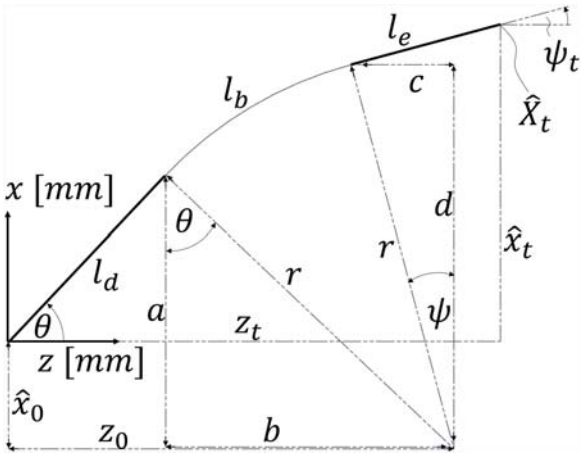

FIGURE 4
Visualizing the arc parameters $\mathbf{j} = [\hat{x}_0, z_0, \theta, r]$ for downward bending; i.e., $\psi_t < \psi_S$.

capable of deflection angles $\theta_{\min} \le \theta \le \theta_{\max}$, where $\theta_{\min}$ and $\theta_{\max}$ are the (mechanical) limits. Figure 3 illustrates when ψ is positive or negative.

Generally, there are three possible robot configurations: downward bending, upward bending, and straight configuration. The distinction between the different cases is relevant as the equations representing the geometric constraints are slightly different for each case. ψ_t describes the end-effector orientation of the projected target pose $\widehat{\mathbf{X}}_t = [\hat{x}_t, z_t, \psi_t]$ and must be compared with ψ_S, which is computed using Equation 6.

$$\psi_S = \operatorname{atan2}(\hat{x}_t, z_t). \tag{6}$$

In the straight configuration, the distal part l_d, the bending part l_b, and the end-effector l_e are aligned as displayed by robot 3 in Figure 3. It should be noted that in this case, the bending part is straight and not bent, and $\theta = \psi_S = \psi_t$. The case distinction is carried out according to Equation 7.

$$\psi_t \begin{cases} < \psi_S, & \text{downward bending} \\ = \psi_S, & \text{straight configuration} \\ > \psi_S, & \text{upward bending} \end{cases}. \tag{7}$$

2.3 Set of equations

For both downward and upward bending, the set of equations representing the geometric constraints is provided in Equations 8–11.

$$r^2 = a^2 + b^2, \tag{8}$$

$$\tan(\theta) = \frac{b}{a}, \tag{9}$$

$$r^2 = c^2 + d^2, \tag{10}$$

$$\tan(\psi) = \frac{c}{d}. \tag{11}$$

In Sections 2.4, 2.5, the placeholders a, b, c, and d are substituted by the arc parameters $\mathbf{j} = [\hat{x}_0, z_0, \theta, r]$. $\hat{x}_0$ and z_0 denote the center coordinates of the projected arc in the global x–z-plane, θ describes the deflection angle of the elbow joint, and r is the radius of the arc. After substitution of the placeholders, the set of equations will only depend on known hardware design parameters (i.e., l_b and l_e) and the arc parameters $\mathbf{j}$. This allows for the numerical solution of the arc parameters $\mathbf{j}$, which in turn can be used to compute the generalized coordinates $\mathbf{q}$.

2.4 Downward bending

For $\psi_t < \psi_S$, the robot performs downward bending. Notably, in downward bending, ψ_t may be either positive ($0° < \psi_t < \psi_S$) or negative ($\psi_t < 0°$). In Figure 4, the arc parameters $\mathbf{j} = [\hat{x}_0, z_0, \theta, r]$ are shown for downward bending.

Equations 8–11 can be trivially derived from the Pythagorean and angular relationships, as displayed in Figure 4. For downward bending, the placeholders a, b, c, and d, defined in the set of equations representing the geometric constraints (Equations 8–11), must be substituted as follows:

$$a = -\hat{x}_0 + \sin(\theta) \cdot l_d, \tag{12}$$

$$b = z_0 - \cos(\theta) \cdot l_d, \tag{13}$$

$$c = z_0 - z_t + \cos(\psi_t) \cdot l_e, \tag{14}$$

$$d = -\hat{x}_0 + \hat{x}_t - \sin(\psi_t) \cdot l_e. \tag{15}$$

In the substituted set of equations, only the arc parameters $\mathbf{j} = [\hat{x}_0, z_0, \theta, r]$ are unknown.

2.5 Upward bending

It is evident that the same straightforward geometric relationships described in the set of equations also apply to upward

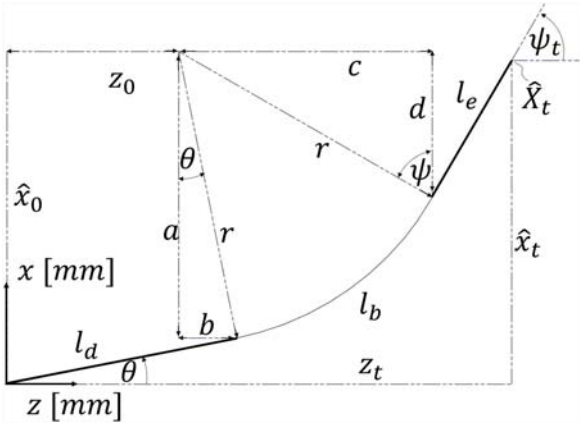

FIGURE 5
Visualizing the arc parameters $\mathbf{j} = [\hat{x}_0, z_0, \theta, r]$ for upward bending; i.e., $\psi_t > \psi_S$.

bending. For $\psi_t > \psi_S$, the robot performs upward bending as shown in Figure 5. For upward bending, the placeholder variables defined in the set of equations representing the geometric constraints (Equations 8–11) must be substituted as follows:

$$a = \hat{x}_0 - \sin(\theta) \cdot l_d, \tag{16}$$

$$b = \cos(\theta) \cdot l_d - z_0, \tag{17}$$

$$c = z_t - \cos(\psi_t) \cdot l_e - z_0, \tag{18}$$

$$d = \hat{x}_0 - \hat{x}_t + \sin(\psi_t) \cdot l_e. \tag{19}$$

The steps so far can be summarized as follows:

The given target pose $\mathbf{X}_t$ in 3D is projected to 2D, i.e., $\widehat{\mathbf{X}}_t$. Then, it needs to be determined in which configuration the robot can reach $\widehat{\mathbf{X}}_t$, i.e., downward or upward bending. Depending on the configuration, the placeholders in the set of equations are substituted accordingly. In the substituted set of equations, only the arc parameters $\mathbf{j} = [\hat{x}_0, z_0, \theta, r]$ are unknown.

2.6 (Almost) Straight configuration

As ψ_t approaches ψ_S, the bending becomes less pronounced, and the radius r increases significantly, as described by Equation 20.

$$(\psi_t \to \psi_S) \Rightarrow (r \to \infty). \tag{20}$$

Finding a numerical solution where the variables approach infinity is unfeasible. Thus, to avoid divergence issues, in cases where the robot is straight or almost straight (i.e., $|\psi_t - \psi_S| < 1°$), the robot is regarded as being in a straight configuration. For $\psi_t = \psi_S = \theta$, the robot performs no bending, as visualized in Figure 6; hence, there is no need to find arc parameters $\mathbf{j}$ using Powell's hybrid method.

Instead, the generalized coordinate l_b is computed analytically according to Equation 21.

$$l_b = \sqrt{(\hat{x}_t - \sin(\psi_t) \cdot (l_e + l_d))^2 + (z_t - \cos(\psi_t) \cdot (l_e + l_d))^2}. \tag{21}$$

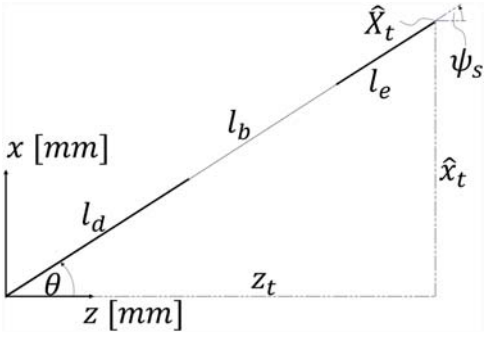

FIGURE 6
Visualizing the straight configuration for $\theta = \psi_t = \psi_S$.

Thus, in case of a straight configuration, the generalized coordinates are $\mathbf{q} = [l_b, \Delta l_b = 0\,\text{mm}, \alpha, \theta = \psi_t]$.

Reverting to a linear equation for the (almost) straight configuration ensures an extremely low runtime while maintaining acceptable accuracy. For the upward and downward bending scenarios, the set of equations is solved using Powell's hybrid method, as described in the next subsection.

2.7 NSGC using Powell's hybrid method

Powell's hybrid method (Powell, 1970), also known as Powell's dog-leg method, is an iterative optimization algorithm used to solve systems of nonlinear equations. It is designed to find the roots of systems of N nonlinear functions in N variables. In this article, it is used to numerically solve the set of nonlinear equations defined in Equations 8–11, where the placeholder variables a, b, c, and d are defined in Equations 12–15 for downward bending and in Equations 16–19 for upward bending. The objective function $\mathbf{f}_{\widehat{\mathbf{X}}_t}$ is stated in Equation 22.

$$\mathbf{f}_{\widehat{\mathbf{X}}_t}(\mathbf{j}) = \mathbf{f}_{\widehat{\mathbf{X}}_t}(\hat{x}_0, z_0, \theta, r) = \mathbf{0}, \tag{22}$$

where $\mathbf{f}_{\widehat{\mathbf{X}}_t}$ represents the substituted set of equations and $\hat{x}_0, z_0, \theta, r$ represent the unknown arc parameters $\mathbf{j}$. Now, Powell's hybrid method is briefly outlined and applied to the equations representing the geometric constraints.

Initialization:

- Set the initial guess $\mathbf{j}_0 = [(\hat{x}_0)_0, (z_0)_0, (\theta)_0, (r)_0]$.
- Define the trust region radius Δ.
- Initialize the scaling matrices $\mathbf{D}_1$ and $\mathbf{D}_x$ if necessary, where $\mathbf{D}_1$ scales the equations and $\mathbf{D}_x$ scales the variables (Chen and Stadtherr, 1981).

Iteration:

1) Evaluate the function $\mathbf{f}_{\widehat{\mathbf{X}}_t}(\mathbf{j}_k)$ and the Jacobian matrix $\mathbf{J}(\mathbf{j}_k)$ or its approximation $\mathbf{B}$ (Chen and Stadtherr, 1981).
2) Calculate the Gauss–Newton step using the Jacobian as defined in Equation 23

$$\mathbf{p}_N = -\mathbf{J}(\mathbf{j}_k)^{-1} \mathbf{f}_{\widehat{\mathbf{X}}_t}(\mathbf{j}_k), \tag{23}$$

or its approximation as defined in Equation 24

$$\mathbf{p}_N = -\mathbf{B}^{-1}\mathbf{f}_{\widehat{\mathbf{X}}_t}(\mathbf{j}_k). \tag{24}$$

If $\mathbf{B}$ is used, it is often an approximation of the Jacobian, such as from a quasi-Newton method (Chen and Stadtherr, 1981).

3) Check whether the Gauss–Newton step is within the trust region as defined in Equation 25

$$\|\mathbf{p}_N\| \leq \Delta. \tag{25}$$

Then, set $\mathbf{p} = \mathbf{p}_N$ and proceed to update the solution.

4) Calculate the Cauchy point: If the Gauss–Newton step is outside the trust region ($\|\mathbf{p}_N\| > \Delta$), calculate the steepest descent direction (Cauchy point) in scaled coordinates in scaled coordinates as defined in Equation 26:

$$\mathbf{g} = -\mathbf{D}_1\mathbf{B}^T\mathbf{D}_1^{-1}\mathbf{f}_{\widehat{\mathbf{X}}_t}(\mathbf{j}_k). \tag{26}$$

The Cauchy point is then found by minimizing the quadratic function along the steepest descent direction as defined in Equation 27:

$$Q(\mu\mathbf{g}) = \frac{1}{2}\left\|\mathbf{D}_1\left(\mathbf{f}_{\widehat{\mathbf{X}}_t}(\mathbf{j}_k) + \mu\mathbf{B}\mathbf{g}\right)\right\|^2. \tag{27}$$

The multiplier μ is chosen to minimize $Q(\mu\mathbf{g})$, which is a quadratic function of μ as stated in Equation 28::

$$\mu = -\frac{\mathbf{f}_{\widehat{\mathbf{X}}_t}(\mathbf{j}_k)^T\mathbf{B}\mathbf{g}}{\mathbf{g}^T\mathbf{B}^T\mathbf{B}\mathbf{g}}. \tag{28}$$

The Cauchy point is then given by $\mathbf{p}_S = \mu\mathbf{g}$.

5) Determine the dog-leg step:
   - If $\|\mathbf{p}_S\| \leq \Delta$, set $\mathbf{p} = \mathbf{p}_S$.
   - If $\|\mathbf{p}_S\| > \Delta$, find the intersection of the dog-leg path with the trust region boundary as defined in Equation 29:

$$\mathbf{p} = \alpha\mathbf{p}_N + (1-\alpha)\mathbf{p}_S, \tag{29}$$

where α is found such that $\|\mathbf{p}\| = \Delta$.

6) Update the solution: $\mathbf{j}_{k+1} = \mathbf{j}_k + \mathbf{p}$.
7) Check for convergence: Evaluate the function at the new point and check whether the tolerance threshold or the maximum number of steps criterion is met. If either condition is satisfied, stop the iteration. Otherwise, repeat the process.

This method ensures global convergence by combining the Gauss–Newton direction with the steepest descent direction within a trust region, making it robust for the set of equations representing the geometric constraints.

2.8 Initial guesses

To ensure that NSGC finds a solution, it is necessary to provide multiple initial guesses $\mathbf{x}_0$ for the algorithm to iterate through. In practice, it is expedient to establish different initial guesses for slight bending (e.g., $|\psi_t - \psi_S| \leq 15°$) and strong bending (e.g., $|\psi_t - \psi_S| > 15°$). This approach allows the NSGC algorithm to avoid iterating through all the initial guesses, bypassing those that are potentially ill-suited for the current bending. For very slight bending (e.g., $|\psi_t - \psi_S| < 1°$), the robot is considered to be in a straight configuration, as detailed in Section 2.6.

2.9 Projection with elbow joint constraints

If the projection of the target pose is always performed positively, such that $\text{proj}(\mathbf{X}_t) = \widehat{\mathbf{X}}_t = [\hat{x}_t, z_t, \psi_t]$ with $\hat{x}_t = \sqrt{x_t^2 + y_t^2}$ according to Equation 4, certain poses become unreachable due to mechanical joint limits. This issue is illustrated in Figure 2 for robot 2; if the projection $\widehat{\mathbf{X}}t$ were carried out positively (i.e., with a rotation of $\alpha_2 = 25°$) and assuming constant curvature of the bending section, the elbow joint would need to bend downward beyond its lower bound $\theta < \theta_{\text{min}}$, thus violating the joint constraint. Consequently, the projection must instead be performed negatively with $\hat{x}_t = -\sqrt{x_t^2 + y_t^2}$ and $\alpha_2 = -155°$.

As mentioned in Section 2.1, it is not known *a priori* whether a given target pose $\mathbf{X}_t$ should be projected positively or negatively. The limit of the feasible end-effector orientation $\bar{\psi}_t$ depends on both the target position $(\hat{x}_t, z_t)$ and the elbow joint constraint. For the case $\theta_{\text{min}} = 0°$, the boundary of the reachable orientation is defined using the following equations (Equations 30–32):

$$r = \hat{x}_0, \tag{30}$$

$$r^2 = (r - \hat{x}_t + \sin(\bar{\psi})\cdot l_w)^2 + (z_t - l_d - \cos(\bar{\psi})\cdot l_w)^2, \tag{31}$$

$$\bar{\psi} = \text{atan2}\quad \begin{matrix}((z_t - l_d - \cos(\bar{\psi})\cdot l_w),\\ (r - \hat{x}_t + \sin(\bar{\psi})\cdot l_w))\end{matrix}. \tag{32}$$

Here, the variables $\hat{x}_0$, r, and $\bar{\psi}$ are unknowns. After solving this system, $\bar{\psi}$ is compared to the target end-effector orientation ψ_t to determine whether a positive or negative projection is appropriate. If $\psi_t \leq \bar{\psi}$, a positive projection is feasible; otherwise, the projection must be carried out negatively. In the latter case, the robot must rotate about its main axis and approach the target from the $-\hat{x}$ direction, thereby staying within the joint constraint $\theta \geq \theta_{\text{min}} = 0°$ and achieving $\widehat{\mathbf{X}}_t = [\hat{x}_t, z_t, \psi_t]$ with $\psi_t > \bar{\psi}$, as illustrated in Figure 7a).

This rotation about the main axis significantly enlarges the workspace, as shown in Figure 7b. When a target pose is only reachable via this axial rotation, the projected pose must be transformed as $\widehat{\mathbf{X}}_t = [\hat{x}_t, z_t, \psi_t] \rightarrow \tilde{\mathbf{X}}_t = [-\hat{x}_t, z_t, -\psi_t]$. This corresponds to the rotated configuration. In that case, α [Equation 5] has to be increased by 180°. The pose $\tilde{\mathbf{X}}_t$ is then used as the target for inverse kinematics and solved using Powell's hybrid method.

2.10 False positives and convergence conditions

NSGC can provide false positives that are mathematically correct but do not represent the physical robot. The predominant cases of false positives are shown in Figure 8.

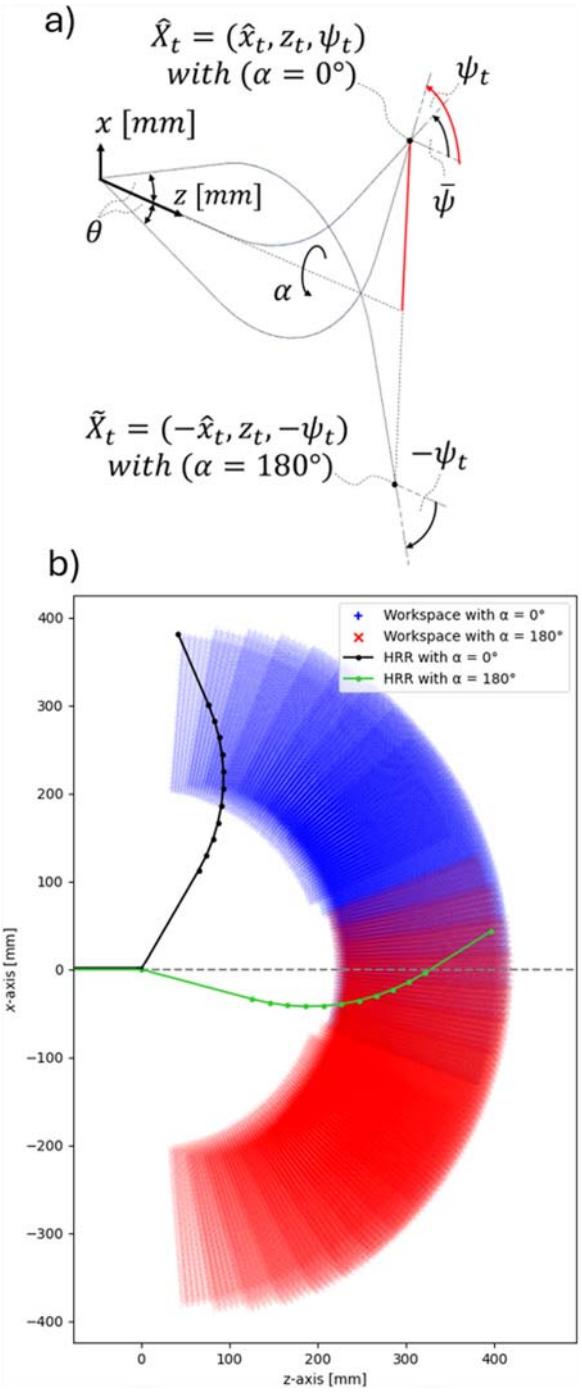

FIGURE 7
(a) Utilizing α to reach target poses with $\psi_t > \bar{\psi}$. **(b)** Workspace enlargement by rotating the robot about its main axis (e.g., $\alpha = 180°$).

It must be ensured that the algorithm does not accept a false positive as a numerical solution. Thus, for each solution is checked against the following three convergence conditions:

I. $0° \overset{!}{\leq} \theta \overset{!}{\leq} 60°$.
II. If $\psi_t < \psi_S$ (i.e., downward bending), then $\theta \overset{!}{>} \psi_S$ (Figure 8a).

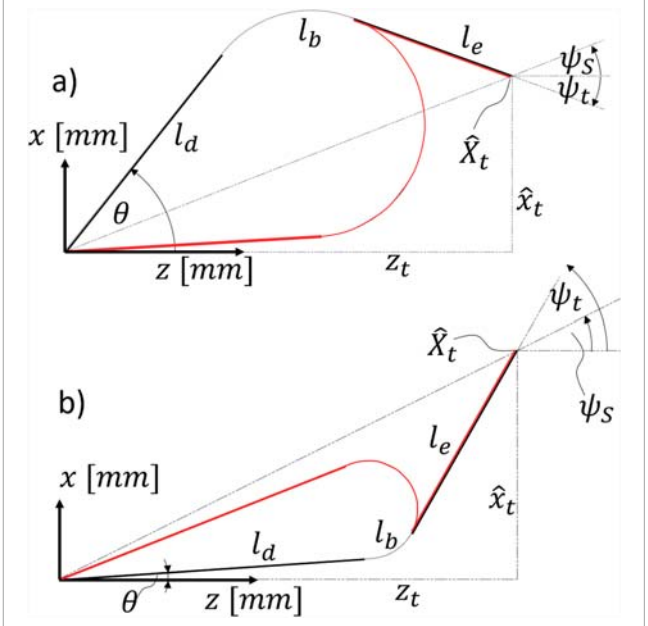

FIGURE 8
Black robots are correct, and the red robots are false positives. **(a)** Downward bending: instead of having $\theta > \psi_S$ and then a downward bending, this NSGC solution has $\theta < \psi_S$ and then an upward bending. **(b)** Upward bending: the numerical solution provides a downward bending robot such that the bending part l_b and end-effector l_e are tangents. However, this solution does not represent the physical nature of the robot and needs to be filtered out.

III. If $\psi_t \overset{!}{>} \psi_S$ (i.e., upward bending), then $\sin(\theta) \cdot l_d \overset{!}{<} \hat{x}_T - \sin(\psi) \cdot l_e$ (Figure 8b).

It should be noted that for upward bending, condition III is stronger than the statement "if $\psi_t > \psi_S$, then $\theta \overset{!}{<} \psi_S$." As shown in Figure 8b, there exists a case where $\theta < \psi_S$, but the solution is false. Such a strong formulation is not possible for downward bending, i.e., condition II.

2.11 Getting from numerical solution to generalized coordinates

w is a design parameter of the robot that describes the distance between the backbone and the imaginary centerline. This distance is identical to the distance between the connectors and the imaginary centerline. The radius r denotes the radius of the imaginary centerline, as shown in Figure 1.

Using the numerically computed deflection angle θ and radius r, the length of the bending part (i.e., l_b) and the difference between l_b and the length of the backbone (i.e., Δl_b) are computed according to Equations 33, 34 for downward bending.

$$l_b = (r - w) \cdot (|\psi| + \theta), \tag{33}$$

$$\Delta l_b = (r + w) \cdot (|\psi| + \theta) - l_b. \tag{34}$$

l_b and Δl_b for upward bending are computed according to Equations 35, 36.

$$l_b = (r + w) \cdot (\psi - \theta), \tag{35}$$

$$\Delta l_b = (r - w) \cdot (\psi - \theta) - l_b. \tag{36}$$

To derive the resulting end-effector pose based on the NSGC solution for the generalized joint coordinates $\mathbf{q} = [l_b, \Delta l_b, \alpha, \theta]$, $\mathbf{q}$ has to be plugged into the forward kinematic model defined by Equation 2. The resulting end-effector pose can then be compared to the target pose, with the target pose $\mathbf{X}_t$.

2.12 Overview of NSGC

In summary, the final NSGC algorithm is expressed in Algorithm 2. Here, $\mathbf{X}_t$ denotes the original target pose in 3D, $\widehat{\mathbf{X}}_t$ denotes the 2D projection, and $\tilde{\mathbf{X}}_t$ denotes a rotation thereof.

The flowchart in Figure 9 visualizes the individual steps of the NSGC algorithm.

3 In silico validation

The proposed NSGC algorithm was implemented and compared with four state-of-the-art IKMs following a helical trajectory: two basic Jacobian-based models with k-factor values of 0.4 and 0.5, one Levenberg–Marquardt (LM) implementation with adaptive damping, and one quadratic programming approach. The LM technique is also known as damped least squares in optimization. A k-factor of 0.5 was the highest value that still allowed all target poses to be achieved. The comparison included the accuracy, median, mean, range, IQR, and standard deviation of the convergence times for 200 target poses $\mathbf{X}_t = [x_t, y_t, z_t, \psi_t]$ along the helix.

3.1 Convergence time and behavior

The convergence times for all five IKMs are visualized in Figure 10 and summarized in Table 1. The mean convergence times of basic Jacobians and the adaptive damping LM IKMs are very similar to each other (in the range of 21 ms29 ms), which is significantly faster than 100 ms needed for real-time applications. NSGC achieves a mean convergence time of 16 ms, which is approximately 24% faster than the fastest competing IKM (i.e., the adaptive damping IKM). As shown in Figure 10, the quadratic programming approach converges with a mean time of 143 ms, which is notably slower than that of the four other models. Therefore, the quadratic programming IKM is not usable for real-time applications.

It has to be noted that the spread of convergence time of NSGC (39 ms) is notably higher than that of all other IKMs, ranging from 6 ms – 28 ms. This large spread of 39 ms can be explained by the necessary iteration through many initial guesses in case the previous initial guess did not lead to a convergence. However, even the longest measured convergence time was only ≈40 ms. In contrast, in some cases, the very first initial guess led to convergence, which resulted in an extremely fast convergence time of less than 5 ms, outperforming all other models. IQR stands for interquartile range and measures statistical dispersion, specifically representing the spread of the middle 50% of a dataset.

The convergence behavior of NSGC is displayed in Figure 11. As observed, NSGC iterates through different initial guesses until it finds the correct solution for the set of equations defined in Section 2.3. It is evident that NSGC's convergence behavior is fundamentally different from that of all other IKM approaches (Figure 11) as there is no convergence in the sense that the error progressively approaches 0. Rather, the convergence of an iteration is completely independent of the previous iteration's

Receive $\mathbf{X}_t \leftarrow [x_t, y_t, z_t, \psi_t]$
▷ target pose in 3D space
Project it into 2D space
$\widehat{\mathbf{X}}_t \leftarrow [\hat{x}_t, z_t, \psi_t]$ [Equation 4]
Compute α [Equation 5]
Determine robot configuration [Equations 6, 7]
for $i = 1 \rightarrow n$ **do**
  $\mathbf{j}_0^i \leftarrow [(\hat{x}_0)_0, (z_0)_0, (\theta)_0, (r)_0]$ ▷ initial guesses
end for
if robot configuration = downward bending, **then**
 Define a set of equations $\mathbf{f}_{\widehat{\mathbf{X}}_t}$ [Equations 8–11] with [Equations 12–15]
 while convergence conditions are not met **do**
  $\mathbf{j} \leftarrow \text{PowellSolve}(\mathbf{f}_{\widehat{\mathbf{X}}_t}, \mathbf{j}_0^i)$ [Equation 22]
  $\mathbf{j}_0^i \leftarrow \mathbf{j}_0^{i+1}$
 end while
 Compute l_b and Δl_b [Equations 33, 34].
else if robot configuration = upward bending, **then**
 Define a set of equations $\mathbf{f}_{\tilde{\mathbf{X}}_t}$ [Equations 8–11] with [Equations 16–19]
 while Convergence conditions are not met **do**
  $\mathbf{j} \leftarrow \text{PowellSolve}(\mathbf{f}_{\tilde{\mathbf{X}}_t}, \mathbf{j}_0^i)$ [Equation 22]
  $\mathbf{j}_0^i \leftarrow \mathbf{j}_0^{i+1}$
  if no convergence, **then**
   while Convergence conditions are not met **do**
    $\tilde{\mathbf{X}}_t \leftarrow [-\hat{x}_t, z_t, -\psi_t]$ ▷ rotate by $\alpha = 180°$
    $\mathbf{j} \leftarrow \text{PowellSolve}(\mathbf{f}_{\tilde{\mathbf{X}}_t}, \mathbf{j}_0^i)$ [Equation 22]
    $\mathbf{j}_0^i \leftarrow \mathbf{j}_0^{i+1}$
   end while
   Update $\alpha \leftarrow \alpha + \pi$
  end if
 end while
 Compute l_b and Δl_b [Equations 35, 36]
else ▷ Straight configuration
 Compute l_b [Equation 21]
 $\Delta l_b \leftarrow 0$
end if
Send $\mathbf{q} = [l_b, \Delta l_b, \alpha, \theta]$ to motors

Algorithm 2. NSGC.

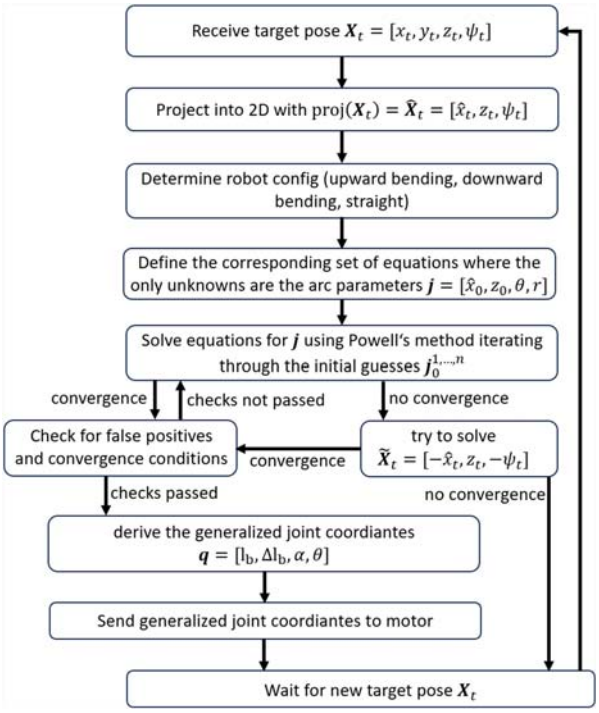

FIGURE 9
NSGC flowchart.

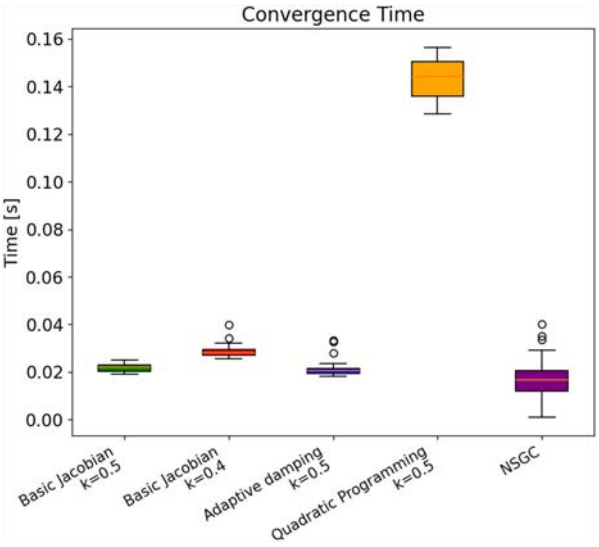

FIGURE 10
Average time to solve 1,000 randomly generated 3D poses.

convergence, as it depends solely on the initial guess. Note that the convergence time is not only influenced by the number of iterations until convergence but also by the time needed per iteration. Generally, NSGC and the quadratic programming approach need more time per iteration.

3.2 Memory footprint and computational complexity

The NSGC algorithm, implemented in Python, exhibits a moderate memory footprint and computational complexity.

3.2.1 Memory footprint

The memory usage of NSGC is primarily driven by the number of initial guesses for solving the nonlinear system, with each being a 4-dimensional vector $(z_0, \hat{x}_0, \theta, r)$ stored in the `initial_guesses` list. Let n be the number of such guesses. Each guess is a list of four floating-point numbers (8 bytes each), so the memory for the guesses alone is $\mathcal{O}(n)$, which is typically a few hundred bytes to a few kilobytes. Additional memory is consumed by temporary arrays during forward kinematics computation, but due to Python's garbage collection and the linear chaining of matrices, peak memory usage remains modest—typically only a few megabytes.

3.2.2 Computational complexity

The dominant cost arises from numerically solving a nonlinear system of four equations with four unknowns using Powell's method. For each initial guess, it performs iterative updates until convergence, which typically takes k iterations per guess, with each requiring evaluation of the function and Jacobian approximation.

Let n be the number of initial guesses and k be the number of iterations per call to Powell's method.

The overall complexity becomes

$$\mathcal{O}(n \cdot k \cdot c),$$

where c is the cost of evaluating the nonlinear system (consisting of elementary operations). Since most branches terminate early upon finding a valid solution, the average-case complexity is significantly lower than the worst-case complexity.

3.2.3 Practical performance

Empirical results indicate solution times in the range of tens of milliseconds on a modern CPU, depending on the initial guess set and configuration. Memory remains bounded and scales linearly with the resolution and number of guesses, making the algorithm suitable for embedded applications with moderate resources.

3.3 Robustness to noise, initial guess sensitivity, and singularities

The NSGC algorithm demonstrates high robustness in practical inverse kinematic scenarios but retains certain sensitivities that merit discussion.

3.3.1 Robustness to noise

When using NSGC, the user has free choice of the position (x_t, y_t, z_t) defined in the global coordinate system (CS) and free choice of the orientation (ψ_t) defined as the rotation about the y_ε-axis of local end-effector CS. Noise in these four target dimensions is inconsequential, as the noisy target pose $\mathbf{X}_{t,noise} = [x_t + \eta_1, y_t + \eta_2, z_t + \eta_3, \psi_t + \eta_4]$ will be solved by NSGC, with η representing random noise. The remaining two orientations about

TABLE 1 Overview of simulation results.

Metric	Basic Jacobian k = 0.5	Basic Jacobian k = 0.4	Adaptive damping k = 0.5	Jacobian with QP	NSGC
Successfully reached poses [%]	100	100	100	100	100
Median time [ms]	22	28	21	144	17
Mean time [ms]	22	29	21	143	16
Range [ms]	6	14	15	28	39
IQR [ms]	3	2	2	15	9
Standard deviation [ms]	1.55	2.39	2.93	8.01	7.94

the local x_ε- and z_ε-axis of the end-effector are always set to 0, indicating that noise in these target dimensions will also not influence the convergence behavior.

3.3.2 Sensitivity to initial guess

The choice of initial guesses significantly impacts both convergence success and computational efficiency. While NSGC employs a coarse angular sweep strategy to sample possible solution regions, poorly chosen or sparse guess sets can lead to missed solutions or divergence. Empirical evidence suggests that a resolution of approximately 10° for the initial guess of $(\theta)_0$ and approximately 30 mm for the initial guess of $(\hat{x}_0)_0, (z_0)_0$ and $(r_0)_0$ provides a good trade-off between coverage and runtime. Due to the forgiving nature of Powell's method, it often converges even with suboptimal initial guesses; e.g., $|(\hat{x}_0)_0 - \hat{x}_0| > 50\,\text{mm}$.

3.3.3 Singularities and degenerate configurations

The algorithm can struggle near geometric singularities, such as when the NiTi segment length l_b approaches 0. The most effective mitigation strategy in this case is to either reduce the length of the distal part l_d and end-effector l_e or to relocate the robot's point of deployment further away from the workspace so that the bending section is long enough to provide the needed maneuverability. The most common singularity occurs when a straight configuration is approached (typically $|\psi_t - \psi_S| < 1°$). This scenario and the appropriate mitigation strategy are described in Section 2.6.

Overall, good initial guesses, checking for false positives, and establishing a straight-line case for divergent bending radii ensure NSGC's performance, stability, and accuracy considerably.

3.4 Simulated case study

As CRs/HRRs are used in endoscopy (Sun and Chen, 2024) and biopsy procedures (Gao et al., 2020), this study uses endoscopic examination of the gallbladder as a case study. During a biopsy, the physician traverses the surface of the organ, and once a suitable location is identified, the biopsy needle is inserted into the tissue.

According to Can et al. (2012), the average gallbladder size is $h \times w \times l = 50\,\text{mm} \times 50\,\text{mm} \times 30\,\text{mm}$. Hence, in this study, a cuboid was used to represent the size of the gallbladder. For this simulated case study, NSGC was used to solve the trajectory of the end-effector traversing the cuboid's surface. The full simulation video is available in the Supplementary Material.

The following section describes the robotic prototype used for the experimental validation.

4 HyNiTi prototype

The robotic prototype is referred to as a hyper-redundant, NiTi-based robot (abbreviated as HyNiTi). The robot's dexterity stems from its ability to adjust the length and incorporate an elbow joint, enabling multiple orientations at the same position (Fritsch et al., 2024).

The HyNiTi features four motors (motors 1–4 in Figure 12) for robot actuation, resulting in four DoFs of the robot. Additionally, there is one motor (motor 5 in Figure 12) for instrument actuation (i.e., advancing and retracting the biopsy needle). The entire robot is rotated about its main axis by approximately α by motor 1. Motor 2 changes the length of the bending part (i.e., l_b), while motor 3 actuates the elbow joint, causing a deflection in the HRR away from the proximal part's main axis. Motor 4 changes the length of the backbone (i.e., Δl_b), causing the robot to bend. For $\Delta l_b < 0$, the robot bends upward, and for $\Delta l_b > 0$, the robot bends downward. A clear overview of the relationship between the motor and the DoF is provided in Table 2. Notably, NSGC solves α, θ, l_b and Δl_b. The needle actuation is controlled by the user directly.

The endoscope (K-FLEX-XC1, Karl Storz SE & Co. KG, Tuttlingen, Germany) is routed through the HRR to its end-effector. The endoscope is not directly actuated but complies with the shape of the HRR. The endoscope features a 3 mm outer diameter, a light source, a camera, and a working channel. The entire robot, including the actuation and electronic components, is shown in Figure 13a, whereas Figures 13b, c show the HRR and the elbow joint in more detail.

4.1 HRR actuation

HyNiTi is an HRR that relies on backbone actuation. The backbone is made of highly elastic metal, such as superelastic

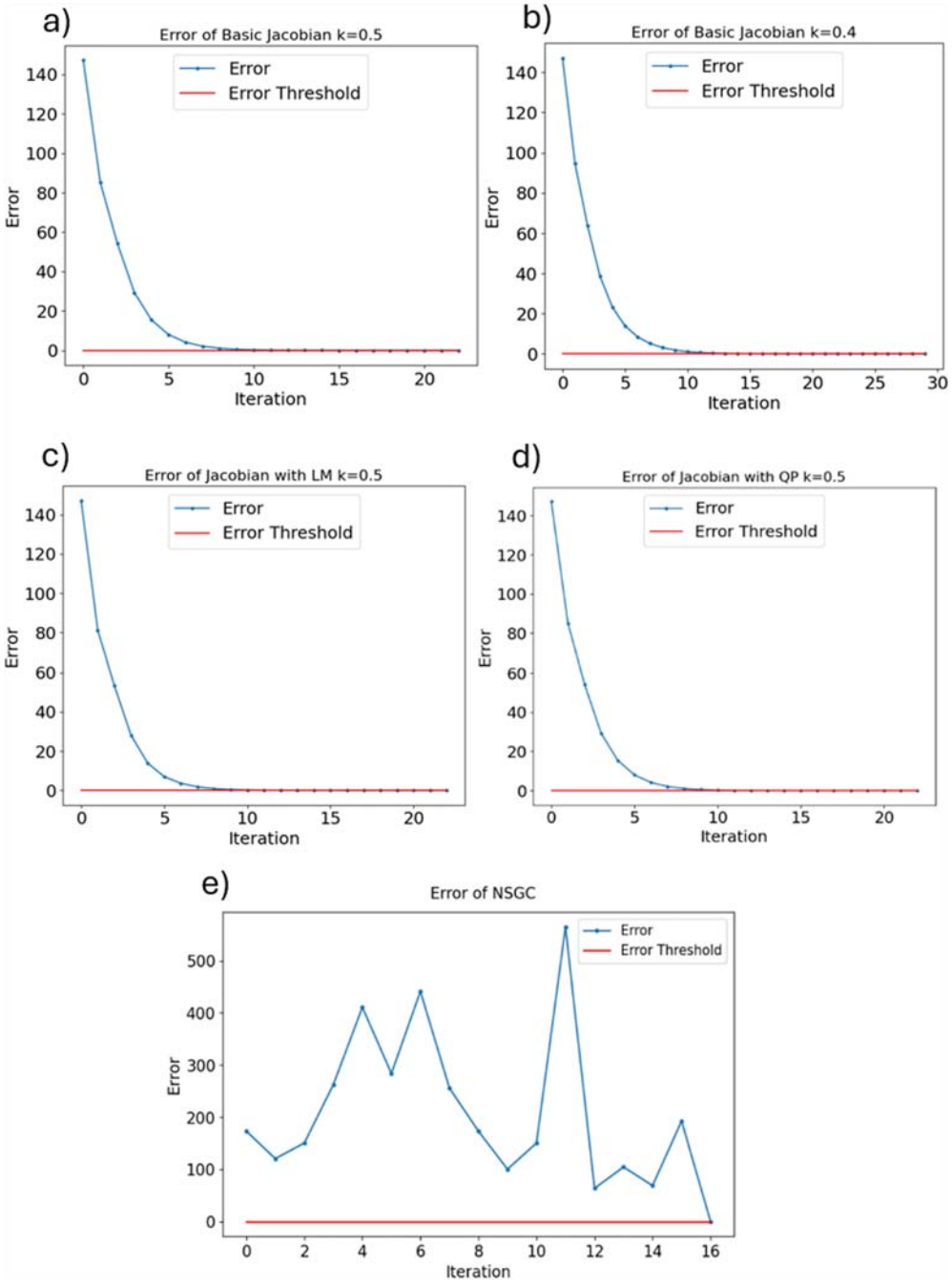

FIGURE 11
Comparison of convergence behavior of NSGC vs. Jacobian-based IKMs with an error threshold of 1e-5. **(a)** the basic jacobian IKM with k=0.5 converges after 22 steps. **(b)** the basic jacobian IKM with k=0.4 converges after 29 steps. **(c)** the LM IKM converges after 22 steps. **(d)** the QP IKM converges after 22 steps, but each step takes significantly longer than using the basic Jacobian-based IKM. **(d)** NSGC needs 16 steps.

NiTi or spring steel. Unlike tendons, the backbone is resistant to stretching and backlash, resulting in more stable and predictable robotic behavior.

Another advantage of this design is the reduced number of channels within the robotic elements. Tendon-driven systems typically require separate channels for each bending direction (e.g., one tendon for upward bending and another for downward bending). In contrast, the pull–push backbone actuation requires only one channel to facilitate movement in two opposing directions, enabling further miniaturization of the robot.

4.2 Elbow joint design

The prototype utilizes an elbow joint that is based on a rack–and-pinion mechanism (Fritsch et al., 2023). The rack is

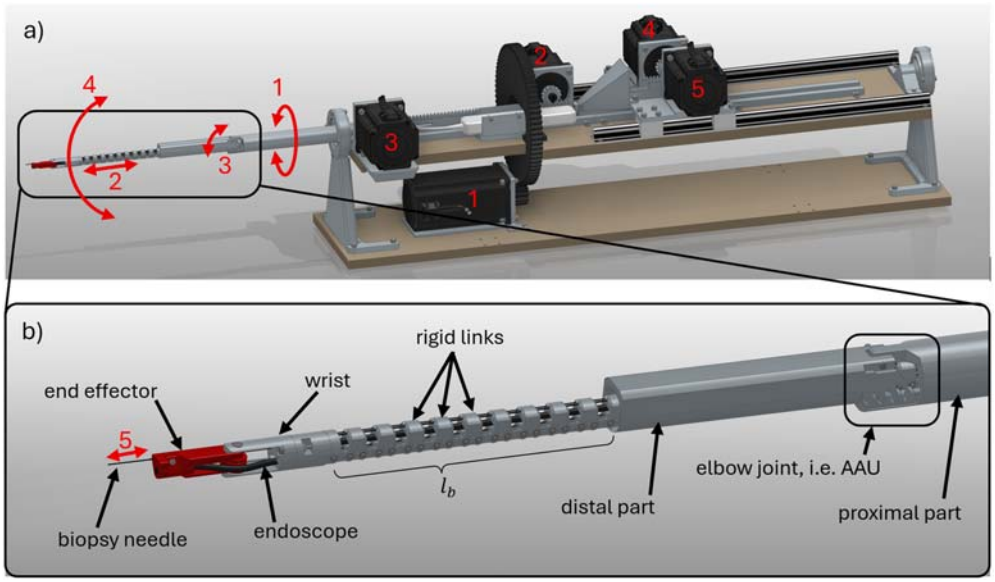

FIGURE 12
CAD model of HyNiTi: **(a)** complete robot including motors and **(b)** details of HRR; scale: l_b = 200 mm in this figure.

TABLE 2 Overview of motors and DoF.

Motor	1	2	3	4	5
DoF	α	l_b	θ	Δl_b	Needle

pushed or pulled by a motor (in this case, by motor 3), causing the pinion to rotate about the axis of rotation. The pinion is structurally integrated into the distal part. To mitigate the risk of tilting or jamming of the distal part, the rack and pinion mechanism is designed with redundancy, incorporating a two-sided configuration.

The elbow joint's deflection angles are in the range of $0° = \theta_{\min} \leq \theta \leq \theta_{\max} = 60°$, where $\theta_{\min}$ and $\theta_{\max}$ are the mechanical limits.

5 Experimental validation

The NSGC algorithm was tested using the HyNiTi robot. The complete test setup is shown in Figure 13a. A stereo camera (ZED 2i, Stereolabs, San Francisco, United States) was used for the optical measurement of the achieved poses. The depth information enables the localization of points in 3D so that the x, y, and z coordinates can be derived with respect to the camera. The camera is extrinsically calibrated to the global coordinate system, which is located at the robot's base. Each of the measured points is transformed into the global coordinate system.

In this setup, each stepper motor features an optical incremental encoder, and the drivers handle the PID control internally; thus, there is low-level closed-loop control. The joint position commands are generated using NSGC and then fed to the stepper motor drivers using a microcontroller.

5.1 Individual target poses

A total of 100 target poses $\mathbf{X}_t$ were solved using NSGC, the generalized coordinates $\mathbf{q} = [l_b, \Delta l_b, \alpha, \theta]$ were calculated and sent to the motors, and the achieved pose was measured and compared to the target pose. A picture including the depth information of each pose was captured using the stereo camera and then analyzed. The analysis was based on two points: the proximal end of the wrist and the distal end of the end-effector. From the coordinates of both points, the achieved orientation about the local y_ε-axis was derived. The differences between the target and measured poses showed good agreement, as shown in Figure 14 and Table 3. RMSE stands for root mean square error.

As shown in Figure 14, the range of error was the highest for the end-effector angle ψ_t, with $\Delta\psi$ ranging from −4° to 6° with an RMSE of 2.09°. The RMSE for all three positional coordinates ranged from 1.7 mm along the global z-axis to 3.53 mm along the global x-axis. The mean positional error ranged from 0.42 mm to 2.82 mm. A leading factor contributing to the pose error is the weight of the robot itself, which induces downward bending. The lack of rigidity can be alleviated to some degree by appropriate robot calibration and by increasing the bending section's stiffness.

5.2 Trajectory tracking

For the application of CR/HRR technology in the real world, good trajectory tracking performance is crucial. In this experiment, three trajectories were tracked: a linear 200 mm movement in the x-direction, a linear 200 mm movement in the y-direction, and a movement along a 3D helix. The helix was characterized by a radius of 50 mm and a length of 200 mm along the z-axis, as shown in Figure 15. The error metrics are given in Table 4.

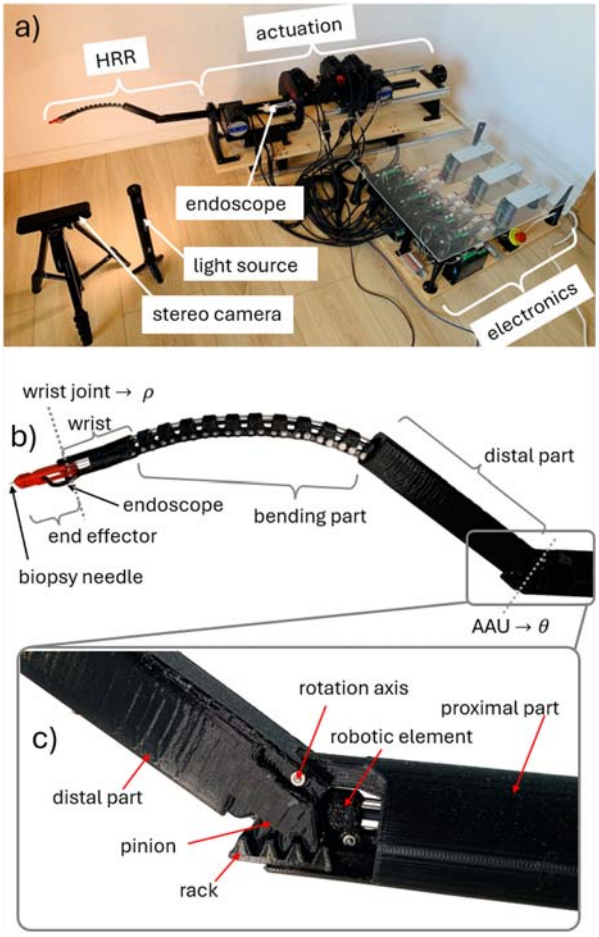

FIGURE 13
Robotic prototype: **(a)** test setup, **(b)** HRR details, and **(c)** elbow joint for scale: distal part of this prototype is $l_d = 130$ mm long.

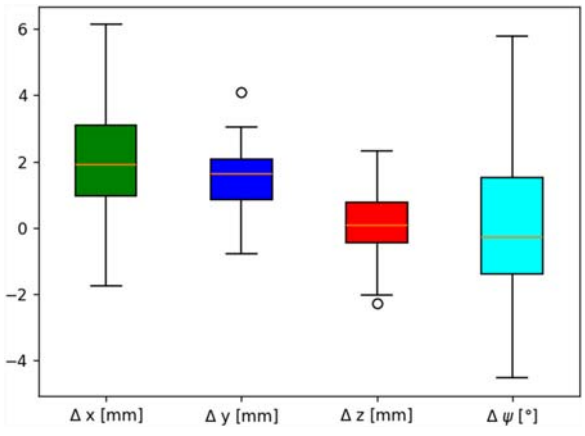

FIGURE 14
Deviation between the measured pose and target pose for 100 individual target poses in 3D.

The inverse kinematics for each point on the helix was computed using the proposed NSGC algorithm. The orientation at each point was kept constant; in other words, the end-effector reached each pose on the trajectory with the same orientation. In all three cases, the target trajectory was compared against the measured trajectory. During trajectory tracking, an image was captured and analyzed every 0.5 s. The trajectory tracking validation confirms that NSGC can be used for the control of length-extensible CRs/HRRs with an elbow joint since all poses along the path were successfully reached, as shown in Figure 15.

The average positional error during trajectory tracking was 4.05 mm along the x-axis and 4.22 mm along the y-axis. These errors are slightly higher than those observed in the measurements of individual points (Section 5.1). This discrepancy is most likely caused by the continuous movement of the robot's end-effector during trajectory tracking, compared to the stationary end-effector used when measuring individual points. With an average positional error of 5.31 mm, trajectory tracking when following the helical path was considerably less accurate than the linear trajectories along the x- and y-axes. This is most likely due to the added dynamics of the end-effector's movement along the z-direction. The superposition of movements along all three axes decreases the overall accuracy.

The inaccuracies are due to the mechanical structure of the robot: play in the joints connecting the rigid links in the bending section and a relatively heavy end-effector, which causes low-frequency mechanical vibrations.

5.3 Case study: gallbladder biopsy

The HyNiTi prototype, equipped with a biopsy needle and an endoscopic camera (as described in Section 4), was used in this case study to simulate a biopsy on a piece of meat representing the gallbladder. The objective of this case study is to demonstrate NSGC's effectiveness in real-life applications.

Three distinct target locations were selected on the tissue sample. Position 1 was aligned along the main axis of the proximal region, while positions 2 and 3 were located on the top and bottom surfaces of the tissue, respectively. Reaching these off-axis targets required actuation of the robot's rotational degree of freedom, denoted as α. As shown in Figure 16, the robot was able to reach all three points on the tissue without any issue.

6 Conclusion

In this article, a novel inverse kinematic model termed NSGC was introduced—a numerical solution of a set of equations representing geometric constraints. NSGC is well-suited for length-extensible CRs and HRRs featuring an elbow joint.

A target pose $\mathbf{X}_t = [x_t, y_t, z_t, \psi_t]$ is given in 3D space, where x_t, y_t, and z_t are the global coordinates and ψ_t is the end-effector orientation about the local y_ε-axis. $\mathbf{X}_t$ is projected onto a plane with $\text{proj}(\mathbf{X}_t) = \widetilde{\mathbf{X}}_t = [\tilde{x}_t, z_t, \psi_t]$. The planar arc is described using four equations. The arc parameters $\mathbf{j} = [\tilde{x}_0, z_0, \theta, r]$ are unknown and

TABLE 3 Statistical error for individual poses.

Error metrics	x	y	z	ψ
RMSE	3.53 mm	3.03 mm	1.70 mm	2.09°
Mean	2.82 mm	2.63 mm	0.42 mm	−0.20°
Standard deviation	2.13 mm	1.52 mm	1.65 mm	2.09°
95% Confidence interval	(2.40 mm, 3.25 mm)	(2.33 mm, 2.93 mm)	(0.09 mm,0.74 mm)	(-0.62°,0.24°)

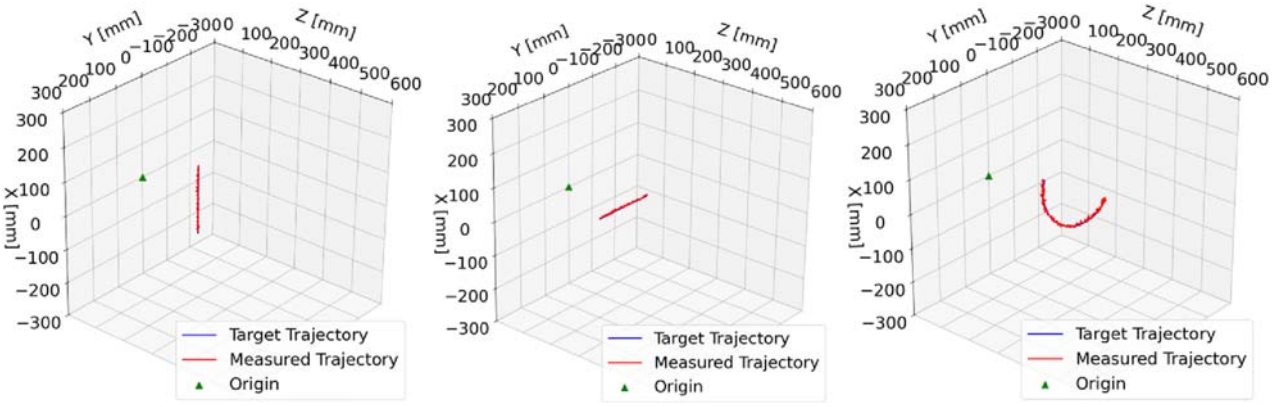

FIGURE 15
Trajectory tracking: left plot: end-effector tracking along the x-axis from +100 mm to −100 mm with 20 *s* duration; center plot: end-effector tracking along the y-axis from +100 mm to −100 mm with 20 *s* duration; right plot: end-effector tracking along a helix path with 30 *s* duration.

TABLE 4 Statistical error for trajectory tracking.

Error metrics	Trajectory along x	Trajectory along y	Helix
RMSE [mm]	4.44	4.47	5.75
Mean [mm]	4.05	4.22	5.31
Standard deviation [mm]	1.80	1.46	2.20
Maximum error [mm]	8.27	7.88	12.05
Minimum error [mm]	0.56	1.19	0.64

solved numerically using Powell's hybrid method while iterating through appropriate initial guesses. For every potential numerical solution for these arc parameters, it is checked whether convergence conditions I–III are met to eliminate false positives. When all conditions are satisfied, the arc parameters $\mathbf{j}$ are used to compute the generalized joint coordinates $\mathbf{q} = [l_b, \Delta l_b, \alpha, \theta]$, which are then sent to the motors. *In silico* validation shows that NSGC solves the given target poses in 16 ms on average and has a median convergence time of 17 ms. Hence, NSGC can be used for real-time applications. NSGC is 24% faster than the next fastest IKM (Jacobian-based with adaptive damping). Therefore, NSGC could be used in scenarios where minimizing latency between the user input (defining the target pose) and the robot's movement is critical.

The HRR prototype relied on backbone actuation, which proved to work reliably and without any backlash or elongation compared to tendon actuation. The elbow joint allowed for precise and repeatable deflection of the HRR away from the proximal part's main axis. Changing the orientation at the same position was enabled by the combination of the elbow joint and the length adjustment of the HRR.

The limitation of NSGC is the narrow range of kinematic structures for which it can be used; the discussed CR/HRR structure with length-adjustment capabilities and an elbow joint can only achieve a position in 3D space and with one orientation about the local y_ε-axis. For such robots, the end-effector orientation about the local x_ε- and z_ε-axes are necessarily 0°. This is the reason why NSGC is a good fit for such kinematics.

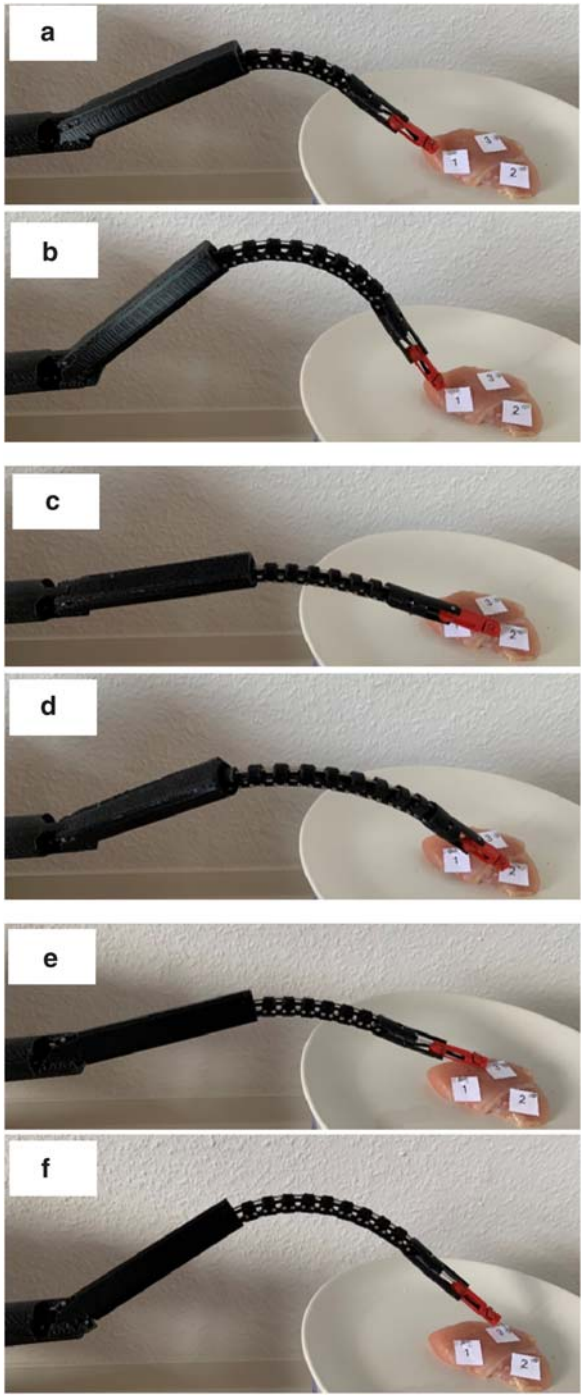

FIGURE 16
Case study of three target locations on the tissue: **(a)** target position 1 is reached at an angle ψ, which is changed in **(b)**. Target position 2 is located to the right of the robot's main axis so that the robot rotates in the counterclockwise direction $\alpha > 0°$. **(c)** and **(d)** show different end-effector orientations ψ at position 2. Target position 3 is located to the left of the robot's main axis so that the robot rotates in the clockwise direction $\alpha < 0°$. The target position can also be reached with a variation in ψ, as shown by the difference in orientation between **(e,f)**.

Incorporating additional orientations about the x_ε- and z_ε-axes of the end-effector poses a challenge since in that case, the 2D projection does not work as elegantly as in the addressed scenario.

In the current trajectory-tracking validation, NSGC is used to compute the generalized joint coordinates q individually. The addition of velocity/acceleration planning via interpolation or spline fitting would further improve the smoothness of the achieved trajectory. Furthermore, if the robot's speed needs to be increased for high-dynamic use cases, using real-time trajectory generators instead of point-to-point stepping would also improve the timing control.

A drawback is the diverging radius as the robot configuration approaches a straight line (i.e., $\psi_t = \psi_S$). To avoid solving NSGC for extremely large radii (e.g., $r > 1{,}000$), which exclusively occur for very slight bending (e.g., $|\psi_t - \psi_S| < 1°$), the robot is considered to be in a straight configuration. Thus, end-effector orientation changes of less than 1° difference from ψ_S cannot be realized with NSGC but are approximated as a straight configuration with $\psi_t = \psi_S$.

Another limitation of this study is that only the positions were considered for trajectory tracking. In future studies, orientation should also be taken into consideration, as was done during the measurement of the individual target poses.

Data availability statement

Publicly available datasets were analyzed in this study. This data can be found here: https://github.com/SvenFritschResearch/NSGC/tree/main.

Author contributions

SF: Conceptualization, Formal Analysis, Methodology, Software, Writing – original draft, Writing – review and editing. DO: Funding acquisition, Project administration, Resources, Supervision, Writing – review and editing.

Funding

The author(s) declare that financial support was received for the research and/or publication of this article. This open access publication was supported by the publication fund of Technical University Berlin.

Acknowledgments

The authors express gratitude to Karl Storz SE & Co. KG for supporting this research by providing the endoscope.

Conflict of interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Technical University Berlin has patented the angle adjustement unit under file number 10 2021 128 809.6 at the German Patent Office. Inventors are Sven Fritsch and Dirk Oberschmidt.

Generative AI statement

The author(s) declare that no Generative AI was used in the creation of this manuscript.

Publisher's note

Supplementary material

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/frobt.2025.1627688/full#supplementary-material

References

Almanzor, E., Ye, F., Shi, J., Thuruthel, T. G., Wurdemann, H. A., and Iida, F. (2023). Static shape control of soft continuum robots using deep visual inverse kinematic models. *IEEE Trans. Robotics* 39, 2973–2988. doi:10.1109/tro.2023.3275375

Bajo, A., Goldman, R. E., Wang, L., Fowler, D., and Simaan, N. (2012). "Integration and preliminary evaluation of an insertable robotic effectors platform for single port access surgery," in *IEEE international conference on robotics and automation*.

Burgner-Kahrs, J., Rucker, D., and Choset, H. (2015). "Continuum robots for medical applications: a survey," in *IEEE transactions on robotics*.

Can, S., Staub, C., Knollm, A., Fiolka, A., Schneider, A., and Feussner, H. (2012). "Design, development and evaluation of a highly versatile robot platform for minimally invasive single-port surgery," in *The fourth IEEE RAS/EMBS international conference on biomedical robotics and biomechatronics*.

Chen, H.-S., and Stadtherr, M. (1981). *A modification of Powell's dogleg method for solving systems of nonlinear equations*. Urbana, IL: Computers & Chemical Engineering, University of Illinois, 143–150.

Chirikjian, G., and Burdick, J. W. (1994). "A hyper-redundant manipulator," in *IEEE robotics and automation magazine*.

Fritsch, S., and Oberschmidt, D. (2023). Articulating robotic arm for minimally invasive surgery, surgical robot and method for production. *U. S. Pat. App* 17/981, 279. Available online at: https://patents.google.com/patent/US20230145905A1/en.

Fritsch, S., and Oberschmidt, D. (2024). Getting from a 3D, dexterous, single-port workspace to a one-segment continuum robot. *Mechatronics* 101, 103194. doi:10.1016/j.mechatronics.2024.103194

Gao, Y., Takagi, K., Kato, T., Shono, N., and Hata, N. (2020). Continuum robot with follow-the-leader motion for endoscopic third ventriculostomy and tumor biopsy. *IEEE Trans. Biomed. Eng.* 67 (2), 379–390. doi:10.1109/tbme.2019.2913752

Garg, A., Vikram, C. S., and Gupta, V. K. (2014). "Design and development of *in vivo* robot for biopsy," in *Mechanics based design of structures and machines: an international journal*.

Giorelli, M., Renda, F., Ferri, G., and Laschi, C. (2013). *A feed-forward neural network learning the inverse kinetics of a soft cable-driven manipulator moving in three-dimensional space*. IEEE/RSJ International Conference on Intelligent Robots and Systems IROS.

Guardiani, P., Ludovico, D., Pistone, A., Abidi, H., Zaplana, I., Lee, J., et al. (2022). "Design and analysis of a fully actuated cable-driven joint for hyper-redundant robots with optimal cable routing," in *Journal of mechanisms and robotics*.

John, B. (1983). "Kinematic programming alternatives for redundant manipulators," in *Proceedings of the IEEE international conference on robotics and automation*.

Jones, B. A., and Walker, I. D. (2006). "Kinematics for multisection continuum robots," in *IEEE transactions on robotics*.

Kim, Y.-J., and Wi, D. (2025). Planar inverse statics and path planning for a tendon-driven discrete continuum robot. *Robotics* 14, 91. doi:10.3390/robotics14070091

Klein, C. A., and Huang, C.-H. (1983). "Review of pseudoinverse control for use with kinematically redundant manipulators," in *IEEE transactions on systems, man, and cybernetics*.

Lai, J., Huang, K., and Henry, K. (2019). "Chu. "A learning-based inverse kinematics solver for a multi-segment continuum robot in robot-independent mapping"," in *Proceedings of the IEEE international conference on robotics and biomimetics*.

Lee, J., Kim, J., and Choi, J.-Y. (2014). *Modeling and control of robotic surgical platform for single-port access surgery*. IEEE/RSJ International Conference on Intelligent Robots and Systems.

Li, M., Kang, R., Geng, S., and Guglielmino, E. (2017). "Design and control of a tendon-driven continuum robot," in *Transactions on the institute of measurement and control*.

Li, Z., Zheng, S., Zhang, Y., Li, H., Jiang, Z., and Shi, P. (2025). "Multilevel motion control of cable-driven hyper-redundant manipulators," in *IEEE/ASME transactions on mechatronics*.

Powell, M. (1970). "A hybrid method for nonlinear equations," in *Numerical methods for nonlinear algebraic equations*, 87–144.

Rone, W. S., and Ben-Tzvi, P. (2014). "Mechanics modeling of multisegment rod-driven continuum robots," in *Journal of mechanisms and robotics*.

Seraji, H. (1989). "Configuration control of redundant manipulators: theory and implementation," in *IEEE transactions on robotics and automation*.

Sheng, Z., Wang, Y., Li, S., Yang, J., Xie, H., and Lu, X. (2024). Research on kinematics modeling and path planning of a hyper-redundant continuum robot. *Proc. Institution Mech. Eng. Part C J. Mech. Eng. Sci.* 238 (12), 5880–5890. doi:10.1177/09544062231224967

Simaan, N., Xu, K., and Taylor, R. (2009). "Design and integration of a telerobotic system for minimally invasive surgery of the throat," in *International journal of robotics research*.

Sun, L., and Chen, X. (2024). "Flexible continuum robot system for minimally invasive endoluminal gastrointestinal endoscopy," in *Machines 12*.

Thuruthel, T., Falotico, E., and Laschi, C. (2016). "Learning global inverse kinematics solutions for a continuum robot," in *Dynamics and control*.

Wang, Z., Hu, D., Wan, D., and Liu, C. (2024). An improved inverse kinematics solution method for the hyper-redundant manipulator with end-link pose constraint. *J. Field Robotics* 41 (6), 1900–1921. doi:10.1002/rob.22362

Wei, H., Zhang, G., Wang, S., Zhang, P., Su, J., and Du, F. (2023). "Coupling analysis of compound continuum robots for surgery: another line of thought," in *Sensors*.

Whitney, D. E. (1972). "The mathematics of coordinated control of prosthetic arms and manipulators," in *Journal of dynamic systems, measurement, and control*.

Wild, S., Zeng, T., Mohammad, A., Billingham, J., Axinte, D., and Dong, X. (2023). "Efficient and scalable inverse kinematics for continuum robots," in *IEEE robotics and automation letter*, 9.1.

Wolf, A., Howard, H. C., and Randall, C. (2005). "Design and control of a mobile hyperredundant urban search and rescue robot," in *Advanced robotics*.

Xu, W., Liu, T., and Li, Y. (2018). "Kinematics, dynamics, and control of a cable-driven hyper-redundant manipulator," in *IEEE/ASME transactions on mechatronics*.

Yeshmukhametov, A., Koganezawa, K., and Yamamoto, Y. (2019). "A novel discrete wire-driven continuum robot arm with passive sliding disc: design, kinematics and passive tension control," in *Robotics*.

Zhan, L., Du, H., Qin, Z., Zhao, Y., and Yang, G. (2024). Real-time optimized inverse kinematics of redundant robots under inequality constraints". *Sci. Rep.* 14 29754. doi:10.1038/s41598-024-81174-8