applied sciences

Article

A General Approach Based on Newton's Method and Cyclic Coordinate Descent Method for Solving the Inverse Kinematics

Yuhan Chen [1], Xiao Luo [2,*], Baoling Han [1], Yan Jia [1], Guanhao Liang [3] and Xinda Wang [1]

[1] School of Mechanical Engineering, Beijing Institute of Technology, No. 5 Zhongguancun South Street, Haidian District, Beijing 100081, China; 3120185244@bit.edu.cn (Y.C.); hanbl@bit.edu.cn (B.H.); tictac0324@163.com (Y.J.); wangxinda1111@163.com (X.W.)

[2] School of Computer Science and Technology, Beijing Institute of Technology, No. 5 Zhongguancun South Street, Haidian District, Beijing 100081, China

[3] School of Mechatronical Engineering, Beijing Institute of Technology, No. 5 Zhongguancun South Street, Haidian District, Beijing 100081, China; simonleungbit@hotmail.com

* Correspondence: luox@bit.edu.cn; Tel.: +86-010-6891-8856

Received: 22 October 2019; Accepted: 9 December 2019; Published: 12 December 2019

Featured Application: The new approach can be used to solve the inverse kinematics of revolute and prismatic joint robots with any number of degrees of freedom as well as any configurations.

Abstract: The inverse kinematics of robot manipulators is a crucial problem with respect to automatically controlling robots. In this work, a Newton-improved cyclic coordinate descent (NICCD) method is proposed, which is suitable for robots with revolute or prismatic joints with degrees of freedom of any arbitrary number. Firstly, the inverse kinematics problem is transformed into the objective function optimization problem, which is based on the least-squares form of the angle error and the position error expressed by the product-of-exponentials formula. Thereafter, the optimization problem is solved by combining Newton's method with the improved cyclic coordinate descent (ICCD) method. The difference between the proposed ICCD method and the traditional cyclic coordinate descent method is that consecutive prismatic joints and consecutive parallel revolute joints are treated as a whole in the former for the purposes of optimization. The ICCD algorithm has a convenient iterative formula for these two cases. In order to illustrate the performance of the NICCD method, its simulation results are compared with the well-known Newton–Raphson method using six different robot manipulators. The results suggest that, overall, the NICCD method is effective, accurate, robust, and generalizable. Moreover, it has advantages for the inverse kinematics calculations of continuous trajectories.

Keywords: inverse kinematics; robotics; general manipulators; numerical solution; Newton's method; cyclic coordinate descent method; product-of-exponentials formula

1. Introduction

Solving typical robotics problems, such as trajectory planning [1] and motion control [2], requires that forward and inverse kinematics problems be addressed. The former involves calculating the position and orientation of a robot's end-effector frame from its joint values, which can easily be solved by using the matrix method of analysis [3]. The latter involves determining the joint variables corresponding to a given end-effector position and orientation. Indeed, the inverse kinematics problem is more complex than the forward kinematics problem because of the existence of nonlinear and multiple solutions [4]. Generally, two types of solutions exist: analytical solutions and numerical solutions. Iterative numerical methods can be applied if the inverse kinematics problem in question does not have any closed-form solutions and if it does not satisfy the Pieper criterion [5]. Even in cases where closed-form solutions do exist, iterative numerical methods are often used to improve the accuracy of these solutions.

From the 1980s, the inverse kinematics problem has attracted the attention of a large number of experts and researchers, all of whom have made remarkable achievements with respect to providing an adequate solution. Tsai and Morgan equated the kinematics problem of the general six-degree-of-freedom robot manipulators to that of a system of eight second-degree equations with eight unknowns solved numerically by polynomial continuation [6]. The authors in [6] conjectured that this problem has at most 16 solutions, the first conclusive proof of which was given in [7]. In order to improve the real-time performance of the solution, Manocha and Canny presented an algorithm for efficient inverse kinematics for a general six revolute (6R) robot manipulator, which performs both symbolic preprocessing and matrix computations and reduces the problem in question to computing the eigendecomposition of a matrix [8]. The generality of this algorithm, however, is limited since it can only be applied to 6R robot manipulators. In order to improve the generality of the inverse kinematics algorithm, some general approaches are proposed. The representative algorithms are the Newton–Raphson (NR) method [9] and the damped least-squares approach [10,11], which are all based on the inverse Jacobian matrix. The common drawback of this kind of algorithm is that it is sensitive to the initial solution. In order to solve this problem, Goldenberg et al. proposed a combined optimization method, which is based on a combination of two nonlinear programming techniques [12]. More specifically, the proposed method uses the cyclic coordinate descent (CCD) method [13] in order to rapidly find a feasible point approximate to the true solution and then the Broyden–Fletcher–Shanno variable metric method [13] in order to obtain a solution accurate to the desired degree of precision. Recently, some novel algorithms have been proposed to solve the inverse kinematics problem of robot manipulators. Kelemen et al. presented an algorithm for the control of kinematically-redundant manipulators considering three secondary tasks: a joint limit avoidance task, a kinematic singularities avoidance task, and an obstacle avoidance task [14]. Indeed, this approach is practical and results in a significant decrease in computing time. Zaplana and Basanez proposed an approach for a closed-form solution for the inverse kinematics of redundant manipulators [15]. This approach involves transforming redundant manipulators into non-redundant ones by selecting a set of joints, identifying the redundant ones, and parametrizing the joint variables. Thereafter, several closed-form methods previously developed for non-redundant manipulators can be applied in order to obtain the requisite solutions. This approach, however, is only suitable for manipulators that conform to the Pieper criterion. Qiao et al. [16] and Menini et al. [17] solved the inverse kinematics problem using double quaternions and Lie symmetries, respectively. In addition, other methods based on heuristic search techniques have been developed to solve the inverse kinematics problem, such as neural networks [18,19], genetic algorithms [20,21], and particle swarm optimization [22], as well as a combination of these methods [23,24].

On the basis of prior research, there is rarely a general and effective method for solving inverse kinematics. Therefore, this paper transforms the inverse kinematics problem into the objective function optimization problem, which is based on the least-squares form of the angle error and the position error expressed by the product-of-exponentials (PoE) formula. Then the new general approach that is suitable for the inverse kinematics of revolute and prismatic joint robots with any number of degrees of freedom as well as any configurations is presented.

The rest of the paper is organized in the following manner. Section 2 provides basic and associated knowledge, including the kinematics description of robot manipulators, Newton's method and the CCD method. Section 3 provides a definition of the objective function with respect to inverse kinematics and then the necessary formulas for Newton's method and the iterative procedure for the improved cyclic coordinate descent (ICCD) method. At the end of this part, the Newton-improved cyclic coordinate descent (NICCD) method is presented to solve the inverse kinematics problem. Section 4 provides the simulation results, followed by the discussion. Finally, Section 5 provides the conclusions and future work.

2. Preliminaries

Generally, this section introduces the preliminary knowledge used in this article. Firstly, the kinematics description of the robot manipulator is introduced, which plays an important role in deriving the formulas. The iteration process for Newton's method and the CCD method are then described, thereby laying the foundation for the proposed NICCD method.

2.1. Kinematics Description of Robot Manipulator

Generally speaking, two methods of describing the forward kinematics of open chains currently exist. The first method relies on the Denavit- Hartenberg parameters and the second relies on the PoE formula. Indeed, the latter method has advantages over the former (e.g., no link frames are necessary) [25], and it is, therefore, the preferred choice with respect to representing forward kinematics.

The PoE formula used to describe the forward kinematics $(\boldsymbol{T}(\boldsymbol{\Theta}), \boldsymbol{\Theta} \in \mathbb{R}^n)$ of an open chain with n-degrees of freedom follows Equation (1)

$$\boldsymbol{T}(\boldsymbol{\Theta}) = \begin{bmatrix} \boldsymbol{R}(\boldsymbol{\Theta}) & \boldsymbol{P}(\boldsymbol{\Theta}) \\ \boldsymbol{0} & 1 \end{bmatrix} = e^{S_1\theta_1} \cdots e^{S_{n-1}\theta_{n-1}} e^{S_n\theta_n} \boldsymbol{M}. \tag{1}$$

Specifically, Equation (1) is the space form of the PoE formula, referring to the fact that the screw axes are expressed in the fixed space frame, $\boldsymbol{R}(\boldsymbol{\Theta}) \in specialorthogonalgroup(SO(3))$, $\boldsymbol{P}(\boldsymbol{\Theta}) \in \mathbb{R}^{3\times1}$, $\boldsymbol{0}$ denotes the (1×3) null vector.

To calculate the homogeneous transformation matrics $(\boldsymbol{T}(\boldsymbol{\Theta}))$, which represent both the position and orientation of the end-effector with respect to the base frame, the elements outlined below are required.

(a) The end-effector configuration $\boldsymbol{M} \in$ Special Euclidean Group $(SE(3))$ is defined as

$$\boldsymbol{M} = \begin{bmatrix} \boldsymbol{R}_m & \boldsymbol{P}_m \\ \boldsymbol{0} & 1 \end{bmatrix}, \tag{2}$$

when the robot manipulator is at its home position.

(b) The screw axes are

$$\boldsymbol{s}_i = \begin{bmatrix} \boldsymbol{\omega}_i \\ \boldsymbol{v}_i \end{bmatrix} = [\omega_{i_1}, \omega_{i_2}, \omega_{i_3}, v_{i_1}, v_{i_2}, v_{i_3}]^T, \quad (i = 1 \cdots n), \tag{3}$$

where

$$\boldsymbol{v}_i = -\boldsymbol{\omega}_i \times \boldsymbol{q}_i \tag{4}$$

with $\boldsymbol{q}_i$ (any arbitrary point on the joint axis i) written in coordinates in the fixed base frame, corresponding to the joint motions when the robot manipulator is at the home position. Since the screw axis $\boldsymbol{s}_i$ is just a normalized twist, $\boldsymbol{S}_i$ represents Equation (5)

$$\boldsymbol{S}_i = \begin{bmatrix} \boldsymbol{\Omega}_i & \boldsymbol{v}_i \\ \boldsymbol{0} & 0 \end{bmatrix}, \tag{5}$$

where $\boldsymbol{\Omega}_i$ is a 3×3 skew-symmetric matrix representation of $\boldsymbol{\omega}_i$ and is defined as

$$\boldsymbol{\Omega_i} = \begin{bmatrix} 0 & -\omega_{i_3} & \omega_{i_2} \\ \omega_{i_3} & 0 & -\omega_{i_1} \\ -\omega_{i_2} & \omega_{i_1} & 0 \end{bmatrix}. \tag{6}$$

(c) The joint variables are represented as follows: $\boldsymbol{\Theta} = [\theta_1, \theta_2, \cdots, \theta_n]^T$.
(d) The matrix exponential $e^{S_i\theta_i}$ can be calculated using Equation (7)

$$e^{S_i\theta_i} = \begin{bmatrix} \boldsymbol{R_i}(\theta_i) & \boldsymbol{P_i}(\theta_i) \\ \boldsymbol{0} & 1 \end{bmatrix}, \tag{7}$$

where $\boldsymbol{R_i}(\theta_i)$ and $\boldsymbol{P_i}(\theta_i)$ are calculated as

$$\boldsymbol{R_i}(\theta_i) = e^{\boldsymbol{\Omega_i}\theta_i} = \boldsymbol{I} + \boldsymbol{\Omega_i} \sin\theta_i + \boldsymbol{\Omega_i^2}(1 - \cos\theta_i), \tag{8}$$

$$\boldsymbol{P_i}(\theta_i) = \Big((\boldsymbol{I} + \boldsymbol{\Omega_i^2})\theta_i + \boldsymbol{\Omega_i}(1 - \cos\theta_i) - \boldsymbol{\Omega_i^2} \sin\theta_i\Big) \boldsymbol{v_i}, \tag{9}$$

where $\boldsymbol{I}$ denotes the (3×3) identity matrix.

The formulas listed in this section will be used for calculating the formulas in the NICCD method.

2.2. Newton's Method

An effective way to solve nonlinear optimization problems is to use Newton's method. This method has quadratic convergence, and it uses both the first-order and second-order partial derivatives of the objective function, thus taking into account gradient changes. Therefore, it can comprehensively determine the appropriate search direction, and it has a more rapid convergence rate than the gradient method [26]. In particular, some scholars have improved the truncation criterion of Newton's method, which avoids "over-solving" of the Newton equation as much as possible. The representative method is the heuristic adaptive truncation criterion proposed by [27,28].

Although Newton's method requires a Hessian matrix, it is an appropriate method for solving low-dimensional optimization problems, such as the inverse kinematics of robot manipulators. The iterative formula for finding the local minimum value of the objective function ($f(\boldsymbol{\Theta}), \boldsymbol{\Theta} \in \mathbb{R}^n$) using Newton's method is given by Equation (10)

$$\boldsymbol{\Theta}^{(k+1)} = \boldsymbol{\Theta}^{(k)} - \mathrm{pinv}(\boldsymbol{H}(\boldsymbol{\Theta}^{(k)})) \bigtriangledown f(\boldsymbol{\Theta}^{(k)}), \tag{10}$$

where pinv() is the pseudoinverse, and where $\bigtriangledown f(\boldsymbol{\Theta}^{(k)})$ is the gradient of the k-th iteration of the objective function, which is defined as

$$\bigtriangledown f(\boldsymbol{\Theta}^{(k)}) = [\frac{\partial f(\boldsymbol{\Theta}^{(k)})}{\partial\theta_1}, \frac{\partial f(\boldsymbol{\Theta}^{(k)})}{\partial\theta_2}, \cdots, \frac{\partial f(\boldsymbol{\Theta}^{(k)})}{\partial\theta_n}]^T. \tag{11}$$

Moreover, $\boldsymbol{H}(\boldsymbol{\Theta}^{(k)})$ must be positive definite matrix for the Hessian matrix of the k-th iteration of the objective function, which is expressed as Equation (12)

$$\boldsymbol{H}(\boldsymbol{\Theta}^{(k)}) = \bigtriangledown^2 f(\boldsymbol{\Theta}^{(k)}) = \begin{bmatrix} \frac{\partial^2 f}{\partial\theta_1^2} & \frac{\partial^2 f}{\partial\theta_1\partial\theta_2} & \cdots & \frac{\partial^2 f}{\partial\theta_1\partial\theta_n} \\ \frac{\partial^2 f}{\partial\theta_2\partial\theta_1} & \frac{\partial^2 f}{\partial\theta_2^2} & \cdots & \frac{\partial^2 f}{\partial\theta_2\partial\theta_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial\theta_n\partial\theta_1} & \frac{\partial^2 f}{\partial\theta_n\partial\theta_2} & \cdots & \frac{\partial^2 f}{\partial\theta_n^2} \end{bmatrix}. \tag{12}$$

Newton's method for finding the optimal solution ($\boldsymbol{\Theta}^*$) consists of the step-by-step procedure outlined below.

Step 1: determine the gradient $\bigtriangledown f(\boldsymbol{\Theta}^{(k)})$ (see Equation (11)).

Step 2: verify convergence with the final solution as follows,

$$\left\| \bigtriangledown f(\boldsymbol{\Theta}^{(k)}) \right\| \leqslant \epsilon, \tag{13}$$

where ϵ is a threshold value supplied by the user. If Equation (13) holds, i.e., then $\boldsymbol{\Theta}^* = \boldsymbol{\Theta}^{(k)}$ is the optimal solution, then terminate the process; otherwise, proceed to the next step.

Step 3: determine the Hessian matrix (see Equation (12)) and $\boldsymbol{\Theta}^{k+1}$ (see Equation (10)). Then, let $k = k+1$ and return to Step 1.

2.3. The CCD Method

The CCD method, also known as the univariate search technique, is a dimension reduction method for unconstrained optimization problems. Its iteration process requires that one searches alternately along different coordinate directions. Indeed, the CCD method is a simpler method of establishing the search direction than the objective function derivative.

Although the convergence rate of the CCD method is slow for high-dimensional optimization problems, it is permissible with respect to the inverse kinematics problem of robot manipulators, and it can reduce the objective function at each iteration.

The CCD method process requires finding the optimal solution for one variable at a time while keeping the remaining $n-1$ variable unchanged. The last point of the previous one-dimensional search is the starting point of this one-dimensional search. The iterative formula for finding the local minimum value of the objective function ($f(\boldsymbol{\Theta}), \boldsymbol{\Theta} \in \mathbb{R}^n$) by the CCD method is Equation (14)

$$\boldsymbol{\Theta}_i^k = \boldsymbol{\Theta}_{i-1}^k + \alpha_i^{(k)} \boldsymbol{D}_{\boldsymbol{i}}^{(k)}, \qquad (i = 1, 2, \cdots, n), \tag{14}$$

where $\boldsymbol{D}_{\boldsymbol{i}}^{(k)}$ is the search direction, defined as

$$\boldsymbol{D}_{\boldsymbol{i}}^{(k)} = [0, \cdots, 1, \cdots, 0]^T, \tag{15}$$

in which the component of the i-th coordinate direction is 1 and the remaining components are 0, Moreover, $\alpha_i^{(k)}$ is the optimal step length in the i-th iteration of the k-th round, which is written as Equation (16)

$$\alpha_i^{(k)} = \arg\min_{\alpha_i^{(k)}} f(\boldsymbol{\Theta}_{i-1}^k + \alpha_i^{(k)} \boldsymbol{D}_{\boldsymbol{i}}^{(k)}). \tag{16}$$

The CCD method for finding the optimal solution ($\boldsymbol{\Theta}^*$) consists of the following step-by-step procedure.

Step 1: let i = 1.

Step 2: determine the optimal step length $\alpha_i^{(k)}$ (see Equation (16)) and $\boldsymbol{\Theta}_i^k$ (see Equation (14)).

Step 3: check whether the iteration round is over as follows:

$$i = n. \tag{17}$$

If Equation (17) holds, i.e., then proceed to the next step, else let $i = i+1$ and return to step 2.

Step 4: verify convergence with the final solution as follows,

$$\left\| \boldsymbol{\Theta}_n^{(k)} - \boldsymbol{\Theta}_0^{(k)} \right\| \leqslant \epsilon, \tag{18}$$

where ϵ is a threshold value supplied by the user. If Equation (18) holds, i.e., then $\boldsymbol{\Theta}^* = \boldsymbol{\Theta}_n^{(k)}$ is the optimal solution, then terminate the process; otherwise, let $\boldsymbol{D}_{\boldsymbol{i}}^{(k+1)} = \boldsymbol{D}_{\boldsymbol{i}}^{(k)} (i = 1, 2, \cdots, n), \boldsymbol{\Theta}_0^{(k+1)} = \boldsymbol{\Theta}_n^{(k)}, k = k+1$ and return to Step 1.

The CCD method does not require the derivative of the objective function in the search process; rather, it only requires the objective function value information. Moreover, it can be used to analytically find the optimal value in each search direction, such as the single joint variable optimization for the objective function proposed in this paper.

3. Numerical Methods for Nonlinear Kinematic Equations

In this section, the inverse kinematics problem is first transformed into the problem of how to get the value of $\mathbf{\Theta}^*$ in order to minimize the objective function, which is based on the least-squares form of the angle error and the position error expressed by the PoE formula. Accordingly, the equivalence of these two problems is proven. Then, the necessary formulas for Newton's method and the ICCD methods are given. Finally, for the NICCD method, the iterative procedure required to find the local minimum value of the objective function with respect to inverse kinematics is proposed.

3.1. Definition of Objective Function in Inverse Kinematics

For an open chain with n-degrees of freedom with respect to forward kinematics $(\boldsymbol{T}(\mathbf{\Theta}), \mathbf{\Theta} \in \mathbb{R}^n)$, the inverse kinematics problem can be stated as follows: Given a homogeneous transformation matrix, $\boldsymbol{E} \in SE(3)$ which is defined as follows

$$\boldsymbol{E} = \begin{bmatrix} \boldsymbol{R_e} & \boldsymbol{P_e} \\ \mathbf{0} & 1 \end{bmatrix}, \tag{19}$$

find solutions $(\mathbf{\Theta}^*)$ that satisfy

$$T(\mathbf{\Theta}^*) = \boldsymbol{E}. \tag{20}$$

As previously mentioned, this problem can be transformed into the problem of how to get the value of $\mathbf{\Theta}^*$ in order to minimize the objective function $(f(\mathbf{\Theta}))$ of inverse kinematics. Formally, the problem can be defined as Equation (21)

$$\mathbf{\Theta}^* = \arg\min_{\mathbf{\Theta}} f(\mathbf{\Theta}), \tag{21}$$

where the objective function $(f(\mathbf{\Theta}))$ of inverse kinematics is written as

$$\begin{aligned} f(\mathbf{\Theta}) &= \mathrm{RErr}(\mathbf{\Theta}) + \lambda \mathrm{PErr}(\mathbf{\Theta}) \\ &= \mathrm{trace}\left((\boldsymbol{R}(\mathbf{\Theta}) - \boldsymbol{R_e})^T(\boldsymbol{R}(\mathbf{\Theta}) - \boldsymbol{R_e})\right) + \lambda\left(\boldsymbol{P}(\mathbf{\Theta}) - \boldsymbol{P_e}\right)^T\left(\boldsymbol{P}(\mathbf{\Theta}) - \boldsymbol{P_e}\right), \end{aligned} \tag{22}$$

where $\lambda > 0$ is a scale factor, and $\boldsymbol{R}(\mathbf{\Theta})$ and $\boldsymbol{P}(\mathbf{\Theta})$ are expressed as

$$\boldsymbol{R}(\mathbf{\Theta}) = \prod_{i=1}^{n+1} \boldsymbol{R_i}(\theta_i), \tag{23}$$

$$\boldsymbol{P}(\mathbf{\Theta}) = \sum_{k=0}^{n} \left(\prod_{i=1}^{k} \boldsymbol{R_i}(\theta_i)\right) \boldsymbol{P_{k+1}}(\theta_{k+1}), \tag{24}$$

where $\boldsymbol{R_i}(\theta_i), \boldsymbol{P_i}(\theta_i)(i = 1, \cdots, n)$ can be calculated using Equations (8) and (9), respectively, and where $\boldsymbol{P_{n+1}}(\theta_{n+1}), \boldsymbol{R_{n+1}}(\theta_{n+1})$ are defined as

$$\boldsymbol{P_{n+1}}(\theta_{n+1}) = \boldsymbol{P_m}, \tag{25}$$

$$\boldsymbol{R_{n+1}}(\theta_{n+1}) = \boldsymbol{R_m}. \tag{26}$$

Equation (22) is based on the least-squares cost function. The first part of Equation (22) represents the sum of the squares of the errors of all elements of the current rotation matrix $(\boldsymbol{R}(\mathbf{\Theta}))$ and the target rotation matrix $(\boldsymbol{R_e})$. The second part of Equation (22) represents λ times the sum of the errors of all elements of the current displacement matrix $(\boldsymbol{P}(\boldsymbol{\Theta}))$ and the target displacement matrix $(\boldsymbol{P_e})$. Therefore, if a solution does not exist for inverse kinematics, then $\boldsymbol{\Theta}^*$ satisfies the required conditions as closely as possible in a least-squares sense. However, if a solution exists for inverse kinematics, then $\boldsymbol{\Theta}^*$ exactly satisfies

$$f(\boldsymbol{\Theta}^*) = \min f(\boldsymbol{\Theta}) = 0. \tag{27}$$

A constructive proof of this claim is provided below.

Proof. If a solution of inverse kinematics $(\boldsymbol{\Theta}^*)$ exists, then $\boldsymbol{R}(\boldsymbol{\Theta}^*) = \boldsymbol{R_e}$ and $\boldsymbol{P}(\boldsymbol{\Theta}^*) = \boldsymbol{P_e}$ must be satisfied. So Equation (27) must also be established. On the contrary, if there is $\boldsymbol{\Theta}^*$ so that Equation (27) holds, then $\boldsymbol{R}(\boldsymbol{\Theta}^*) = \boldsymbol{R_e}$ and $\boldsymbol{P}(\boldsymbol{\Theta}^*) = \boldsymbol{P_e}$ must also be satisfied. So Equation (20) must be established.

For the convenience of calculation, the objective function $(f(\boldsymbol{\Theta}), \boldsymbol{\Theta} \in \mathbb{R}^n)$ of inverse kinematics can also be defined as

$$\begin{aligned} f(\boldsymbol{\Theta}) &= \mathrm{RErr}(\boldsymbol{\Theta}) + \lambda \mathrm{PErr}(\boldsymbol{\Theta}) \\ &= \mathrm{trace}\left((\boldsymbol{R_i}(\theta_i)\boldsymbol{R_{ai}} - \boldsymbol{R_{ti}})^T (\boldsymbol{R_i}(\theta_i)\boldsymbol{R_{ai}} - \boldsymbol{R_{ti}}) \right) \\ &\quad + \lambda \left(\boldsymbol{R_i}(\theta_i)\boldsymbol{P_{ai}} + \boldsymbol{P_i}(\theta_i) - \boldsymbol{P_{ti}}\right)^T \left(\boldsymbol{R_i}(\theta_i)\boldsymbol{P_{ai}} + \boldsymbol{P_i}(\theta_i) - \boldsymbol{P_{ti}}\right), (i = 1, \cdots, n), \end{aligned} \tag{28}$$

where $\boldsymbol{R_{ai}}$, $\boldsymbol{P_{ai}}$, $\boldsymbol{R_{ti}}$ and $\boldsymbol{P_{ti}}$ are expressed as follows:

$$\boldsymbol{R_{ai}} = \prod_{j=i+1}^{n+1} \boldsymbol{R_j}(\theta_j), \tag{29}$$

$$\boldsymbol{P_{ai}} = \sum_{k=i}^{n} \left(\prod_{j=i+1}^{k} \boldsymbol{R_j}(\theta_j) \right) \boldsymbol{P_{k+1}}(\theta_{k+1}), \tag{30}$$

$$\boldsymbol{R_{ti}} = \left(\prod_{j=1}^{i-1} \boldsymbol{R_j}(\theta_j) \right)^{-1} \boldsymbol{R_e}, \tag{31}$$

$$\boldsymbol{P_{ti}} = \left(\prod_{j=1}^{i-1} \boldsymbol{R_j}(\theta_j) \right)^{-1} \left(\boldsymbol{P_e} - \sum_{k=0}^{i-2} \left(\prod_{j=1}^{k} \boldsymbol{R_j}(\theta_j) \right) \boldsymbol{P_{k+1}}(\theta_{k+1}) \right), \tag{32}$$

where $\boldsymbol{P_{n+1}}(\theta_{n+1})$ and $\boldsymbol{R_{n+1}}(\theta_{n+1})$ are defined as Equations (25) and (26), respectively. Equation (28) and Equation (22) are completely equivalent. □

3.2. *Necessary Formulas for Newton's Method*

Newton's method involves calculating the gradient and Hessian matrix of the object function, and its iterative formula for finding the local minimum value of the objective function (Equation (28)) is Equation (10). The important formulas for Newton's method are given below.

3.2.1. Determining the Gradient of Objective Function

For both prismatic and revolute joints, the gradient of the objective function of inverse kinematics is calculated using Equation (11), where

$$\frac{\partial f(\boldsymbol{\Theta})}{\partial \theta_i} = r_{i1}\cos(\theta_i) - r_{i2}\sin(\theta_i) + r_{i3}\theta_i + r_{i4}, \tag{33}$$

where

$$r_{i1} = 2\lambda \left((\boldsymbol{P}_{ti} - \boldsymbol{P}_{ai})^T \boldsymbol{\Omega}_i^2 \boldsymbol{v}_i - \boldsymbol{P}_{ti}^T \boldsymbol{\Omega}_i \boldsymbol{P}_{ai} \right) - 2\text{trace} \left(\boldsymbol{R}_{ti}^T \boldsymbol{\Omega}_i \boldsymbol{R}_{ai} \right), \tag{34}$$

$$r_{i2} = 2\lambda \left((\boldsymbol{P}_{ai} + \boldsymbol{P}_{ti})^T \boldsymbol{\Omega}_i \boldsymbol{v}_i - \boldsymbol{v}_i^T \boldsymbol{\Omega}_i^T \boldsymbol{\Omega}_i \boldsymbol{v}_i + \boldsymbol{P}_{ti}^T \boldsymbol{\Omega}_i^2 \boldsymbol{P}_{ai} \right) + 2\text{trace} \left(\boldsymbol{R}_{ti}^T \boldsymbol{\Omega}_i^2 \boldsymbol{R}_{ai} \right), \tag{35}$$

$$r_{i3} = 2\lambda \boldsymbol{v}_i^T (\boldsymbol{I} + \boldsymbol{\Omega}_i^2) \boldsymbol{v}_i, \tag{36}$$

$$r_{i4} = 2\lambda (\boldsymbol{P}_{ai} - \boldsymbol{P}_{ti})^T (\boldsymbol{I} + \boldsymbol{\Omega}_i^2) \boldsymbol{v}_i, \tag{37}$$

where $\boldsymbol{R}_{ai}$, $\boldsymbol{P}_{ai}$, $\boldsymbol{R}_{ti}$, and $\boldsymbol{P}_{ti}$ are defined as Equations (29), (30), (31), and (32), respectively.

If the robot manipulator joint is a prismatic joint ($\boldsymbol{\Omega}_i = \boldsymbol{0}$), then the elements in Equation (11) are simplified to

$$\frac{\partial f(\boldsymbol{\Theta})}{\partial \theta_i} = 2 \left(\theta_i + \lambda (\boldsymbol{P}_{ai} - \boldsymbol{P}_{ti})^T \boldsymbol{v}_i \right). \tag{38}$$

Conversely, if the robot manipulator joint is a revolute joint ($(\boldsymbol{I} + \boldsymbol{\Omega}_i^2)\boldsymbol{v}_i = \boldsymbol{0}$), then the elements in Equation (11) are simplified to

$$\frac{\partial f(\boldsymbol{\Theta})}{\partial \theta_i} = r_{i1} \cos(\theta_i) - r_{i2} \sin(\theta_i), \tag{39}$$

where r_{i1} and r_{i2} are defined as Equations (34) and (35), respectively.

3.2.2. Determining the Hessian Matrix of Objective Function

The Hessian matrix of the objective function of inverse kinematics defined by Equation (12) is a symmetric matrix, so

$$\frac{\partial^2 f}{\partial \theta_i \partial \theta_j} = \frac{\partial^2 f}{\partial \theta_j \partial \theta_i}. \tag{40}$$

On the one hand, if the robot manipulator's i-th joint is a prismatic joint, when $i = j$ is satisfied, then, according to Equation (38), the elements in the Hessian matrix can be defined as

$$\frac{\partial^2 f}{\partial \theta_i^2} = 2\lambda. \tag{41}$$

When $i < j$ is satisfied, the elements in the Hessian matrix can be calculated as

$$\frac{\partial^2 f}{\partial \theta_i \partial \theta_j} = 2\lambda \boldsymbol{v}_i^T \boldsymbol{h}_{ij1} \left(\boldsymbol{h}_{ij2} \boldsymbol{h}_{ij3} + \boldsymbol{h}_{ij4} \right), \tag{42}$$

where

$$\boldsymbol{h}_{ij1} = \prod_{k=i+1}^{j-1} \boldsymbol{R}_k, \tag{43}$$

$$\boldsymbol{h}_{ij2} = \boldsymbol{\Omega}_j \cos(\theta_j) + \boldsymbol{\Omega}_j^2 \sin(\theta_j), \tag{44}$$

$$\boldsymbol{h}_{ij3} = \sum_{e=j}^{n} \left(\prod_{k=j+1}^{e} \boldsymbol{R}_k \right) \boldsymbol{P}_{e+1}, \tag{45}$$

$$\boldsymbol{h}_{ij4} = \boldsymbol{\Omega}_j \boldsymbol{v}_j \sin(\theta_j) - \boldsymbol{\Omega}_j^2 \boldsymbol{v}_j \cos(\theta_j) + (\boldsymbol{I} + \boldsymbol{\Omega}_j^2) \boldsymbol{v}_j. \tag{46}$$

When $i > j$ is satisfied, the elements in the Hessian matrix can be calculated according to Equation (40).

On the other hand, if the robot manipulator's i-th joint is a revolute joint, when $i = j$ is satisfied, the elements in the Hessian matrix can be defined as

$$\frac{\partial^2 f}{\partial \theta_i^2} = -r_{i1}\sin(\theta_i) - r_{i2}\cos(\theta_i), \tag{47}$$

where r_{i1} and r_{i2} are defined as Equations (34) and (35), respectively. When $i < j$ is satisfied, the elements in the Hessian matrix can be expressed as

$$\begin{aligned}\frac{\partial^2 f}{\partial \theta_i \partial \theta_j} = &-2\lambda\left(\boldsymbol{v}_i^T \boldsymbol{d}_{ij1} + \boldsymbol{P}_{ti}^T \boldsymbol{d}_{ij4}\right)\boldsymbol{h}_{ij1}\left(\boldsymbol{d}_{ij2}\boldsymbol{h}_{ij3} + \boldsymbol{d}_{ij3}\right) \\ &- 2\mathrm{trace}\left(\boldsymbol{R}_{ti}^T \boldsymbol{d}_{ij4}\boldsymbol{h}_{ij1}\boldsymbol{\Omega}_j\boldsymbol{d}_{ij5}\right)\cos(\theta_j) - 2\mathrm{trace}\left(\boldsymbol{R}_{ti}^T \boldsymbol{d}_{ij4}\boldsymbol{h}_{ij1}\boldsymbol{\Omega}_j^2\boldsymbol{d}_{ij5}\right)\sin(\theta_j),\end{aligned} \tag{48}$$

where

$$\boldsymbol{d}_{ij1} = \boldsymbol{\Omega}_i^2\cos(\theta_i) - \boldsymbol{\Omega}_i\sin(\theta_i), \tag{49}$$

$$\boldsymbol{d}_{ij2} = \boldsymbol{\Omega}_j\cos(\theta_j) + \boldsymbol{\Omega}_j^2\sin(\theta_j), \tag{50}$$

$$\boldsymbol{d}_{ij3} = \boldsymbol{\Omega}_j\boldsymbol{v}_j\sin(\theta_j) - \boldsymbol{\Omega}_j^2\boldsymbol{v}_j\cos(\theta_j) + \left(\boldsymbol{I} + \boldsymbol{\Omega}_j^2\right)\boldsymbol{v}_j, \tag{51}$$

$$\boldsymbol{d}_{ij4} = \boldsymbol{\Omega}_i\cos(\theta_i) + \boldsymbol{\Omega}_i^2\sin(\theta_i), \tag{52}$$

$$\boldsymbol{d}_{ij5} = \prod_{k=j+1}^{n+1}\boldsymbol{R}_k. \tag{53}$$

When $i > j$ is satisfied, the elements in the Hessian matrix can be calculated according to Equation (40).

3.3. The ICCD Method

For the inverse kinematics problem, the CCD method has an analytical solution for the optimization of a single joint variable, but all joints are calculated independently, which makes the convergence slow. In order to improve this problem, this paper proposes the ICCD method in which multiple joint variables including consecutive prismatic joints and consecutive parallel revolute joints (Figure 1) can be adjusted concordantly. Although there are analytical solutions when the consecutive prismatic joints are perpendicular to each other, in order to improve the generality of the proposed method, the Newton's method and the CCD method are combined to solve these two special cases. Associated formulas and the iterative procedure for the ICCD method are given below.

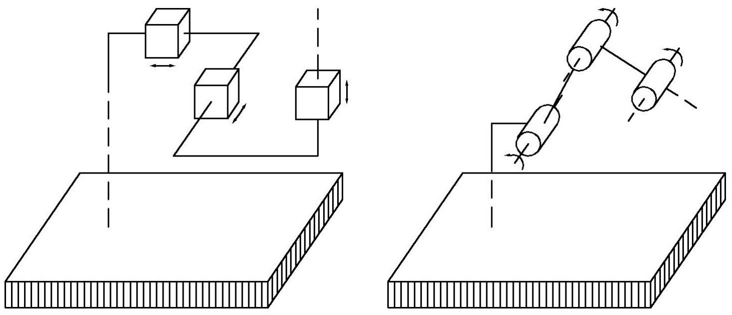

Figure 1. The consecutive prismatic joints and consecutive parallel revolute joints.

3.3.1. Necessary Formulas for Solving a Single Joint Variable

For a single joint variable, each iteration makes the objective function (Equation (28)) get a local minimum by adjusting only one joint variable, which has an analytical solution. If the joint is a prismatic joint, then the iteration formula for the ICCD method is calculated as

$$\theta_i^{(k+1)} = \lambda \left(\boldsymbol{P_{ti}}(\theta_i^{(k)}) - \boldsymbol{P_{ai}}(\theta_i^{(k)}) \right)^T \boldsymbol{v_i}, \tag{54}$$

where $\boldsymbol{P_{ti}}$ and $\boldsymbol{P_{ai}}$ are defined as Equations (32) and (30), respectively. If the joint is a revolute joint, then the iteration formula is defined as

$$\theta_i^{(k+1)} = \text{atan2}\left(-r_{i1}(\theta_i^{(k)}), -r_{i2}(\theta_i^{(k)}) \right), \tag{55}$$

where r_{i1} and r_{i2} are defined as Equation (34) and Equation (35), respectively.

The proof for the iteration formula outlined above is shown in Appendix A.

3.3.2. Necessary Formulas for Solving Consecutive Prismatic Joints

When there are consecutive prismatic joints in a robot manipulator, they are regarded as a whole. Moreover, these joint variables can be adjusted simultaneously in order to minimize the objective function of inverse kinematics. Assuming that the robot manipulator in question has consecutive prismatic joints from the i-th joint to the h-th joint, then the following can be achieved:

$$\boldsymbol{R_i} = \boldsymbol{R_{i+1}} = \cdots = \boldsymbol{R_h} = \boldsymbol{I}, \quad (1 \leqslant i < h \leqslant n). \tag{56}$$

According to Equations (28), the objective function of inverse kinematics can be simplified to

$$\begin{aligned} f(\boldsymbol{\Theta_{i\sim h}}) = \text{trace}\left((\boldsymbol{R_{cp}} - \boldsymbol{R_{ti}})^T (\boldsymbol{R_{cp}} - \boldsymbol{R_{ti}}) \right) + \sum_{k=i}^{h} \theta_k^2 + 2 \sum_{k=i}^{h-1} \sum_{j=k+1}^{h} \boldsymbol{v_k^T v_j} \theta_k \theta_j \\ + \lambda \left(2 (\boldsymbol{P_{cp}} - \boldsymbol{P_{ti}})^T \left(\sum_{k=i}^{h} \boldsymbol{v_k} \theta_k \right) + (\boldsymbol{P_{cp}} - \boldsymbol{P_{ti}})^T (\boldsymbol{P_{cp}} - \boldsymbol{P_{ti}}) \right), \end{aligned} \tag{57}$$

where consecutive prismatic joints ($\theta_i \cdots \theta_h$) are joint variables and other joints are constant, and where

$$\boldsymbol{\Theta_{i\sim h}} = [\theta_i, \cdots, \theta_h]^T, \tag{58}$$

$$\boldsymbol{R_{cp}} = \prod_{j=h+1}^{n+1} \boldsymbol{R_j}, \tag{59}$$

$$\boldsymbol{P_{cp}} = \sum_{k=h}^{n} \prod_{j=h+1}^{k} \boldsymbol{R_j P_{k+1}}, \tag{60}$$

$\boldsymbol{R_{ti}}$ and $\boldsymbol{P_{ti}}$ are defined as Equation (31) and Equation (32), respectively.

It's not hard to conclude that when the joints are perpendicular to each other ($\boldsymbol{v_k^T v_j} = 0$), there is an analytical solution. In order to show the generality of the method, including the case that the joints are not perpendicular to each other ($\boldsymbol{v_k^T v_j} \neq 0$), this paper uses the combination of Newton's method and the CCD method to solve this problem. But note that this numerical solution is the same as the analytical solution when the joint satisfies the mutually perpendicular condition ($\boldsymbol{v_k^T v_j} = 0$). The gradient, the Hessian matrix, and the iteration formula of the CCD method with respect to the objective function (Equation (57)) are given below.

The gradient of the objective function (Equation (57)) is defined as

$$\triangledown f(\boldsymbol{\Theta_{i\sim h}}) = \left[\frac{\partial f}{\partial \theta_i}, \cdots, \frac{\partial f}{\partial \theta_h} \right]^T, \tag{61}$$

where

$$\frac{\partial f}{\partial \theta_j} = 2\left(\theta_j + \boldsymbol{v_j^T} \left(\sum_{k=i,k\neq j}^{h} \boldsymbol{v_k} \theta_k \right) + \lambda (\boldsymbol{P_{cp}} - \boldsymbol{P_{ti}})^T \boldsymbol{v_j} \right), \quad (j = i, \cdots, h). \tag{62}$$

The Hessian matrix of the objective function (Equation (57)) is defined as

$$\boldsymbol{H}(\boldsymbol{\Theta_{i\sim h}}) = \bigtriangledown^2 f(\boldsymbol{\Theta_{i\sim h}}) = \begin{bmatrix} \frac{\partial^2 f}{\partial \theta_i^2} & \frac{\partial^2 f}{\partial \theta_i \partial \theta_{i+1}} & \cdots & \frac{\partial^2 f}{\partial \theta_i \partial \theta_h} \\ \frac{\partial^2 f}{\partial \theta_{i+1} \partial \theta_i} & \frac{\partial^2 f}{\partial \theta_{i+1}^2} & \cdots & \frac{\partial^2 f}{\partial \theta_{i+1} \partial \theta_h} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial \theta_h \partial \theta_i} & \frac{\partial^2 f}{\partial \theta_h \partial \theta_{i+1}} & \cdots & \frac{\partial^2 f}{\partial \theta_h^2} \end{bmatrix}, \tag{63}$$

where

$$\frac{\partial^2 f}{\partial \theta_j^2} = 2, \tag{64}$$

$$\frac{\partial^2 f}{\partial \theta_j \partial \theta_k} = 2\boldsymbol{v}_j^T \boldsymbol{v_k}, \quad (j \neq k). \tag{65}$$

It is not difficult to conclude that the Hessian matrix ($\boldsymbol{H}(\boldsymbol{\Theta_{i\sim h}})$) is always positive definite matrix, so when

$$\bigtriangledown f(\boldsymbol{\Theta_{i\sim h}}) = \boldsymbol{0} \tag{66}$$

is satisfied, the objective function (Equation (57)) obtains the local minimum value. Accordingly, the iteration formula of the CCD method of the objective function (Equation (57)) is calculated as

$$\theta_j^{(k+1)} = \left(\lambda \left(\boldsymbol{P_{ti}} - \boldsymbol{P_{cp}}\right) - \left(\sum_{l=i}^{j-1} \boldsymbol{v_l} \theta_l^{(k+1)} + \sum_{l=j+1}^{h} \boldsymbol{v_l} \theta_l^{(k)}\right)\right)^T \boldsymbol{v_j}, \quad (j = i, \cdots, h). \tag{67}$$

3.3.3. Necessary Formulas for Solving Consecutive Parallel Revolute Joints

When there are consecutive parallel revolute joints in a robot manipulator, they can be regarded as a whole. Moreover, these joint variables can be adjusted simultaneously in order to minimize the objective function of inverse kinematics. Assuming that the robot manipulator in question has consecutive parallel revolute joints from the i-th joint to the h-th joint,

$$\boldsymbol{\Omega_i} = \boldsymbol{\Omega_{i+1}} = \cdots = \boldsymbol{\Omega_h}, \quad (1 \leqslant i < h \leqslant n), \tag{68}$$

then the rotation matrix satisfies the following property:

$$\boldsymbol{R_i R_{i+1}} \cdots \boldsymbol{R_h} = e^{\boldsymbol{\Omega_i}\theta_i} e^{\boldsymbol{\Omega_{i+1}}\theta_{i+1}} \cdots e^{\boldsymbol{\Omega_h}\theta_h} = e^{\boldsymbol{\Omega_i}(\theta_i + \theta_{i+1} + \cdots + \theta_h)}. \tag{69}$$

When there are several consecutive parallel revolute joints $(\theta_i, \cdots, \theta_h)$ in the robot manipulator configuration, the several joints are regarded as variables, and the others as constants. Accordingly, the gradient of the objective function (Equation (28)) is defined as Equation (61) The Hessian matrix of the objective function(Equation (28)) is also defined as Equation (63).

In order to simplify the calculation, only the main diagonal elements of the Hessian matrix ($\boldsymbol{H}(\boldsymbol{\Theta_{i\sim h}})$) are considered here. When

$$\frac{\partial^2 f}{\partial \theta_i^2} \geqslant 0, \quad \frac{\partial^2 f}{\partial \theta_{i+1}^2} \geqslant 0, \quad \cdots, \quad \frac{\partial^2 f}{\partial \theta_h^2} \geqslant 0 \tag{70}$$

and

$$\frac{\partial f}{\partial \theta_i} = \frac{\partial f}{\partial \theta_{i+1}} = \cdots = \frac{\partial f}{\partial \theta_h} = 0 \tag{71}$$

are satisfied at the same time, the objective function (Equation (28)) obtains the local minimum. Thereafter, the iterative formulas of the CCD method for joint angles $(\boldsymbol{\Theta}_{i\sim h})$ can be calculated as

$$\theta_j^{(k+1)} = \operatorname{atan2}\left(g\left(\boldsymbol{\Theta}_{i\sim h}^{(k_{j-i+1})}\right), h\left(\boldsymbol{\Theta}_{i\sim h}^{(k_{j-i+1})}\right)\right), (j = i, i+1, \cdots, h), \tag{72}$$

where $g\,()$ and $h\,()$ are functions of $\boldsymbol{\Theta}_{i\sim h}^{(k_t)}$ which means that the t-th calculation in the k-round iteration, and where $\boldsymbol{\Theta}_{i\sim h}^{(k_{j-i+1})}$ are defined as

$$\boldsymbol{\Theta}_{i\sim h}^{(k_{j-i+1})} = [\theta_i^{(k+1)}, \cdots, \theta_{j-1}^{(k+1)}, \theta_j^{(k)}, \cdots, \theta_h^{(k)}]^T. \tag{73}$$

The configuration of more than three consecutive parallel revolute joints is rare in robot manipulators, Accordingly, necessary formulas with respect to minimizing the objective function by adjusting two or three consecutive parallel revolute joints at the same time are given in Appendix B.

3.3.4. ICCD Method of the Iterative Procedure

The ICCD method proposed in this section is responsible for finding $\boldsymbol{\Theta}^{(k+1)}$ from $\boldsymbol{\Theta}^{(k)}$ as shown in Algorithm 1. In line 1 groups are determined to facilitate subsequent computation. Specifically, consider a single revolute joint, a single prismatic joint, consecutive parallel revolute joints and consecutive prismatic joints as the groups, e.g., the UR robot with three consecutive parallel revolute joints$(\theta_2, \theta_3, \theta_4)$ is grouped into $\boldsymbol{G} = \{\boldsymbol{g_1}, \boldsymbol{g_2}, \boldsymbol{g_3}, \boldsymbol{g_4}\} = \{\{\theta_1\}, \{\theta_2, \theta_3, \theta_4\}, \{\theta_5\}, \{\theta_6\}\}$. In lines $3 \sim 44$, taking g_i as a variable and all others as constants, the goal is to find g_i such that the objective function obtains a local minimum value. In lines $3 \sim 6$, the single revolute joint and prismatic joint are calculated separately. In lines $8 \sim 44$ consecutive parallel revolute joints and consecutive prismatic joints are calculated in order to find the minimum value of the objective function. ϵ_{ca} is a threshold value to verify convergence supplied by the user. If the Hessian matrix $(\boldsymbol{H}(\boldsymbol{\Theta}_{i\sim h}))$ of either the consecutive parallel revolute joints or the consecutive prismatic joints is positive definite matrix, then Newton's method should be used (let $\text{flag}_{ca2} = 1$). Otherwise, Newton's method and the CCD method are used simultaneously, and the more effective result of the two methods is adopted (let $\text{flag}_{ca2} = 2$). The variable flag_{ca2} is a flag used to record which method is used in this iteration. If $\text{flag}_{ca2} = 1$, it means the former. On the contrary, if $\text{flag}_{ca2} = 2$, it means the latter. In order to ensure that the Newton's method and the CCD method converge to the local minimum value at the same time, $\text{flag}_{ca1} = 0$ is used, so that Newton's method and CCD method are simultaneously calculated in the case where the Hessian matrix is positive definite matrix. To be specific, the ICCD method consists of the step-by-step procedure shown in Algorithm 1.

3.4. The NICCD Method for The Inverse Kinematics Problem

The CCD method has great robustness and simple calculation process [13], but the convergence rate is slow when the iteration point approaches the optimal point. The ICCD method has the same problem. Conversely, Newton's method (with a quadratic rate of convergence) converges quickly when the iteration point approaches the optimal point. However, Newton's method often fails to converge when the initial estimate is not sufficiently close, and cannot search the descent direction when the second-order Hessian matrix is not positive definite matrix. In order to inherit the advantages of the two algorithms and discard their disadvantages, the NICCD method is proposed to solve the inverse kinematics problem. The determination of the scale factor and the iterative procedure for the NICCD method are given below.

Algorithm 1 The improved cyclic coordinate descent (ICCD) method.

```
 1: Initialization: Set $\mathrm{Err}_g = +\infty$, $\mathrm{flag}_{ca2} = 0$, and determine groups $\boldsymbol{G}^{(k)} = \left\{\boldsymbol{g}_1^{(k)}, \boldsymbol{g}_2^{(k)}, \cdots, \boldsymbol{g}_l^{(k)}\right\}$
 2: for each $i \in [1, l]$ do
 3:   if $\boldsymbol{g}_i^{(k)}$ is a group of single revolute joint then
 4:     Determine $\boldsymbol{g}_i^{(k+1)}$ (see Equation (55))
 5:   else if $\boldsymbol{g}_i^{(k)}$ is a group of single prismatic joint then
 6:     Determine $\boldsymbol{g}_i^{(k+1)}$ (see Equation (54))
 7:   else
 8:     Set $\mathrm{flag}_{ca1} = 1$
 9:     while $\mathrm{Err}_g > \epsilon_{ca}$ or $\mathrm{flag}_{ca2} \neq 2$ do
10:       if $\boldsymbol{g}_i^{(k)}$ is a group of consecutive parallel revolute joints then
11:         Calculate $\bigtriangledown f(\boldsymbol{g}_i^{(k)})$ and $\boldsymbol{H}(\boldsymbol{g}_i^{(k)})$ (see Equations (61), (63), and Appendix B)
12:       else
13:         Calculate $\bigtriangledown f(\boldsymbol{g}_i^{(k)})$ and $\boldsymbol{H}(\boldsymbol{g}_i^{(k)})$ (see Equations (61 $\sim$ 65))
14:       end if
15:       if $\boldsymbol{H}(\boldsymbol{g}_i^{(k)})$ is a positive definite matrix and $\mathrm{flag}_{ca1} = 1$ then
16:         Set $\mathrm{flag}_{ca2} = 1$
17:         Determine $\boldsymbol{g}_i^{(k+1)}$ by Newton's method (see Equation (10))
18:       else
19:         Set $\mathrm{flag}_{ca2} = 2$
20:         Determine $\boldsymbol{g}_{Ni}^{(k+1)}$ by Newton's method (see Equation (10))
21:         if $\boldsymbol{g}_i^{(k)}$ is a group of consecutive parallel revolute joints then
22:           Calculate $\boldsymbol{g}_{Ci}^{(k+1)}$ by CCD method (see Equation (72) and Appendix B)
23:         else
24:           Calculate $\boldsymbol{g}_{Ci}^{(k+1)}$ by CCD method (see Equation (67))
25:         end if
26:         Set $\boldsymbol{G}_C^{(k+1)} = \left\{\boldsymbol{g}_1^{(k+1)}, \cdots, \boldsymbol{g}_{Ci}^{(k+1)}, \boldsymbol{g}_{i+1}^{(k)}, \cdots, \boldsymbol{g}_l^{(k)}\right\}$ and $\boldsymbol{G}_N^{(k+1)} = \left\{\boldsymbol{g}_1^{(k+1)}, \cdots, \boldsymbol{g}_{Ni}^{(k+1)}, \boldsymbol{g}_{i+1}^{(k)}, \cdots, \boldsymbol{g}_l^{(k)}\right\}$
27:         Calculate $f_{caC} = f(\boldsymbol{G}_C^{(k+1)})$ and $f_{caN} = f(\boldsymbol{G}_N^{(k+1)})$ (see Equation (28))
28:         if $f_{caC} < f_{caN}$ then
29:           Set $\boldsymbol{g}_i^{(k+1)} = \boldsymbol{g}_{Ci}^{(k+1)}$
30:         else
31:           Set $\boldsymbol{g}_i^{(k+1)} = \boldsymbol{g}_{Ni}^{(k+1)}$
32:         end if
33:       end if
34:       Set $\mathrm{Err}_g = \left\|\boldsymbol{g}_i^{(k+1)} - \boldsymbol{g}_i^{(k)}\right\|$
35:       if $\mathrm{Err}_g \leqslant \epsilon_{ca}$ then
36:         if $\mathrm{flag}_{ca2} \neq 2$ then
37:           Set $\mathrm{flag}_{ca1} = 0$
38:         end if
39:       else
40:         Set $\mathrm{flag}_{ca1} = 1$
41:       end if
42:       Set $\boldsymbol{g}_i^{(k)} = \boldsymbol{g}_i^{(k+1)}$
43:     end while
44:   end if
45: end for
46: Set $\boldsymbol{\Theta}^{(k+1)} = \boldsymbol{G}^{(k+1)}$
```

3.4.1. The Scale Factor

There is a scale factor (λ) in the objective function (Equation (22)), which is used to scale the position error so that the difference between the position error and the angle error is not very large. This scale factor has an impact on the convergence speed of the proposed approach and the ability to avoid the local minimum.

From the demonstration in Appendix C, it can be concluded that

$$\max_{\mathbf{\Theta}} \mathrm{RErr}(\mathbf{\Theta}) = \max_{\mathbf{\Theta}} \mathrm{trace}\left((\boldsymbol{R}(\mathbf{\Theta}) - \boldsymbol{R}_e)^T (\boldsymbol{R}(\mathbf{\Theta}) - \boldsymbol{R}_e) \right) = 8. \tag{74}$$

Therefore, the scale factor (λ) can be defined as

$$\lambda = \frac{\max_{\mathbf{\Theta}} \mathrm{RErr}(\mathbf{\Theta})}{\max_{\mathbf{\Theta}} \mathrm{PErr}(\mathbf{\Theta})} = \frac{8}{\max_{\mathbf{\Theta}} \mathrm{PErr}(\mathbf{\Theta})}, \tag{75}$$

where the $\mathrm{PErr}(\mathbf{\Theta})$ is defined as Equation (22) and $\max_{\mathbf{\Theta}} \mathrm{PErr}(\mathbf{\Theta})$ can be calculated by the robot manipulator configuration in advance.

3.4.2. The NICCD Approach of the Iterative Procedure

The method proposed in this section seeks a solution $\mathbf{\Theta}^*$ to the problem defined by Equation (27) as shown in Algorithm 2. Some important parameters in Algorithm 2 are introduced as follows. Parameters t and $\max_k$ represent the coefficient of the scale factor (λ) and the maximum number of iterations respectively. Parameters ϵ_1 and ϵ_2 represent the convergence threshold of objective function (f) and independent variable ($\mathbf{\Theta}$) respectively. The roles of flag_1 and flag_2 can be analogized to flag_{ca1} and flag_{ca2} in Algorithm 1, respectively. In lines $3 \sim 17$, if the Hessian matrix is positive definite matrix and $\mathrm{flag}_1 = 1$, then $\mathbf{\Theta}^{(k+1)}$ can be calculated using Newton's method (let $\mathrm{flag}_2 = 1$); otherwise, Newton's method and the ICCD method are executed simultaneously, with the more efficient method being used to calculate $\mathbf{\Theta}^{(k+1)}$ (let $\mathrm{flag}_2 = 2$). Lines 18 and 19 involve the global and local convergence judgments of the method, respectively. In lines $23 \sim 30$, the method of jumping out of the local optimal solution is used. The NICCD method consists of the step-by-step procedure shown in Algorithm 2.

4. Simulation and Discussion

In order to illustrate the effectiveness, accuracy, robustness, and generality of the inverse kinematics approach proposed in this article, five simulations of six robots—a three-link planar arm (3R), a selective compliance assembly robot arm (SCARA), a Cartesian manipulator (3P), Universal Robot's UR5, a Stanford Arm, and Barrett Technology's WAM7R robot manipulator—are conducted using the NICCD method, the results of which are compared with the NR method using the Jacobian matrix. The NR method is described in section 6.2.2 of [25]. The PoE parameters of the six robots with the current configurations relative to the base frame are shown in Tables 1 and 2. The last simulation verifies the improvement of ICCD compared with CCD method and the evolution process of NICCD method. All simulations are performed on a desktop computer (Core i5 3.40GHz, 16GB RAM, MATLAB 2015b software program).

Algorithm 2 The Newton-improved cyclic coordinate descent (NICCD) method.

```
1: Initialization: Set $t = 1$, $\text{flag}_1 = 1$ and determine the scale factor $\lambda$ (see Equation (75))
2: for each $k \in [1, \max_k]$ do
3:     Calculate $\nabla f(\boldsymbol{\Theta}^{(k)})$ and $\boldsymbol{H}(\boldsymbol{\Theta}^{(k)})$ (see Equations (11), (12), and (33)~(39))
4:     if $\boldsymbol{H}(\boldsymbol{\Theta}^{(k)})$ is positive definite matrix and $\text{flag}_1 = 1$ then
5:         Set $\text{flag}_2 = 1$
6:         Determine $\boldsymbol{\Theta}^{(k+1)}$ by Newton's method (see Equation (10))
7:     else
8:         Set $\text{flag}_2 = 2$
9:         Determine $\boldsymbol{\Theta}_N^{(k+1)}$ by Newton's method (see Equation (10))
10:        Determine $\boldsymbol{\Theta}_C^{(k+1)}$ by ICCD method (see Algorithm 1 )
11:        Calculate $f_N = f(\boldsymbol{\Theta}_N^{(k+1)})$ and $f_C = f(\boldsymbol{\Theta}_C^{(k+1)})$ (see Equation (28))
12:        if $f_C < f_N$ then
13:            Set $\boldsymbol{\Theta}^{(k+1)} = \boldsymbol{\Theta}_C^{(k+1)}$
14:        else
15:            $\boldsymbol{\Theta}^{(k+1)} = \boldsymbol{\Theta}_N^{(k+1)}$
16:        end if
17:    end if
18:    if $\left\| f(\boldsymbol{\Theta}^{(k+1)}) \right\| > \epsilon_1$ then
19:        if $\left\| \boldsymbol{\Theta}^{(k+1)} - \boldsymbol{\Theta}^{(k)} \right\| \leqslant \epsilon_2$ then
20:            if $\text{flag}_2 \neq 2$ then
21:                Set $t = 1, \text{flag}_1 = 0$
22:            else
23:                if $\text{RErr}(\boldsymbol{\Theta}^{(k+1)}) < \text{PErr}(\boldsymbol{\Theta}^{(k+1)})$ then
24:                    Set $t = \frac{t}{2}$
25:                else
26:                    Set $t = 2t$
27:                end if
28:                Set $\lambda = t\lambda$
29:                Determine $\boldsymbol{\Theta}^{(k+1)}$ by ICCD method (see Algorithm 1 )
30:                Set $\lambda = \frac{\lambda}{t}$
31:            end if
32:        else
33:            Set $\text{flag}_1 = 1$
34:        end if
35:    else
36:        Set $\boldsymbol{\Theta}^* = \boldsymbol{\Theta}^{(k+1)}$
37:        break
38:    end if
39: end for
```

Table 1. The product-of-exponentials (PoE) parameters of three-link planar arm (3R), a selective compliance assembly robot arm (SCARA), a Cartesian manipulator (3P).

i	**3R**		**SCARA**		**3P**	
	ω_i^T	v_i^T	ω_i^T	v_i^T	ω_i^T	v_i^T
1	$[0,0,1]$	$[0,0,0]$	$[0,0,1]$	$[0,0,0]$	$[0,0,0]$	$[0,0,1]$
2	$[0,0,1]$	$[0,-400,0]$	$[0,0,1]$	$[0,-450,0]$	$[0,0,0]$	$[0,1,0]$
3	$[0,0,1]$	$[0,-800,0]$	$[0,0,0]$	$[0,0,-1]$	$[0,0,0]$	$[1,0,0]$
4			$[0,0,1]$	$[0,-850,0]$		
M	$\begin{bmatrix}1&0&0&1200\\0&1&0&0\\0&0&1&1\\0&0&0&1\end{bmatrix}$		$\begin{bmatrix}1&0&0&850\\0&1&0&0\\0&0&1&-420\\0&0&0&1\end{bmatrix}$		$\begin{bmatrix}1&0&0&400\\0&1&0&400\\0&0&1&400\\0&0&0&1\end{bmatrix}$	

Table 2. The PoE parameters of UR5, Stanford Arm, and WAM7R.

i	**UR5**		**Stanford Arm**		**WAM7R**	
	ω_i^T	v_i^T	ω_i^T	v_i^T	ω_i^T	v_i^T
1	$[0,0,1]$	$[0,0,0]$	$[0,0,1]$	$[0,0,0]$	$[0,0,1]$	$[0,0,0]$
2	$[0,1,0]$	$[-89,0,0]$	$[1,0,0]$	$[0,0,0]$	$[0,1,0]$	$[0,0,0]$
3	$[0,1,0]$	$[-89,0,425]$	$[0,0,0]$	$[0,0,1]$	$[0,0,1]$	$[0,0,0]$
4	$[0,1,0]$	$[-89,0,817]$	$[0,0,1]$	$[0,-200,0]$	$[0,1,0]$	$[-550,0,45]$
5	$[0,0,-1]$	$[-109,817,0]$	$[0,1,0]$	$[0,0,200]$	$[0,0,1]$	$[0,0,0]$
6	$[0,1,0]$	$[6,0,817]$	$[1,0,0]$	$[0,400,0]$	$[0,1,0]$	$[-850,0,0]$
7					$[0,0,1]$	$[0,0,0]$
M	$\begin{bmatrix}-1&0&0&817\\0&0&1&191\\0&1&0&-6\\0&0&0&1\end{bmatrix}$		$\begin{bmatrix}1&0&0&692\\0&1&0&0\\0&0&1&400\\0&0&0&1\end{bmatrix}$		$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&910\\0&0&0&1\end{bmatrix}$	

4.1. Simulation I

This simulation aims to verify the effectiveness of the proposed approach using SCARA and UR5. The specific values of the target joint variables ($\boldsymbol{\Theta}_{\mathbf{goal}}$) and the initial joint variables ($\boldsymbol{\Theta}_{\mathbf{in1}}, \boldsymbol{\Theta}_{\mathbf{in2}}$) with respect to SCARA and UR5 are shown in Table 3. According to Equation (1), the goal configurations of the end-effector frames of the two robot manipulators relative to $\boldsymbol{\Theta}_{\mathbf{goal}}$ are described as

$$\begin{aligned} T_{\mathbf{goal}} &= \begin{bmatrix}-0.8479 & -0.5301 & 0 & 160.1408\\ 0.5301 & -0.8479 & 0 & 383.1681\\ 0 & 0 & 1.0000 & -520.0000\\ 0 & 0 & 0 & 1.0000\end{bmatrix}, \text{(SCARA)},\\ T_{\mathbf{goal}} &= \begin{bmatrix}-0.9592 & -0.0838 & 0.2699 & -93.1191\\ 0.2823 & -0.3247 & 0.9027 & -20.4293\\ 0.0120 & 0.9421 & 0.3351 & -716.5883\\ 0 & 0 & 0 & 1.0000\end{bmatrix}, \text{(UR5)}. \end{aligned} \tag{76}$$

Table 3. The specific values of the target and initial joint variables of SCARA and UR5.

	SCARA				**UR5**					
	θ_1**(rad)**	θ_2**(rad)**	d_3**(mm)**	θ_4**(rad)**	θ_1**(rad)**	θ_2**(rad)**	θ_3**(rad)**	θ_4**(rad)**	θ_5**(rad)**	θ_6**(rad)**
$\boldsymbol{\Theta}_{\mathbf{goal}}$	0.2169	2.1269	100.0000	0.2391	3.0076	1.3364	0.0030	−0.1817	−2.7670	1.1434
$\boldsymbol{\Theta}_{\mathbf{in1}}$	−2.1142	2.6458	−11.9929	0.4863	−1.9350	−2.2690	1.2332	−2.5521	0.1596	0.1907
$\boldsymbol{\Theta}_{\mathbf{in2}}$	−1.5218	−0.6484	−20.0000	1.1567	−1.0585	0.4914	2.2274	0.8946	−0.5020	0.2885

The inverse kinematics are calculated using the NICCD and the NR method, with $\mathbf{\Theta_{in1}}$ and $\mathbf{\Theta_{in2}}$ as initial values. In order to illustrate the effectiveness of the proposed method, the angle error, and position error are defined as

$$\begin{aligned} \mathrm{Err}_R &= \sqrt{\mathrm{trace}\left((\boldsymbol{R}_{\mathbf{goal}} - \boldsymbol{R}_{\mathbf{out}})^T(\boldsymbol{R}_{\mathbf{goal}} - \boldsymbol{R}_{\mathbf{out}})\right)}, \\ \mathrm{Err}_P &= \sqrt{(\boldsymbol{P}_{\mathbf{goal}} - \boldsymbol{P}_{\mathbf{out}})^T(\boldsymbol{P}_{\mathbf{goal}} - \boldsymbol{P}_{\mathbf{out}})}, \end{aligned} \tag{77}$$

where $\boldsymbol{R}_{\mathbf{goal}}$ and $\boldsymbol{P}_{\mathbf{goal}}$ are the elements in $\boldsymbol{T}_{\mathbf{goal}}$, and $\boldsymbol{R}_{\mathbf{out}}$ and $\boldsymbol{P}_{\mathbf{out}}$ are the elements in $\boldsymbol{T}_{\mathbf{out}}$, which are obtained by the forward kinematics calculated by either the NICCD or the NR method with respect to $\mathbf{\Theta_{out}}$. The position and angle errors of the initial selected joint angles of the SCARA and the UR5 versus the convergence times of the NICCD and the NR methods are illustrated in Figure 2.

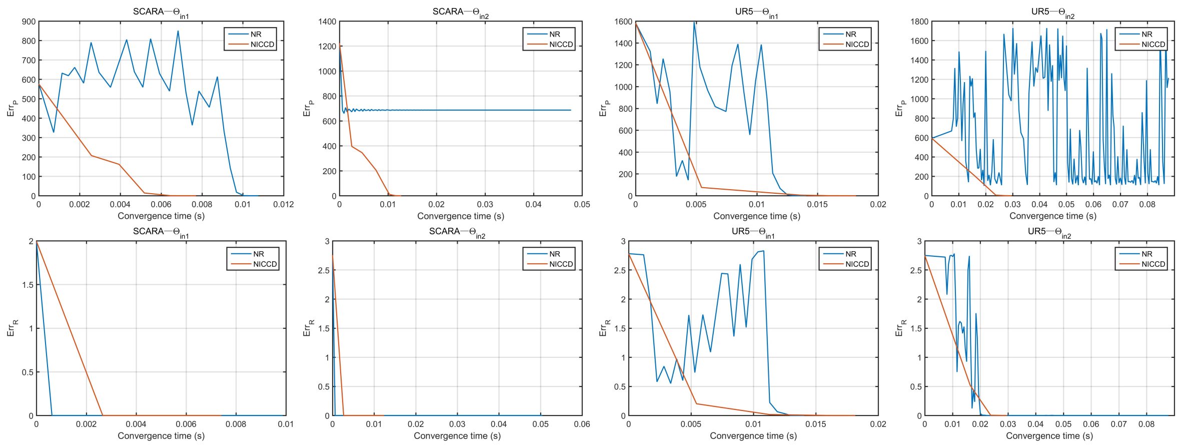

Figure 2. The position and angle errors of the initial selected joint angles of the SCARA and the UR5 versus the convergence times of the NICCD and the Newton–Raphson (NR) methods.

From Figure 2, when the initial value is $\mathbf{\Theta_{in1}}$, the NICCD method converges more rapidly than the NR method with respect to SCARA. However, it is slower with respect to UR5. When the initial value is $\mathbf{\Theta_{in2}}$, the NICCD method still converges, but the NR method diverges with respect to the inverse kinematics of SCARA and UR5. In addition, the NICCD method has less fluctuation in both the angle error and the position error during the iteration process. Therefore, it can be concluded that the proposed approach is more stable and effective than the NR method.

4.2. Simulation II

This second simulation involves testing the accuracy of the proposed approach using the Stanford Arm and Barrett Technology's WAM7R robot manipulator. The specific values of the target joint variables ($\mathbf{\Theta_{goal}}$) and the initial joint variables ($\mathbf{\Theta}_{\mathrm{in}}$) of the Stanford Arm and the WAM7R manipulator are shown in Table 4. According to Equation (1), the goal configurations of the end-effector frames of the two robot manipulators relative to $\mathbf{\Theta_{goal}}$ are described as

$$\begin{aligned} \boldsymbol{T}_{\mathbf{goal}} &= \begin{bmatrix} 0.6641 & -0.0505 & 0.7459 & 549.2836 \\ 0.6213 & -0.5177 & -0.5882 & -42.3986 \\ 0.4159 & 0.8541 & -0.3124 & 582.0788 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}, \text{(Stanford Arm)}, \\ \boldsymbol{T}_{\mathbf{goal}} &= \begin{bmatrix} -0.0765 & -0.9970 & -0.0139 & 346.4601 \\ 0.7721 & -0.0504 & -0.6335 & 309.6627 \\ 0.6309 & -0.0592 & 0.7736 & -381.6887 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}, \text{(WAM7R)}. \end{aligned} \tag{78}$$

In the case of closing the convergence judgment, 150 iterations of the NICCD and NR methods were conducted. The position and angle errors of the initial selected joint angles of the Stanford Arm and the WAM7R manipulator versus the number of iterations of the NICCD and NR methods are given in Figure 3, where the error is represented on the logarithmic axis. Indeed, from Figure 3, it is clear that the approach proposed in this paper is more accurate than the NR method with respect to the position error and angle error.

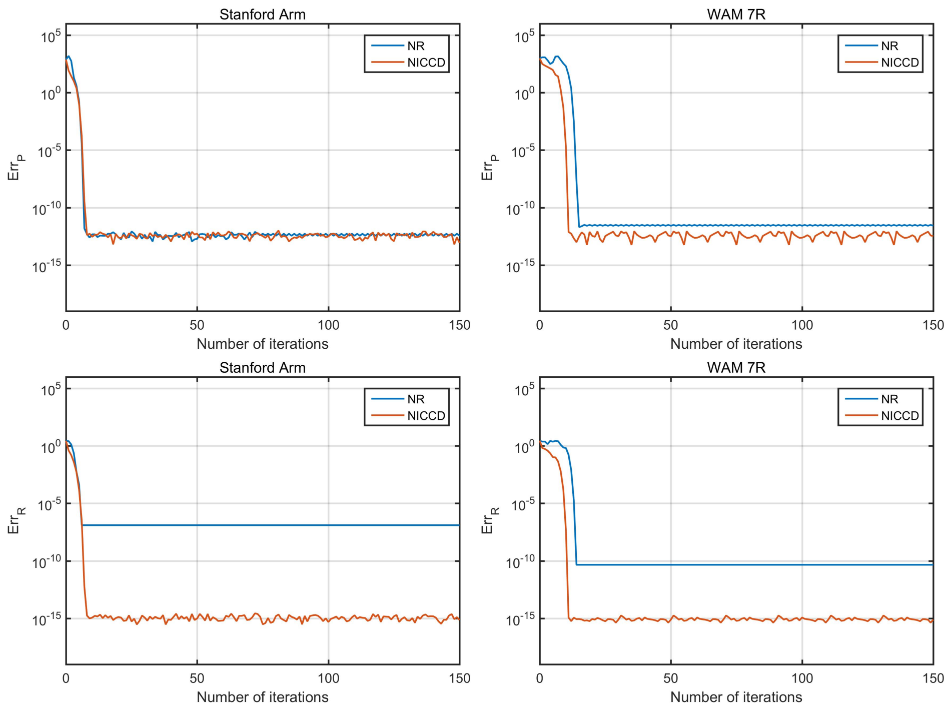

Figure 3. The position and angle errors of the initial selected joint angles of the Stanford Arm and the WAM7R manipulator versus the number of iterations of the NICCD and NR methods.

Table 4. The specific values of the target and initial joint variables of Stanford Arm and WAM7R manipulator.

	Stanford Arm						WAM7R						
	θ_1(rad)	θ_2(rad)	d_3(mm)	θ_4(rad)	θ_5(rad)	θ_6(rad)	θ_1(rad)	θ_2(rad)	θ_3(rad)	θ_4(rad)	θ_5(rad)	θ_6(rad)	θ_7(rad)
Θ_{goal}	−0.1290	0.8754	100.0000	0.9256	0.2757	1.3889	0.4088	2.0346	−2.3493	−1.2559	−3.1283	2.8344	1.6732
Θ_{in}	−1.0520	2.2184	−0.3619	2.5406	−2.9331	0.2037	−0.5576	2.2993	2.6431	1.8048	−0.9740	−2.7061	1.6552

4.3. Simulation III

This simulation is conducted in order to validate the robustness of the proposed approach by using the 3P, 3R, SCARA, UR5, and Barrett Technology's WAM7R robot manipulator. The difference between this simulation and the prior two is that 500 sets of target joint variables and 500 sets of initial joint variables are generated randomly. The end-effector configurations of the four robots are calculated by forward kinematics relative to the target joint variables. Thereafter, the inverse kinematics of the four robots are solved using the NICCD and NR method. The correct rate of solving the inverse kinematics of these four robots is shown in Figure 4. Indeed, from Figure 4, for the WAM7R robot manipulator, it is clear that the approach proposed in this paper has a correct rate that is smaller than that of the NR method. Moreover, from Figure 4, for the remaining four robots, it is clear that the approach proposed in this paper has a correct rate that is larger than that of the NR method.

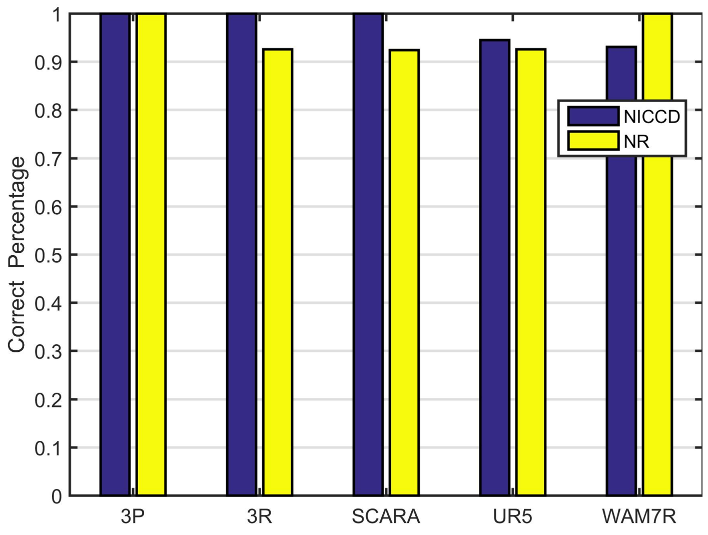

Figure 4. The correct rate of solving inverse kinematics of these four robots.

4.4. Simulation IV

This simulation aims to illustrate the fact that the proposed approach has offline programming abilities by using the UR5. The goal here is to create a trajectory that will allow the UR5 to draw the letter 'F' (the red trajectory in Figure 5) in 20 s. The orientation of the end-effector is constant, with the $\boldsymbol{y}$-axis parallel to that of the base frame and the $\boldsymbol{x}$- and $\boldsymbol{z}$-axes opposite to that of the base frame. Specifically, the desired trajectory of UR5 is shown in the left figure of Figure 5, and the actual trajectory is shown in the right figure. The latter is obtained after discrete and smooth processing of the former. The inverse kinematics solution for each sample point is calculated. and the position, velocity, and acceleration of the joints are given in Figure 6.

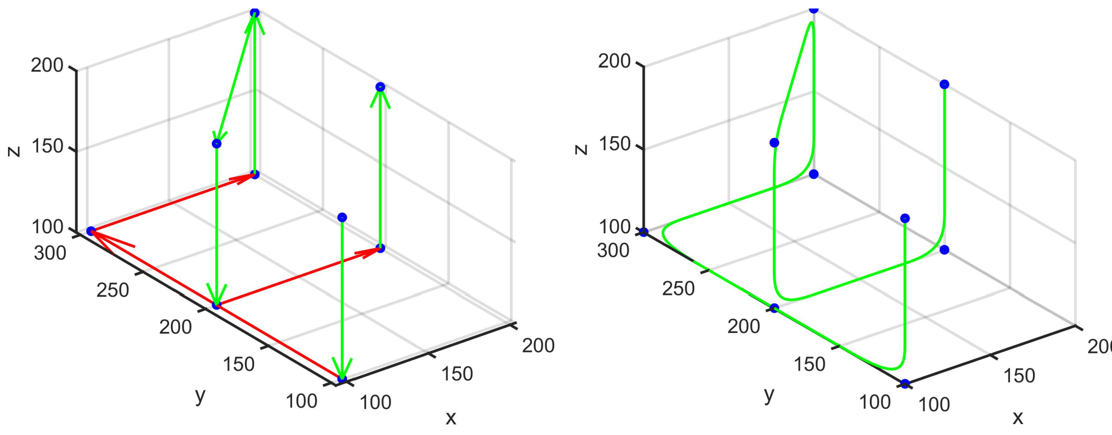

Figure 5. The UR5 motion trajectory.

The proposed approach takes an average of 0.0057 s with respect to the calculation process, which demonstrates how efficient it is. By comparing Figure 6 of Simulation IV and Figure 2 of Simulation I, the average calculation time of the former is shorter than the latter. This is because Simulation IV follows a continuous trajectory, which means that the solution of the last step is taken as the initial value of the next step. In other words, the difference between the initial joint variables and the target joint variables is smaller in Simulation IV than in Simulation I. Accordingly, it can be concluded that the proposed approach is appropriate for offline programming and that it is advantageous with respect to the inverse kinematics of continuous trajectories.

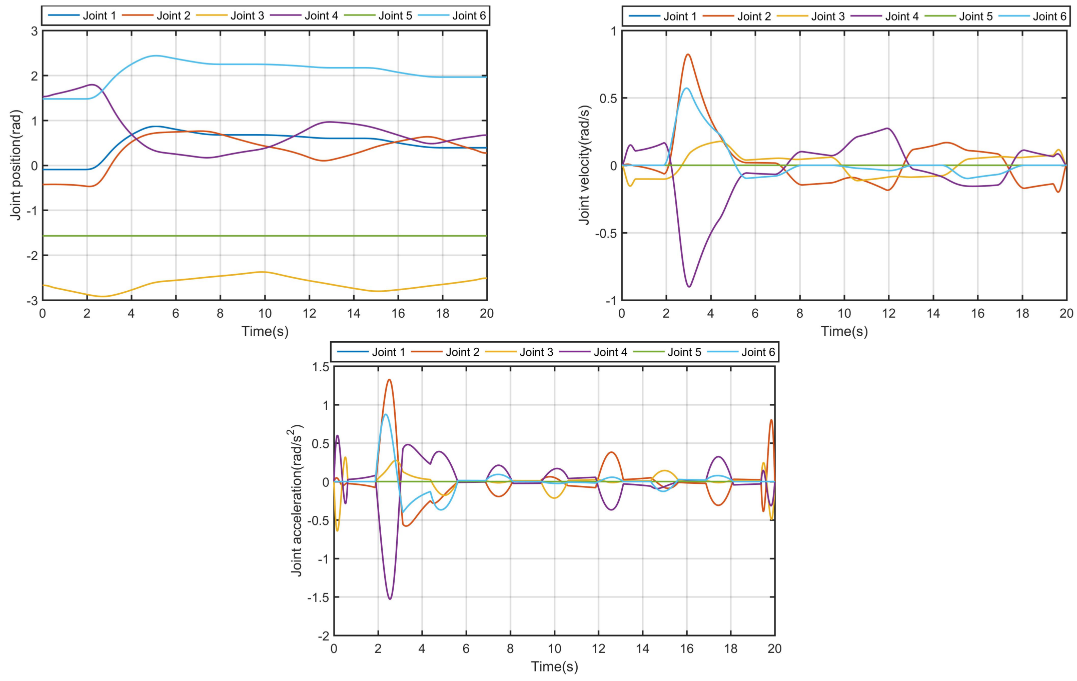

Figure 6. The position, velocity, and acceleration of joints.

4.5. Simulation V

This simulation aims to verify the improvement of ICCD method (consider consecutive prismatic joints or consecutive parallel revolute joints as the group) compared with the traditional CCD method (each joint is adjusted individually) and the evolution process of NICCD method. UR5 manipulator with consecutive parallel revolute joints meets the configuration requirements. The specific values of its target and initial angles of UR5 are the same as the data of $\mathbf{\Theta_{goal}}$ and $\mathbf{\Theta_{in2}}$ in Simulation I (see Table 3). According to Equation (1), the goal configuration of the end-effector frame of UR5 relative to $\mathbf{\Theta_{goal}}$ is described as Equation (76).

The inverse kinematics are calculated using the CCD, ICCD, and NICCD methods, with $\mathbf{\Theta_{in2}}$ as initial value. The position and angle errors of the initial selected joint angles of the UR5 versus the convergence times of the CCD, ICCD, and NICCD methods are illustrated in Figure 7 and versus the number of iterations are illustrated in Figure 8, where the error is represented on the logarithmic axis.

From Figures 7 and 8, ICCD method is better than CCD method in convergence time and iteration times. Especially at the initial iteration, the ICCD method drops much more rapidly than the CCD method in terms of positional error. Specifically, the ICCD method has a position error of 5.8159 after 2 iterations (0.0032 s), while the CCD method needs to iteration 29 times (0.0364 s) to achieve the same error. A similar conclusion can be drawn for angle error.

It is not difficult to conclude from Figures 7 and 8 that the convergence rate of the NICCD method is significantly more rapidly than that of the other two methods. To analyze this reason, Figure 9 shows the evolution process of the NICCD method. From Figure 9, the NICCD method converges after four iterations among which the ICCD method is used for the first two iterations and Newton's method is used for the last two iterations. NICCD method proposed in this paper uses the ICCD method to rapidly find a feasible point that is near to the true solution and then uses Newton's method to obtain a solution that achieves the desired precision. This avoids the common disadvantage of CCD and ICCD methods, that is, the convergence rate is slow when the iteration point approaches the optimal point.

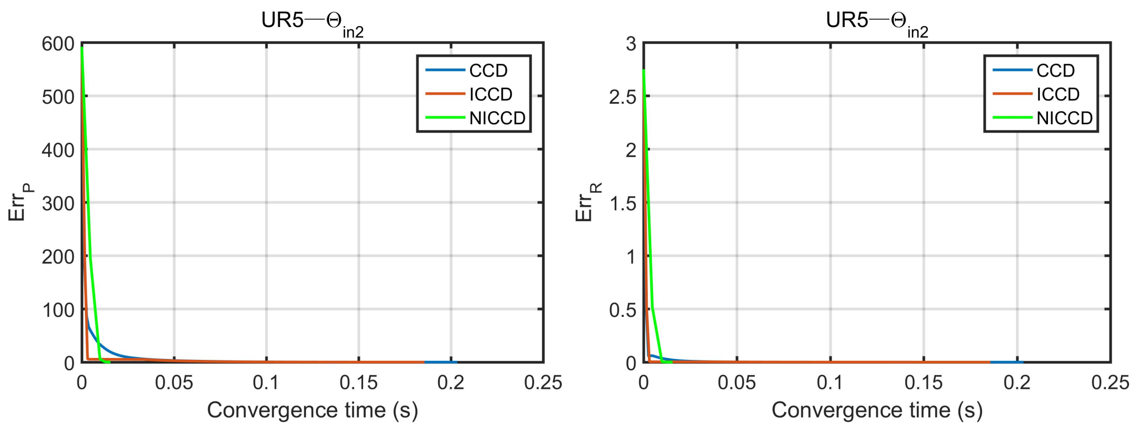

Figure 7. The position and angle errors of the initial selected joint angles of the UR5 versus the convergence times of the CCD, ICCD, and NICCD methods.

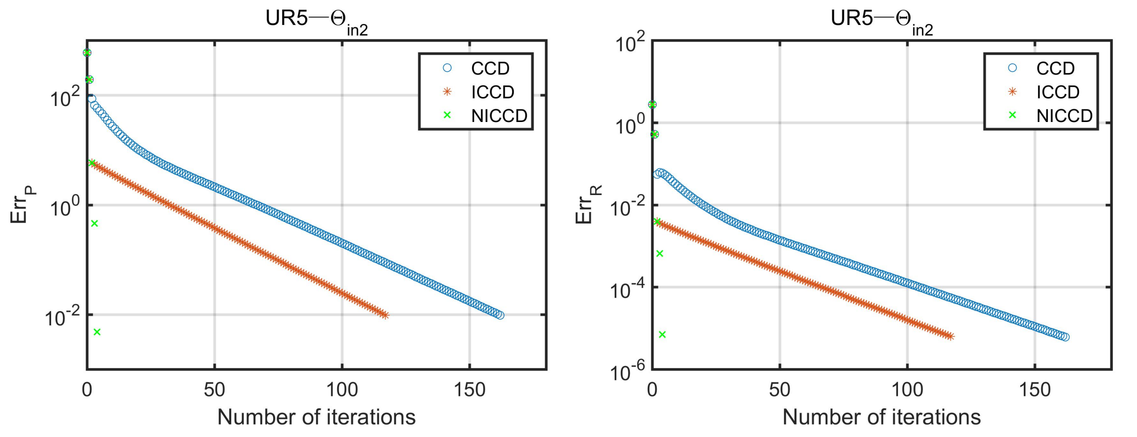

Figure 8. The position and angle errors of the initial selected joint angles of the UR5 versus the number of iterations of the CCD, ICCD, and NICCD methods.

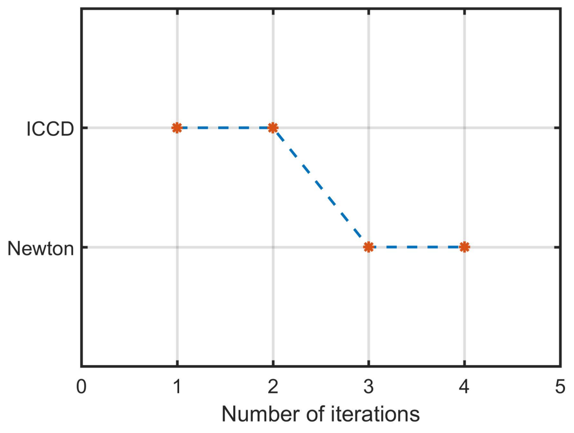

Figure 9. The evolution process of NICCD method.

5. Conclusions and Future Work

This article presented an efficient approach for inverse kinematics that is suitable for any configurations of robots with either revolute or prismatic joints with any arbitrary number of degrees of freedom. Compared with the traditional method, this paper transformed the inverse kinematics problem into the objective function optimization problem, which is based on the least-squares form of the angle error and the position error expressed by the PoE formula. Moreover, the necessary gradient and Hessian matrix required to solve the formula are given. Finally, the NICCD method was proposed to solve the proposed objective function optimization problem.

Five simulations were given in order to illustrate the effectiveness, accuracy, robustness, and generality of the proposed method. Simulation I verified the effectiveness of the proposed method, the results of which suggest that it has less fluctuation and is more effective than the NR method. Simulation II tests the accuracy of the proposed method. The results of which suggest that the approach proposed in this paper can accurately calculate the error of position and the angle to reach 10^{-13} and 10^{-15} respectively, but the NR method provides the error of the angle within 10^{-11}. This also proves the validity of the objective function constructed in Section 3.1. Simulation III demonstrated the fact that the proposed approach is not sensitive to the initial joint variables. Indeed, the correct rate for inverse kinematics calculations with respect to 3P, 3R, SCARA, UR5, and WAM7R are approximately 100%, 100%, 100%, 94.5%, and 93.1%. Simulation IV illustrated the fact that the proposed approach has offline programming abilities and that it is advantageous with respect to the inverse kinematics of continuous trajectories since each iteration only takes 0.0057 s to calculate. Simulation V verified that the ICCD method drops much more rapidly than the CCD method in positional and angle errors in the initial iteration, and the NICCD method combines the respective advantages of the ICCD and Newton's methods while trying to avoid their shortcomings.

The future work will mainly focus on the following aspects. First of all, we will introduce the heuristic adaptive truncation criterion [27] into the NICCD method to improve the convergence rate. Next, we will improve the NICCD method to become a more effective method with respect to avoiding a singularity location and local minimum, which, in turn, will enhance the accuracy and robustness of the method. Last but not least, the NICCD method will be applied to practical applications such as trajectory planning and motion control of the robot.

Author Contributions: Y.C. and X.L.: necessary formula and algorithm design; B.H. and Y.J.: programming; G.L. and X.W.: simulations and data analysis.

Funding: This research was funded by National Key R & D Program of China (2016YFC0803000, 2016YFC0803005)

Acknowledgments: Thanks to Shuo Wang and Jiaqi Feng for providing language help and all our colleagues for providing all types of help during the preparation of this manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

NICCD	Newton-improved cyclic coordinate descent
ICCD	improved cyclic coordinate descent
CCD	cyclic coordinate descent
NR	Newton–Raphson
6R	six revolute
PoE	product-of-exponentials
3R	three-link planar arm
SCARA	selective compliance assembly robot arm
3P	Cartesian manipulator

Appendix A

The two iteration formulas (Equations (54) and (55)) of the ICCD algorithm for a single prismatic joint and a single revolute joints are analytical solutions.

Proof. The second-order sufficient condition for the local minimization of the objective function (f) of inverse kinematics is the following: suppose that $\bigtriangledown^2 f$ is continuous in an open neighborhood of θ^* and that $\bigtriangledown f(\theta^*) = 0$ and $\bigtriangledown^2 f(\theta^*)$ is positive definite matrix; then θ^* is a strict local minimization of f [26].

Injecting Equation (54) in Equation (38), we obtain

$$\frac{\partial f}{\partial \theta_i} = 2\left(\lambda\left(\boldsymbol{P_{ti}} - \boldsymbol{P_{ai}}\right)^T \boldsymbol{v_i} + \lambda(\boldsymbol{P_{ai}} - \boldsymbol{P_{ti}})^T \boldsymbol{v_i}\right) = 0. \tag{A1}$$

According to Equation (41), it can be concluded that

$$\frac{\partial^2 f}{\partial \theta_i^2} = 2\lambda > 0. \tag{A2}$$

Accordingly, Equation (54) is the iteration formula for a single prismatic joint.
Injecting Equation (55) in Equation (39) to get

$$\frac{\partial f}{\partial \theta_i} = r_{1i}\frac{-r_{2i}}{r_{1i}^2 + r_{2i}^2} - r_{2i}\frac{-r_{1i}}{r_{1i}^2 + r_{2i}^2} = 0. \tag{A3}$$

Injecting Equation (55) in Equation (47), we obtain

$$\frac{\partial^2 f}{\partial \theta_i^2} = -r_{1i}\frac{-r_{1i}}{r_{1i}^2 + r_{2i}^2} - r_{2i}\frac{-r_{2i}}{r_{1i}^2 + r_{2i}^2} = 1 > 0. \tag{A4}$$

Accordingly, Equation (55) is the iteration formula for a single revolute joint. □

Appendix B

Here, the required formulas for minimizing the objective function by adjusting two and three consecutive parallel revolute joints at the same time are given.

When there are two consecutive parallel revolute joints (θ_i, θ_{i+1}) in the robot manipulator configuration, the two joints are regarded as variables, and the others are regarded as constants. Accordingly, the gradient of the objective function (Equation (28)) is defined as

$$\bigtriangledown f(\boldsymbol{\Theta}_{i\sim i+1}) = \left[\frac{\partial f}{\partial \theta_i}, \frac{\partial f}{\partial \theta_{i+1}}\right]^T, \tag{A5}$$

where

$$\frac{\partial f}{\partial \theta_i} = a_{12}\cos(\theta_i + \theta_{i+1}) + b_{12}\sin(\theta_i + \theta_{i+1}) + a_1\cos(\theta_i) + b_1\sin(\theta_i), \tag{A6}$$

$$\frac{\partial f}{\partial \theta_{i+1}} = a_{12}\cos(\theta_i + \theta_{i+1}) + b_{12}\sin(\theta_i + \theta_{i+1}) + a_2\cos(\theta_{i+1}) + b_2\sin(\theta_{i+1}), \tag{A7}$$

where

$$a_{12} = 2\lambda\left(-\boldsymbol{P}_{ai+1}^T\boldsymbol{\Omega}_i^2\boldsymbol{v}_i - \boldsymbol{P}_{ti}^T\boldsymbol{\Omega}_i\boldsymbol{P}_{ai+1} + \boldsymbol{P}_{ti}^T\boldsymbol{\Omega}_i^2\boldsymbol{v}_{i+1} + \boldsymbol{v}_{i+1}^T\boldsymbol{\Omega}_i\boldsymbol{v}_i\right) - 2\text{trace}(\boldsymbol{R}_{ti}^T\boldsymbol{\Omega}_i\boldsymbol{R}_{ai+1}), \tag{A8}$$

$$b_{12} = 2\lambda\left(-\boldsymbol{P}_{ai+1}^T\boldsymbol{\Omega}_i\boldsymbol{v}_i - \boldsymbol{P}_{ti}^T\boldsymbol{\Omega}_i^2\boldsymbol{P}_{ai+1} - \boldsymbol{P}_{ti}^T\boldsymbol{\Omega}_i\boldsymbol{v}_{i+1} + \boldsymbol{v}_{i+1}^T\boldsymbol{\Omega}_i^T\boldsymbol{\Omega}_i\boldsymbol{v}_i\right) - 2\text{trace}(\boldsymbol{R}_{ti}^T\boldsymbol{\Omega}_i^2\boldsymbol{R}_{ai+1}), \tag{A9}$$

$$a_1 = 2\lambda\left(\boldsymbol{P}_{ti}^T\boldsymbol{\Omega}_i^2\left(\boldsymbol{v}_i - \boldsymbol{v}_{i+1}\right) - \boldsymbol{v}_{i+1}^T\boldsymbol{\Omega}_i\boldsymbol{v}_i\right), \tag{A10}$$

$$b_1 = 2\lambda\left(\boldsymbol{\Omega}_i\boldsymbol{v}_i - \boldsymbol{P}_{ti}\right)^T\boldsymbol{\Omega}_i\left(\boldsymbol{v}_i - \boldsymbol{v}_{i+1}\right), \tag{A11}$$

$$a_2 = 2\lambda\left(\boldsymbol{P}_{ai+1}^T\boldsymbol{\Omega}_i^2\left(\boldsymbol{v}_i - \boldsymbol{v}_{i+1}\right) - \boldsymbol{v}_{i+1}^T\boldsymbol{\Omega}_i\boldsymbol{v}_i\right), \tag{A12}$$

$$b_2 = 2\lambda\left(\boldsymbol{P}_{ai+1} - \boldsymbol{\Omega}_i\boldsymbol{v}_{i+1}\right)^T\boldsymbol{\Omega}_i\left(\boldsymbol{v}_i - \boldsymbol{v}_{i+1}\right), \tag{A13}$$

where $\boldsymbol{R_{ai+1}}$, $\boldsymbol{P_{ai+1}}$, $\boldsymbol{R_{ti}}$, and $\boldsymbol{P_{ti}}$ are defined as Equations (29), (30), (31), and (32), respectively. The Hessian matrix of the objective function (Equation(28)) is defined as

$$\boldsymbol{H}(\boldsymbol{\Theta}_{i\sim i+1}) = \bigtriangledown^2 f(\boldsymbol{\Theta}_{i\sim i+1}) = \begin{bmatrix} \frac{\partial^2 f}{\partial \theta_i^2} & \frac{\partial^2 f}{\partial \theta_i \partial \theta_{i+1}} \\ \frac{\partial^2 f}{\partial \theta_{i+1} \partial \theta_i} & \frac{\partial^2 f}{\partial \theta_{i+1}^2} \end{bmatrix}, \tag{A14}$$

where

$$\frac{\partial^2 f}{\partial \theta_i^2} = -a_{12}\sin(\theta_i+\theta_{i+1}) + b_{12}\cos(\theta_i+\theta_{i+1}) - a_1\sin(\theta_j) + b_1\cos(\theta_i), \tag{A15}$$

$$\frac{\partial^2 f}{\partial \theta_i \partial \theta_{i+1}} = \frac{\partial^2 f}{\partial \theta_{i+1} \partial \theta_i} = -a_{12}\sin(\theta_i+\theta_{i+1}) + b_{12}\cos(\theta_i+\theta_{i+1}), \tag{A16}$$

$$\frac{\partial^2 f}{\partial \theta_{i+1}^2} = -a_{12}\sin(\theta_i+\theta_{i+1}) + b_{12}\cos(\theta_i+\theta_{i+1}) - a_2\sin(\theta_{i+1}) + b_2\cos(\theta_{i+1}). \tag{A17}$$

When

$$\frac{\partial f}{\partial \theta_i} = 0, \quad \frac{\partial f}{\partial \theta_{i+1}} = 0 \tag{A18}$$

and

$$\frac{\partial^2 f}{\partial \theta_i^2} > 0, \quad \frac{\partial^2 f}{\partial \theta_{i+1}^2} > 0 \tag{A19}$$

are satisfied at the same time, the objective function (Equation (28)) obtains the local minimum. Accordingly, the iterative formula of the CCD method for joint angles (θ_i, θ_{i+1}) can be calculated as

$$\theta_i^{(k+1)} = \text{atan2}\left(-a_{12}\cos(\theta_{i+1}^{(k)}) - b_{12}\sin(\theta_{i+1}^{(k)}) - a_1, b_{12}\cos(\theta_{i+1}^{(k)}) - a_{12}\sin(\theta_{i+1}^{(k)}) + b_1\right), \tag{A20}$$

$$\theta_{i+1}^{(k+1)} = \text{atan2}\left(-a_{12}\cos(\theta_i^{(k+1)}) - b_{12}\sin(\theta_i^{(k+1)}) - a_2, b_{12}\cos(\theta_i^{(k+1)}) - a_{12}\sin(\theta_i^{(k+1)}) + b_2\right). \tag{A21}$$

When there are three consecutive parallel revolute joints $(\theta_i, \theta_{i+1}, \theta_{i+2})$ in the robot manipulator configuration, the three joints are regarded as variables, and the others are regarded as constants. Accordingly, the gradient of the objective function (Equation (28)) is defined as

$$\bigtriangledown f(\boldsymbol{\Theta}_{i\sim i+2}) = \left[\frac{\partial f}{\partial \theta_i}, \frac{\partial f}{\partial \theta_{i+1}}, \frac{\partial f}{\partial \theta_{i+2}}\right]^T, \tag{A22}$$

where

$$\begin{aligned} \frac{\partial f}{\partial \theta_i} &= a_{123}\cos(\theta_i+\theta_{i+1}+\theta_{i+2}) + b_{123}\sin(\theta_i+\theta_{i+1}+\theta_{i+2}) \\ &\quad + a_{12}\cos(\theta_i+\theta_{i+1}) + b_{12}\sin(\theta_i+\theta_{i+1}) + a_1\cos(\theta_i) + b_1\sin(\theta_i), \end{aligned} \tag{A23}$$

$$\begin{aligned} \frac{\partial f}{\partial \theta_{i+1}} &= a_{123}\cos(\theta_i+\theta_{i+1}+\theta_{i+2}) + b_{123}\sin(\theta_i+\theta_{i+1}+\theta_{i+2}) \\ &\quad + a_{12}\cos(\theta_i+\theta_{i+1}) + b_{12}\sin(\theta_i+\theta_{i+1}) \\ &\quad + a_{23}\cos(\theta_{i+1}+\theta_{i+2}) + b_{23}\sin(\theta_{i+1}+\theta_{i+2}) + a_2\cos(\theta_{i+1}) + b_2\sin(\theta_{i+1}), \end{aligned} \tag{A24}$$

$$\begin{aligned} \frac{\partial f}{\partial \theta_{i+2}} &= a_{123}\cos(\theta_i+\theta_{i+1}+\theta_{i+2}) + b_{123}\sin(\theta_i+\theta_{i+1}+\theta_{i+2}) \\ &\quad + a_{23}\cos(\theta_{i+1}+\theta_{i+2}) + b_{23}\sin(\theta_{i+1}+\theta_{i+2}) + a_3\cos(\theta_{i+2}) + b_3\sin(\theta_{i+2}), \end{aligned} \tag{A25}$$

where

$$a_{123} = 2\lambda\left(-\boldsymbol{P}_{ai+2}^{T}\boldsymbol{\Omega}_{i}^{2}\boldsymbol{v}_{i} - \boldsymbol{P}_{ti}^{T}\boldsymbol{\Omega}_{i}\boldsymbol{P}_{ai+2} + \boldsymbol{P}_{ti}^{T}\boldsymbol{\Omega}_{i}^{2}\boldsymbol{v}_{i+2} + \boldsymbol{v}_{i+2}^{T}\boldsymbol{\Omega}_{i}\boldsymbol{v}_{i}\right) - 2\mathrm{Trace}\left(\boldsymbol{R}_{ti}^{T}\boldsymbol{\Omega}_{i}\boldsymbol{R}_{ai+2}\right), \tag{A26}$$

$$b_{123} = 2\lambda\left(-\boldsymbol{P}_{ai+2}^{T}\boldsymbol{\Omega}_{i}\boldsymbol{v}_{i} - \boldsymbol{P}_{ti}^{T}\boldsymbol{\Omega}_{i}^{2}\boldsymbol{P}_{ai+2} - \boldsymbol{P}_{ti}^{T}\boldsymbol{\Omega}_{i}\boldsymbol{v}_{i+2} - \boldsymbol{v}_{i+2}^{T}\boldsymbol{\Omega}_{i}^{2}\boldsymbol{v}_{i}\right) - 2\mathrm{Trace}\left(\boldsymbol{R}_{ti}^{T}\boldsymbol{\Omega}_{i}^{2}\boldsymbol{R}_{ai+2}\right), \tag{A27}$$

$$a_{12} = 2\lambda\left(\boldsymbol{P}_{ti}^{T}\boldsymbol{\Omega}_{i} - \boldsymbol{v}_{i}^{T}\right)\boldsymbol{\Omega}_{i}\left(\boldsymbol{v}_{i+1} - \boldsymbol{v}_{i+2}\right), \tag{A28}$$

$$b_{12} = 2\lambda\left(\boldsymbol{P}_{ti}^{T} + \boldsymbol{v}_{i}^{T}\boldsymbol{\Omega}_{i}\right)\boldsymbol{\Omega}_{i}\left(\boldsymbol{v}_{i+2} - \boldsymbol{v}_{i+1}\right), \tag{A29}$$

$$a_{23} = 2\lambda\left(\boldsymbol{P}_{ai+2}^{T}\boldsymbol{\Omega}_{i} - \boldsymbol{v}_{i+2}^{T}\right)\boldsymbol{\Omega}_{i}\left(\boldsymbol{v}_{i} - \boldsymbol{v}_{i+1}\right), \tag{A30}$$

$$b_{23} = 2\lambda\left(\boldsymbol{P}_{ai+2}^{T} + \boldsymbol{v}_{i+2}^{T}\boldsymbol{\Omega}_{i}\right)\boldsymbol{\Omega}_{i}\left(\boldsymbol{v}_{i} - \boldsymbol{v}_{i+1}\right), \tag{A31}$$

$$a_{1} = 2\lambda\boldsymbol{P}_{ti}^{T}\boldsymbol{\Omega}_{i}^{2}\left(\boldsymbol{v}_{i} - \boldsymbol{v}_{i+1}\right) - 2\lambda\boldsymbol{v}_{i+1}^{T}\boldsymbol{\Omega}_{i}\boldsymbol{v}_{i}, \tag{A32}$$

$$b_{1} = 2\lambda\left(\boldsymbol{P}_{ti}^{T} + \boldsymbol{v}_{i}^{T}\boldsymbol{\Omega}_{i}\right)\boldsymbol{\Omega}_{i}\left(\boldsymbol{v}_{i+1} - \boldsymbol{v}_{i}\right), \tag{A33}$$

$$a_{2} = -2\lambda\boldsymbol{v}_{i+1}^{T}\boldsymbol{\Omega}_{i}\left(\boldsymbol{v}_{i} - \boldsymbol{v}_{i+2}\right) + 2\lambda\boldsymbol{v}_{i+2}^{T}\boldsymbol{\Omega}_{i}\boldsymbol{v}_{i}, \tag{A34}$$

$$b_{2} = -2\lambda\left(\boldsymbol{v}_{i+1}^{T} - \boldsymbol{v}_{i+2}^{T}\right)\boldsymbol{\Omega}_{i}^{2}\left(\boldsymbol{v}_{i+1} - \boldsymbol{v}_{i}\right), \tag{A35}$$

$$a_{3} = -2\lambda\boldsymbol{P}_{ai+2}^{T}\boldsymbol{\Omega}_{i}^{2}\left(\boldsymbol{v}_{i+2} - \boldsymbol{v}_{i+1}\right) - 2\lambda\boldsymbol{v}_{i+2}^{T}\boldsymbol{\Omega}_{i}\boldsymbol{v}_{i+1}, \tag{A36}$$

$$b_{3} = 2\lambda\left(\boldsymbol{P}_{ai+2}^{T} + \boldsymbol{v}_{i+2}^{T}\boldsymbol{\Omega}_{i}\right)\boldsymbol{\Omega}_{i}\left(\boldsymbol{v}_{i+1} - \boldsymbol{v}_{i+2}\right), \tag{A37}$$

where $\boldsymbol{R}_{ai+2}$, $\boldsymbol{P}_{ai+2}$, $\boldsymbol{R}_{ti}$, and $\boldsymbol{P}_{ti}$ are defined as Equations (29), (30), (31), and (32), respectively. The Hessian matrix of the objective function (Equation (28)) is defined as

$$\boldsymbol{H}(\boldsymbol{\Theta}_{i\sim i+2}) = \bigtriangledown^{2} f(\boldsymbol{\Theta}_{i\sim i+2}) = \begin{bmatrix} \frac{\partial^2 f}{\partial\theta_i^2} & \frac{\partial^2 f}{\partial\theta_i \partial\theta_{i+1}} & \frac{\partial^2 f}{\partial\theta_i \partial\theta_{i+2}} \\ \frac{\partial^2 f}{\partial\theta_{i+1} \partial\theta_i} & \frac{\partial^2 f}{\partial\theta_{i+1}^2} & \frac{\partial^2 f}{\partial\theta_{i+1} \partial\theta_{i+2}} \\ \frac{\partial^2 f}{\partial\theta_{i+2} \partial\theta_i} & \frac{\partial^2 f}{\partial\theta_{i+2} \partial\theta_{i+1}} & \frac{\partial^2 f}{\partial\theta_{i+2}^2} \end{bmatrix}, \tag{A38}$$

where

$$\begin{aligned}
\frac{\partial^2 f}{\partial \theta_i^2} &= -a_{123}\sin(\theta_i+\theta_{i+1}+\theta_{i+2}) + b_{123}\cos(\theta_i+\theta_{i+1}+\theta_{i+2}) \\
&\quad - a_{12}\sin(\theta_i+\theta_{i+1}) + b_{12}\cos(\theta_i+\theta_{i+1}) - a_1\sin(\theta_i) + b_1\cos(\theta_i), \qquad \text{(A39)} \\
\frac{\partial^2 f}{\partial \theta_i \partial \theta_{i+1}} = \frac{\partial^2 f}{\partial \theta_{i+1} \partial \theta_i} &= -a_{123}\sin(\theta_i+\theta_{i+1}+\theta_{i+2}) + b_{123}\cos(\theta_i+\theta_{i+1}+\theta_{i+2}) \\
&\quad - a_{12}\sin(\theta_i+\theta_{i+1}) + b_{12}\cos(\theta_i+\theta_{i+1}), \qquad \text{(A40)} \\
\frac{\partial^2 f}{\partial \theta_i \partial \theta_{i+2}} = \frac{\partial^2 f}{\partial \theta_{i+2} \partial \theta_i} &= -a_{123}\sin(\theta_i+\theta_{i+1}+\theta_{i+2}) + b_{123}\cos(\theta_i+\theta_{i+1}+\theta_{i+2}) \qquad \text{(A41)} \\
\frac{\partial^2 f}{\partial \theta_{i+1}^2} &= -a_{123}\sin(\theta_i+\theta_{i+1}+\theta_{i+2}) + b_{123}\cos(\theta_i+\theta_{i+1}+\theta_{i+2}), \\
&\quad - a_{12}\sin(\theta_i+\theta_{i+1}) + b_{12}\cos(\theta_i+\theta_{i+1}) - a_{23}\sin(\theta_{i+1}+\theta_{i+2}) \\
&\quad + b_{23}\cos(\theta_{i+1}+\theta_{i+2}) - a_2\sin(\theta_{i+1}) + b_2\cos(\theta_{i+1}), \qquad \text{(A42)} \\
\frac{\partial^2 f}{\partial \theta_{i+1} \partial \theta_{i+2}} = \frac{\partial^2 f}{\partial \theta_{i+2} \partial \theta_{i+1}} &= -a_{123}\sin(\theta_i+\theta_{i+1}+\theta_{i+2}) + b_{123}\cos(\theta_i+\theta_{i+1}+\theta_{i+2}) \\
&\quad - a_{23}\sin(\theta_{i+1}+\theta_{i+2}) + b_{23}\cos(\theta_{i+1}+\theta_{i+2}), \qquad \text{(A43)} \\
\frac{\partial^2 f}{\partial \theta_{i+2}^2} &= -a_{123}\sin(\theta_i+\theta_{i+1}+\theta_{i+2}) + b_{123}\cos(\theta_i+\theta_{i+1}+\theta_{i+2}) \\
&\quad - a_{23}\sin(\theta_{i+1}+\theta_{i+2}) + b_{23}\cos(\theta_{i+1}+\theta_{i+2}) - a_3\sin(\theta_{i+2}) + b_3\cos(\theta_{i+2}). \qquad \text{(A44)}
\end{aligned}$$

When

$$\frac{\partial f}{\partial \theta_i} = 0, \qquad \frac{\partial f}{\partial \theta_{i+1}} = 0, \qquad \frac{\partial f}{\partial \theta_{i+2}} = 0 \qquad \text{(A45)}$$

and

$$\frac{\partial^2 f}{\partial \theta_i^2} > 0, \qquad \frac{\partial^2 f}{\partial \theta_{i+1}^2} > 0, \qquad \frac{\partial^2 f}{\partial \theta_{i+2}^2} > 0 \qquad \text{(A46)}$$

are satisfied at the same time, the objective function (Equation (28)) obtains the local minimum. Accordingly, the iterative formula of the CCD method for joint angles $(\theta_i, \theta_{i+1}, \theta_{i+2})$ can be calculated as

$$\begin{aligned}
\theta_i^{(k+1)} &= \text{atan2}\Big(-a_{123}\cos(\theta_{i+1}^{(k)}+\theta_{i+2}^{(k)}) - b_{123}\sin(\theta_{i+1}^{(k)}+\theta_{i+2}^{(k)}) - a_{12}\cos(\theta_{i+1}^{(k)}) - b_{12}\sin(\theta_{i+1}^{(k)}) - a_1, \\
&\qquad b_{123}\sin(\theta_{i+1}^{(k)}+\theta_{i+2}^{(k)}) - a_{123}\cos(\theta_{i+1}^{(k)}+\theta_{i+2}^{(k)}) + b_{12}\cos(\theta_{i+1}^{(k)}) - a_{12}\sin(\theta_{i+1}^{(k)}) + b_1\Big), \qquad \text{(A47)} \\
\theta_{i+1}^{(k+1)} &= \text{atan2}\Big(-a_{123}\cos(\theta_i^{(k+1)}+\theta_{i+2}^{(k)}) - b_{123}\sin(\theta_i^{(k+1)}+\theta_{i+2}^{(k)}) - a_{12}\cos(\theta_i^{(k+1)}) \\
&\qquad - b_{12}\sin(\theta_{i+1}^{(k)}) - a_{23}\cos(\theta_{i+2}^{(k)}) - b_{23}\sin(\theta_{i+2}^{(k)}) - a_2, \\
&\qquad b_{123}\sin(\theta_i^{(k+1)}+\theta_{i+2}^{(k)}) - a_{123}\cos(\theta_i^{(k+1)}+\theta_{i+2}^{(k)}) + b_{12}\cos(\theta_i^{(k+1)}) \\
&\qquad - a_{12}\sin(\theta_i^{(k+1)}) + b_{23}\sin(\theta_{i+2}^{(k)}) - a_{23}\cos(\theta_{i+2}^{(k)}) + b_2\Big), \qquad \text{(A48)} \\
\theta_{i+2}^{(k+1)} &= \text{atan2}\Big(-a_{123}\cos(\theta_i^{(k+1)}+\theta_{i+1}^{(k+1)}) - b_{123}\sin(\theta_i^{(k+1)}+\theta_{i+1}^{(k+1)}) - a_{23}\cos(\theta_{i+1}^{(k+1)}) - b_{23}\sin(\theta_{i+1}^{(k+1)}) - a_3, \\
&\qquad b_{123}\sin(\theta_i^{(k+1)}+\theta_{i+1}^{(k+1)}) - a_{123}\cos(\theta_i^{(k+1)}+\theta_{i+1}^{(k+1)}) + b_{23}\cos(\theta_{i+1}^{(k+1)}) - a_{23}\sin(\theta_{i+1}^{(k+1)}) + b_3\Big). \qquad \text{(A49)}
\end{aligned}$$

Appendix C

Prove that Equation (74) is established.

Proof. It's obvious that this proof can be transformed into proof

$$\max_{\theta_a,\theta_b,\boldsymbol{\omega}_a,\boldsymbol{\omega}_b} g\left(\theta_a,\theta_b,\boldsymbol{\omega}_a,\boldsymbol{\omega}_b\right)=8, \tag{A50}$$

where

$$g\left(\theta_a,\theta_b,\boldsymbol{\omega}_a,\boldsymbol{\omega}_b\right)=\operatorname{trace}\left(\left(\boldsymbol{R_a}(\theta_a,\boldsymbol{\omega}_a)-\boldsymbol{R_b}(\theta_b,\boldsymbol{\omega}_b)\right)^T\left(\boldsymbol{R_a}(\theta_a,\boldsymbol{\omega}_a)-\boldsymbol{R_b}(\theta_b,\boldsymbol{\omega}_b)\right)\right), \tag{A51}$$

where both $\boldsymbol{R}(\theta_a,\boldsymbol{\omega}_a)$ and $\boldsymbol{R}(\theta_b,\boldsymbol{\omega}_b)$ can be calculated by Equation (8) and $\theta_a,\theta_b,\boldsymbol{\omega}_a,\boldsymbol{\omega}_b$ are all arbitrary.

Injecting Equation (8) in Equation (A51), we can obtain

$$\begin{aligned} g\left(\theta_a,\theta_b,\boldsymbol{\omega}_a,\boldsymbol{\omega}_b\right)=-2\operatorname{trace}\Big(&\boldsymbol{\Omega}_a^2+\boldsymbol{\Omega}_b^2-\boldsymbol{\Omega}_a\boldsymbol{\Omega}_b\sin(\theta_a)\sin(\theta_b)-\boldsymbol{\Omega}_a^2\cos(\theta_a)-\boldsymbol{\Omega}_b^2\cos(\theta_b)\\ &+\boldsymbol{\Omega}_a^2\boldsymbol{\Omega}_b^2(1-\cos(\theta_a))(1-\cos(\theta_b))\Big). \end{aligned} \tag{A52}$$

Because of

$$\operatorname{trace}\left(\boldsymbol{\Omega}_a^2\right)=-2, \tag{A53}$$

$$\operatorname{trace}\left(\boldsymbol{\Omega}_b^2\right)=-2, \tag{A54}$$

$$\operatorname{trace}\left(\boldsymbol{\Omega}_a^2\boldsymbol{\Omega}_b^2\right)=1+\left(\boldsymbol{\omega}_a\cdot\boldsymbol{\omega}_b\right)^2, \tag{A55}$$

$$\operatorname{trace}\left(\boldsymbol{\Omega}_a\boldsymbol{\Omega}_b\right)=-2\left(\boldsymbol{\omega}_a\cdot\boldsymbol{\omega}_b\right), \tag{A56}$$

Equation (A52) can be simplified as follows,

$$\begin{aligned} g\left(\theta_a,\theta_b,t\right)=&(6-2t^2)+(2t^2-2)\cos(\theta_a)+(2t^2-2)\cos(\theta_b)\\ &-4t\sin(\theta_a)\sin(\theta_b)+(2+2t^2)\cos(\theta_a)\cos(\theta_b), \end{aligned} \tag{A57}$$

where

$$t=\boldsymbol{\omega}_a\cdot\boldsymbol{\omega}_b\in[-1,1]. \tag{A58}$$

In order to find the extreme value, $\bigtriangledown g\left(\theta_a^*,\theta_b^*,t^*\right)=0$ must be satisfied.

$$\left.\frac{\partial g}{\partial t}\right|_{\theta_a=\theta_a^*,\theta_b=\theta_b^*,t=t^*}=-4\Big(t^*-t^*\cos(\theta_a^*)-t^*\cos(\theta_b^*)+\sin(\theta_a^*)\sin(\theta_b^*)+t^*\cos(\theta_a^*)\cos(\theta_b^*)\Big)=0, \tag{A59}$$

$$\left.\frac{\partial g}{\partial \theta_a}\right|_{\theta_a=\theta_a^*,\theta_b=\theta_b^*,t=t^*}=-(2t^{*2}-2)\sin(\theta_a^*)-4t^*\cos(\theta_a^*)\sin(\theta_b^*)+(2+2t^{*2})\sin(\theta_a^*)\cos(\theta_b^*)=0, \tag{A60}$$

$$\left.\frac{\partial g}{\partial \theta_b}\right|_{\theta_a=\theta_a^*,\theta_b=\theta_b^*,t=t^*}=-(2t^{*2}-2)\sin(\theta_b^*)-4t^*\sin(\theta_a^*)\cos(\theta_b^*)+(2+2t^{*2})\cos(\theta_a^*)\sin(\theta_b^*)=0. \tag{A61}$$

In order to solve Equations (A59)∼(A61), we can add Equation (A60) and Equation (A61) to get

$$\left.\left(\frac{\partial g}{\partial \theta_a}+\frac{\partial g}{\partial \theta_b}\right)\right|_{\theta_a=\theta_a^*,\theta_b=\theta_b^*,t=t^*}=2(1-t)\Big((1+t)\left(\sin(\theta_a^*)+\sin(\theta_b^*)\right)+(1-t)\sin(\theta_a^*+\theta_b^*)\Big)=0. \tag{A62}$$

Therefore, Equation (A62) is established if and only if the following conditions are met.

$$t^* = -1, \quad \theta_a^* + \theta_b^* = 0; \tag{A63}$$

$$t^* = 1; \tag{A64}$$

$$t^* = -1, \quad \theta_a^* + \theta_b^* = \pi; \tag{A65}$$

$$t^* = 1, \quad \theta_a^* - \theta_b^* = \pi; \tag{A66}$$

$$\theta_a^* = 0, \quad \theta_b^* = 0. \tag{A67}$$

Equations (A63) and (A64) do not satisfy Equation (A59), but others all satisfy Equations (A59)∼(A61). Injecting Equations (A65)∼(A67) in Equation (A57), we can obtain

$$g\left(\theta_a, \theta_b, t\right) = 8, \quad \left(\theta_a^* + \theta_b^* = \pi, t^* = -1\right); \tag{A68}$$

$$g\left(\theta_a, \theta_b, t\right) = 8, \quad \left(\theta_a^* - \theta_b^* = \pi, t^* = 1\right); \tag{A69}$$

$$g\left(\theta_a, \theta_b, t\right) = 0, \quad \left(\theta_a^* = 0, \theta_b^* = 0\right). \tag{A70}$$

Therefore, Equation (74) is established.

$$\max_{\theta_a, \theta_b, \boldsymbol{\omega}_a, \boldsymbol{\omega}_b} g\left(\theta_a, \theta_b, \boldsymbol{\omega}_a, \boldsymbol{\omega}_b\right) = \max_{\theta_a, \theta_b, t} g\left(\theta_a, \theta_b, t\right) = 8. \tag{A71}$$

□

References

1. Jahanpour, J.; Motallebi, M.; Porghoveh, M. A Novel Trajectory Planning Scheme for Parallel Machining Robots Enhanced with NURBS Curves. *J. Intell. Robot. Syst.* **2016**, *82*, 257–275. [CrossRef]
2. Pivarciova, E.; Bozek, P.; Turygin, Y.; Zajacko, I.; Shchenyatsky, A.; Vaclav, S.; Cisar, M.; Gemela, B. Analysis of control and correction options of mobile robot trajectory by an inertial navigation system. *Int. J. Adv. Robot. Syst.* **2018**, *15*. doi:10.1177/1729881418755165. [CrossRef]
3. Hartenberg, R.S.; Denavit, J. *Kinematic Synthesis of Linkages*; McGraw-Hill: New York, NY, USA, 1965.
4. Siciliano, B.; Sciavicco, L.; Villani, L.; Oriolo, G. *Robotics Modelling, Planning and Control*; Springer: London, UK, 2009.
5. Pieper, L.D. Kinematics of Manipulators under Computer Control. Ph.D. Thesis, Stanford University, Stanford, CA, USA, 1968.
6. Tsai, L.W.; Morgan, A.P. Solving the kinematics of the most general six- and five-degree-of-freedom manipulators by continuation methods. *J. Mech. Des.* **1985**, *107*, 189. [CrossRef]
7. Primrose, E.J.F. On the input-output equation of the general 7R-mechanism. *Mech. Mach. Theory* **1986**, *21*, 509–510. [CrossRef]
8. Manocha D, C.J.F.. Efficient inverse kinematics for general 6R manipulators. *IEEE Trans. Robot. Autom.* **1994**, *10*, 648–657. [CrossRef]
9. Khatib, O. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE J. Robot. Autom.* **1987**, *3*, 43–53. [CrossRef]
10. Chiaverini, S.; Siciliano, B.; Egeland, O. Review of damped least squares inverse kinematics with experiments on an industrial robot manipulator. *IEEE Trans Control. Syst. Technol.* **1994**, *2*, 123–134. [CrossRef]
11. Xu, W.; Zhang, J.; Liang, B.; Bing, L. Singularity analysis and avoidance for robot manipulators with non-spherical wrists. *IEEE Trans. Ind. Electron.* **2015**, *63*, 277–290. [CrossRef]
12. Goldenberg, A.A.; Benhabib, B.; Fenton, R.G. A complete generalized solution to the inverse kinematics of robots. *IEEE J. Robot. Autom.* **1985**, *1*, 14–20. [CrossRef]
13. Luenberger, D.G.; Ye, Y. *Linear and Nonlinear Programming*; Springer: Berlin, Germany, 2008.
14. Kelemen, M.; Virgala, I.; Lipták, T.; Miková, L.; Filakovský, F.; Bulej, V. A novel approach for a inverse kinematics solution of a redundant manipulator. *Appl. Sci.* **2018**, *8*. doi:10.3390/app8112229. [CrossRef]

15. Zaplana, I.; Basanez, L. A novel closed-form solution for the inverse kinematics of redundant manipulators through workspace analysis. *Mech. Mach. Theory* **2018**, *121*, 829–843. [CrossRef]
16. Qiao, S.; Liao, Q.; Wei, S.; Su, H.J. Inverse kinematic analysis of the general 6R serial manipulators based on double quaternions. *Mech. Mach. Theory* **2010**, *45*, 193–199. [CrossRef]
17. Menini, L.; Tornambè, A. A Lie symmetry approach for the solution of the inverse kinematics problem. *Nonlinear Dyn.* **2012**, *69*, 1965–1977. [CrossRef]
18. Köker, R.; Öz, C.; Çakar, T.; Ekiz, H. A study of neural network based inverse kinematics solution for a three-joint robot. *Robot. Auton. Syst.* **2004**, *49*, 227–234. [CrossRef]
19. Köker, R.; Çakar, T.; Sari, Y. A neural-network committee machine approach to the inverse kinematics problem solution of robotic manipulators. *Eng. Comput.* **2014**, *30*, 641–649. [CrossRef]
20. Kalra, P.; Mahapatra, P.B.; Aggarwal, D.K. An evolutionary approach for solving the multimodal inverse kinematics problem of industrial robots. *Mech. Mach. Theory* **2006**, *41*, 1213–1229. [CrossRef]
21. Tabandeh, S.; Melek, W.W.; Clark, C.M. An adaptive niching genetic algorithm approach for generating multiple solutions of serial manipulator inverse kinematics with applications to modular robots. *Robotica* **2010**, *28*, 493–507. [CrossRef]
22. Rokbani, N.; Alimi, A.M. Inverse kinematics using particle swarm optimization, a statistical analysis. *Procedia Eng.* **2013**, *64*, 1602–1611. [CrossRef]
23. Koker, R. A genetic algorithm approach to a neural-network-based inverse kinematics solution of robotic manipulators based on error minimization. *Inf. Sci.* **2013**, *222*, 528–543. [CrossRef]
24. Köker, R.; Çakar, T. A neuro-genetic-simulated annealing approach to the inverse kinematics solution of robots: a simulation based study. *Eng. Comput.* **2016**, *32*, 1–13. [CrossRef]
25. Lynch, K.; Park, F. *Modern Robotics: Mechanics, Planning, and Control*; Cambridge Univeristy Press: Cambridge, UK, 2017.
26. Nocedal, J.; Wright, S.J. *Numerical Optimization*; Springer: Berlin, Germany, 2006.
27. Caliciotti, A.; Fasano, G.; Nash, S.G.; Roma, M. An adaptive truncation criterion, for linesearch-based truncated Newton methods in large scale nonconvex optimization. *Oper. Res. Lett.* **2017**, *46*, 7–12. [CrossRef]
28. Caliciotti, A.; Fasano, G.; Nash, S.G.; Roma, M. Data and performance profiles applying an adaptive truncation criterion, within linesearch-based truncated Newton methods, in large scale nonconvex optimization. *Data Brief* **2018**, *17*, 246–255. [CrossRef] [PubMed]