

Computation of Inverse Kinematics of Redundant Manipulator Using Particle Swarm Optimization Algorithm and Its Combination with Artificial Neural Networks [†]

Pedram Monfared, Xiaoning Fei and Wei Peng *

Department of Industrial Systems Engineering, University of Regina, Regina, SK S4S 0A2, Canada; pmp160@uregina.ca (P.M.); feixiaoning96@gmail.com (X.F.)

* Correspondence: wei.peng@uregina.ca

[†] Presented at the 1st International Conference on Industrial, Manufacturing, and Process Engineering (ICIMP-2024), Regina, Canada, 27–29 June 2024.

Abstract: The industry heavily relies on robotic manipulators at present. Despite their age, recent methodologies have significantly advanced their functionality, integrating them into daily life. Rescue robots are popular. However, their precision and reaction speed are the main issues for real-world rescuing. This research aims to enhance the precision of rescue robotic manipulators' end-effectors in real time. It achieves this by deriving and solving inverse kinematics formulations for 2-, 3-, and 4-link manipulators using Particle Swarm Optimization (PSO). The PSO method proves highly accurate, outperforming previous studies utilizing Artificial Neural Networks (ANNs). While PSO requires more time than ANNs, a hybrid approach, PSO-ANN, balances accuracy and speed, offering real-time solutions with minimal errors, and contributing to a precise methodology for real-time robotic manipulator operations.

Keywords: robotics manipulator; inverse kinematics; Particle Swarm Optimization; Artificial Neural Network; PSO-ANN



Citation: Monfared, P.; Fei, X.; Peng, W. Computation of Inverse Kinematics of Redundant Manipulator Using Particle Swarm Optimization Algorithm and Its Combination with Artificial Neural Networks. *Eng. Proc.* **2024**, *76*, 58. <https://doi.org/10.3390/engproc2024076058>

Academic Editors: Golam Kabir, Sharfuddin Khan, Mohammad Khondoker and Hussameldin Ibrahim

Published: 29 October 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Robotics manipulators are one of the most important parts of today's industry. Most fields across industries, from small companies to the largest ones, are using them in different types of applications. Even though these manipulators are very effective in industry, they have many applications in other fields, like laboratories, surgical, theoretical, nuclear applications, etc. [1]. Researchers have been working on manipulators for so many years. The fields do not have a new application for robotics manipulators, but as the technology is rapidly changing, there are always new ways and applications for the manipulators. New methodologies usually provide better accuracy than the previous methods, and that is why there is always research to do on robotics manipulators.

Degrees of freedom (DOF) are the most important factor in any manipulator. Sometimes the DOF of a manipulator would be more than the ones it needs to perform a task; in this case, that manipulator is called a kinematically redundant manipulator [2,3]. To be specific, redundant manipulators are the ones whose end-effector can follow a desired path by using fewer DOF than the DOF which the manipulator already has. Most applications are using redundant manipulators, as the factor of redundancy is helping the manipulator to follow the desired path without encountering any obstacles.

The logic of kinematics is to study robots (manipulator, mobile, drone, etc.) without considering any force applied by the environment [4]. Kinematics itself has two types, forward and inverse kinematics. Forward Kinematics (FK) is when the position and the orientation of the end-effector can be calculated using the joint angles. Inverse Kinematics (IK) is when the joint angles can be calculated using the end-effector's coordinates [5].

Inverse kinematics has been used in industries for a long time and is usually the best way to solve problems in the real world, where there is always a final solution and the way to realize it is the question. For instance, in industries where there is a desired path and the end-effector of a manipulator must track that path, there is no information available about the position of the link in each second of the movement of the manipulator. So, the answers to these questions are always inverse kinematics. Moreover, the main point of this method is that as there are always obstacles in the real world, all these obstacles can be simulated in this method as the angular constraints for each link.

The end-effector's path planning of a two-link manipulator is not challenging because there are no more than two answers for each position of the end-effector, and it is not redundant. But, when the number of links becomes more than two, it will be challenging to solve the inverse kinematics of the manipulator because there will be more than one solution for the inverse kinematics method. The method is either numerical or non-numerical solutions, and the manipulator would count as a redundant manipulator. Moreover, in the literature review of this research, many solutions aim to solve the inverse kinematics problems, via either numerical or non-numerical methods. Also, these solutions are for redundant and non-redundant manipulators.

Therefore, the objective of this research is to achieve a more precise position for end-effectors of rescue robotic manipulator in real time and within reasonable calculation time. This paper will develop a combination approach mixing the Particle Swarm Optimization (PSO) method and the Artificial Neural Network (ANN) technique, namely PSO-ANN, to solve the inverse kinematics of a manipulator and achieve the objectives of this project.

In Section 2, the literature review will discuss numerical and non-numerical methods to solve IK problems. Section 3 will focus on deriving IK formulations for various manipulators to enhance understanding and programming. Section 4 will discuss the results and comparison of the PSO method and the hybrid PSO-ANN approach.

2. Literature Review

Robotic manipulators are integral components in various industries, performing tasks ranging from assembly line operations to surgical procedures. Central to the functionality of these manipulators is the ability to accurately determine the joint configurations required to achieve the desired end-effector positions and orientations, a problem known as inverse kinematics. Traditionally, the methodology of solving an inverse kinematics problem was very difficult and it took a lot of time to solve [5,6]. Many conventional methods work on primitive robots, such as very simple iterative methods, just by observing the results or algebraic methods, but as the robots are becoming more complex with higher DOF, the primitive methods do not work anymore and they have to be replaced with some new ways [7,8]. In the remainder of this section, a brief literature review on this topic and how PSO is related to inverse kinematics is presented.

Géradin et al. [9] and Gomez et al. [10] utilized geometrical and numerical methods for lower DOF manipulators, illustrating their limitations in handling more intricate systems.

In response to the shortcomings of traditional methods, researchers have explored alternative approaches, leveraging advanced computational techniques. Aghajarian and Kiani [11] applied Artificial Neuro-Fuzzy Inference Systems (ANFIS) on a PUMA 560 robotics arm to solve inverse kinematics for a 6-DOF manipulator. While ANFIS exhibited good modeling capabilities, it stopped updating and cost more time than other modern versions to solve IK problems.

Artificial Intelligence (AI) has also played a significant role in IK solutions. Tejomurtola and Kak [12] focused on a PUMA 500 robotic arm that has three links and three joints. They considered all the links and joints rigid. The proposed Neural Network using back-propagation methodology achieved higher accuracy than before. Ming et al. [13] employed Fuzzy Logic theory for a 2-DOF manipulator, demonstrating enhanced accuracy and speed compared to primitive methods. Similarly, Batista et al. [14] integrated Particle Swarm

Optimization (PSO) with Least Square (LS) and Recursive Least Square (RLS) for a 3-DOF manipulator, achieving superior results in terms of accuracy and convergence.

Recently, numerical methods and geometrical algorithms have emerged as promising avenues for addressing IK challenges in higher DOF manipulators. Zhao et al. [15] explored Newton-Raphson and optimization algorithms for 7-DOF manipulators, and Dereli and Köker [16] used an optimization method called firefly algorithm.

Song et al. [17] optimized neural networks using the Artificial Fish Swarm Algorithm (AFSA) to improve gyroscopic measurements. Their approach highlighted the potential of combining computational intelligence techniques to enhance accuracy and efficiency in robotic systems.

Oh et al. [18] applied the Newton-Raphson method to address end-effector orientation constraints in a 7-DOF manipulator, underscoring the importance of incorporating task-specific constraints into inverse kinematics solutions. Additionally, Zhao et al. [19] utilized Ant Colony Optimization (ACO) to navigate obstacles in a 10-DOF manipulator, introducing novel geometric methodologies to optimize robotic trajectories.

Recent research efforts have also focused on integrating modern optimization techniques with existing methodologies to achieve superior inverse kinematics solutions. Liu et al. [20], in 2021, utilized PSO to transform inverse kinematics into an optimal solution problem for a 6-DOF manipulator, achieving promising theoretical and experimental results.

Starke et al. [21], in 2016, proposed a biologically inspired approach, merging genetic and particle swarm optimization to efficiently solve inverse kinematics problems across various joint chains. Their hybrid methodology demonstrated high accuracy and adaptability, particularly in real-time robotic applications.

Dereli and Koker [22], in 2018, worked on a 7-DOF serial manipulator. The end-effector of the manipulator reached the predetermined position with an error of 3.64×10^{-3} . They even compared their results with the conventional PSO and the Random Inertia Weight too. Their Global-Local Best Inertia Weight PSO has the most accurate answers.

In summary, recent advancements in solving robotic inverse kinematics have shifted towards utilizing modern optimization techniques such as PSO. Many recent papers have applied the PSO method on different types of robotic arms, and the results show that the PSO method is more accurate and faster than the conventional methods. Also, combining PSO methods with other methods can achieve even better results [23].

3. Methodology

3.1. Inverse Kinematics Computations

The robotic manipulator, a chain of linked joints, was initially designed for inaccessible tasks such as handling radioactive materials. Today, advancements in technology have expanded their usage, incorporating flexible links and various joint types, like revolute and prismatic. A revolute joint's function is akin to that of elbows, rotating around their axis with applied torque, while prismatic joints move along their axis, useful in compact spaces. These manipulators find applications in surgery, space, welding, etc. [1].

Joint space encompasses all joint variables, mainly for positioning individual links and the end-effector. Degrees of freedom (DOF) are determined by the number of joints. The end-effector, vital for tasks like grasping and drilling, requires precise coordination. Forward kinematics is the easiest way to determine the coordination of the end-effector. However, in practical scenarios, the end-effector's position is predetermined, necessitating inverse kinematics solutions. Industrial applications often face constraints on both end-effector position and manipulator configuration, leading to complex problem-solving, especially with redundant manipulators.

3.1.1. Two-DOF with Two Revolute Joints Robot Manipulators

The schematic of a two-link manipulator with two revolute joints is shown in Figure 1a and forward kinematics calculates the coordination values, x and y , by Equations (1) and (2), respectively. θ_2 is calculated by Equation (3):

$$x = l_1 * \cos(\theta_1) + l_2 * \cos(\theta_1 + \theta_2) \quad (1)$$

$$y = l_1 * \sin(\theta_1) + l_2 * \sin(\theta_1 + \theta_2) \quad (2)$$

$$\theta_2 = \pi \pm \alpha, \quad \alpha = \cos^{-1} \left(\frac{l_1^2 + l_2^2 - r^2}{2 l_1 l_2} \right) \quad (3)$$

where $r = \sqrt{x^2 + y^2}$. For $\alpha \neq 0$ there are two answers of θ_1 . θ_2 , obtained by Equation (4):

$$\theta_1 = \tan^{-1}(y, x) \pm \beta, \quad \beta = \cos^{-1} \left(\frac{r^2 + l_1^2 - l_2^2}{2 l_1 r} \right) \quad (4)$$

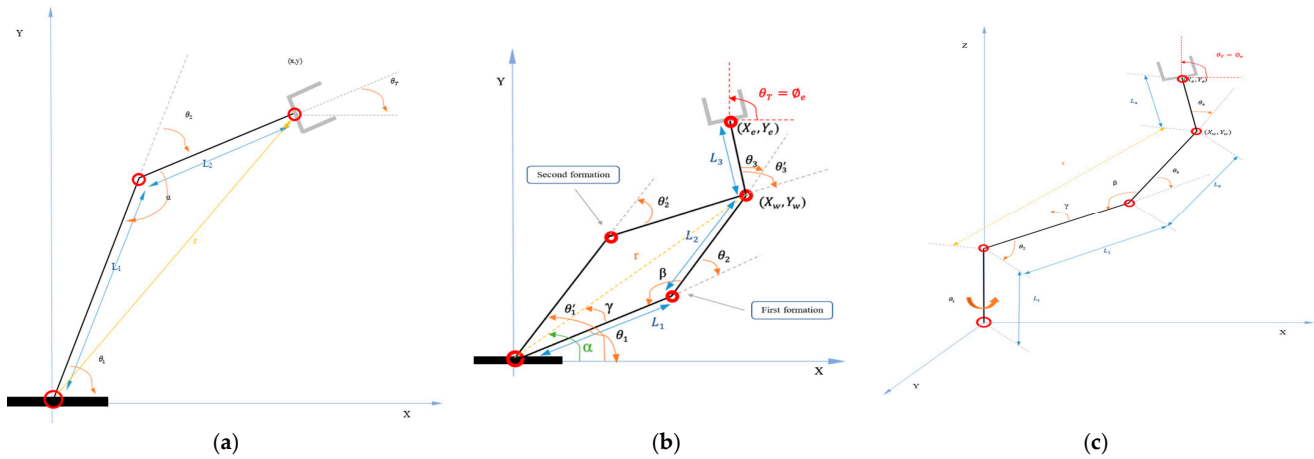


Figure 1. (a) 2-DOF robot manipulator; (b) 3-DOF redundant manipulator; (c) 4-DOF redundant manipulator [23].

There is a possibility of multiple solutions if the desired position of the end-effector is in the workspace of the robotic manipulator and more than one joint configuration is can lead the end-effector to that specific point [4].

3.1.2. Three-DOF Redundant Manipulator

The forward kinematics equations of a 3-DOF redundant manipulator are given by Equations (5) and (6) [4]:

$$x_e = l_1 \cos \theta_1 + l_2 \cos(\theta_1 + \theta_2) + l_3 \cos(\theta_1 + \theta_2 + \theta_3) \quad (5)$$

$$y_e = l_1 \sin \theta_1 + l_2 \sin(\theta_1 + \theta_2) + l_3 \sin(\theta_1 + \theta_2 + \theta_3) \quad (6)$$

where l_1 , l_2 , and l_3 are the fixed length of the links and θ_1 , θ_2 , and θ_3 are the joint angles. Values x_e and y_e are the coordinate values of the manipulator end-effector. A 3-DOF manipulator is shown in Figure 1b, which is redundant because all the links and joints are moving in a two-dimensional workspace. Furthermore, we concluded from Equations (5) and (6), the IK of this manipulator does not have a unique answer. However, by considering at least one constraint as a joint variable, it can be possible to find an explicit solution.

3.1.3. Four-DOF Redundant Manipulator

Figure 1c illustrates a 4-DOF redundant manipulator in a three-dimensional space with four joint angles. Forward kinematics of this manipulator is formulated as follows Equations (7)–(9) [24]:

$$x_e = \cos \theta_1 [l_2 \cos \theta_2 + l_3 \cos (\theta_2 + \theta_3)] + l_4 \cos \theta_1 \cos (\theta_2 + \theta_3 + \theta_4) \quad (7)$$

$$y_e = \sin \theta_1 [l_2 \cos \theta_2 + l_3 \cos (\theta_2 + \theta_3)] + l_4 \sin \theta_1 \cos (\theta_2 + \theta_3 + \theta_4) \quad (8)$$

$$z_e = [l_1 + l_2 \sin \theta_2 + l_3 \sin (\theta_2 + \theta_3)] + l_4 \sin (\theta_2 + \theta_3 + \theta_4) \quad (9)$$

where x_e , y_e , and z_e are the coordination of the end-effector and θ_1 , θ_2 , θ_3 , and θ_4 are the joint angles. The manipulator operates in three dimensions, yet it has four joint angles, making it impossible to determine a unique solution for its IK. To solve this, constraints must be imposed carefully, to avoid ineffective solutions. Two cases of constraints are presented below, offering explicit solutions for the manipulator's inverse kinematics. The end-effector's orientation is determined by the summation of the last three joints of the manipulator as shown in Equation (10).

$$\varnothing_T = \theta_2 + \theta_3 + \theta_4 \quad (10)$$

$$x_w = [x_e - \cos \theta_1 l_4 \cos \varnothing_T] \quad (11)$$

$$y_w = [y_e - \sin \theta_1 l_4 \cos \varnothing_T] \quad (12)$$

$$z_w = [z_e - l_4 \sin \varnothing_T] \quad (13)$$

The orientations of the end-effector's coordinate values can be found from Equations (11)–(13), the constraints of which define the end-effector's orientation, with θ_1 serving as an additional constraint. Equations (14)–(18) enable deriving θ_1 as the end-effector's coordination. Similarly, coordinating the wrist and end-effector orientation leads to a singular solution for IK. All components of the manipulator act like a chain; adding a wrist coordinate constraint may solve IK, but the end-effector's coordination becomes dependent on the wrist's solution. By integrating end-effector coordination and constraints, a unique solution for the 4-DOF redundant manipulator's IK is obtained as follows:

$$l_2^2 + l_3^2 - 2 l_2 l_3 \cos \beta = r^2, \quad r^2 = (x_w^2 + y_w^2) + (z_w - l_1)^2 \Rightarrow \beta = \cos^{-1} \left(\frac{l_3^2 + l_2^2 - r^2}{2 l_3 l_2} \right) \quad (14)$$

$$\theta_3 = \pi - \beta = \theta_3 = \cos^{-1} \left(\frac{r^2 - l_3^2 - l_2^2}{2 l_3 l_2} \right) \quad (15)$$

$$r^2 + l_2^2 - 2 r l_2 \cos \gamma = l_3^2 \Rightarrow \gamma = \cos^{-1} \left(\frac{r^2 + l_2^2 - l_3^2}{2 r l_2} \right) \text{ OR } \gamma = \tan^{-1} \left(\frac{l_3 \sin \theta_3}{l_2 + l_3 \cos \theta_3} \right) \quad (16)$$

$$\alpha = \tan^{-1} \left(\frac{z_w - l_1}{\sqrt{(x_w^2 + y_w^2)}} \right) \Rightarrow \theta_2 = \alpha - \gamma \quad (17)$$

$$\theta_4 = \varnothing_T - \theta_2 - \theta_3 \quad (18)$$

This subsection focuses on formulating inverse kinematics for 2-, 3-, and 4-DOF redundant manipulators. For 2-DOF manipulators, with a workspace in two dimensions, both forward and inverse kinematics are straightforward. However, for 3-DOF manipulators, additional constraints are necessary due to the lack of a specific solution. Similarly, the formulation for 4-DOF manipulators requires additional constraints, given their redundancy.

3.2. Particle Swarm Optimization (PSO)

PSO is a popular optimization method mimicking animal social behavior. It utilizes particles representing individuals in a group, each with a position and mission. Path planning, which could be complex and time-consuming without environmental information, is a popular application of

PSO, where PSO outperforms conventional methods such as fuzzy logic methods in accuracy, despite longer online computation time [17].

3.2.1. Methodology of PSO

PSO's philosophy stems from bird flocks or bee swarms sharing knowledge. Just as bees communicate information about food sources, PSO particles exchange information to optimize a defined cost function. PSO algorithms vary in how knowledge travels between particles. The popular Neighborhood Topologies are star topology, wheel topology, and ring topology, as shown in Figure 2.

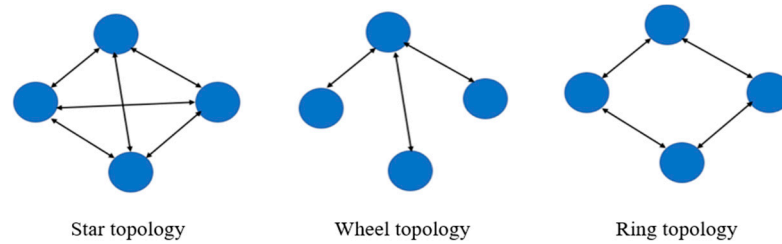


Figure 2. Different types of topologies for the PSO method [25].

In optimization problems, the variable to optimize is crucial. The vector B ($[B_1, B_2, B_3, \dots, B_n]$), also known as the position vector, aims to minimize or maximize the cost function $f(B)$. This function, also called the fitness function, measures the proximity of the end-effector to a designated point or path. During optimization, particles adjust their dynamic traveling velocity based on personal and neighbor experiences.

3.2.2. Algorithm of PSO

The PSO algorithm operates through the following steps:

1. Initialization: Create and uniformly distribute a population of particles over B . Evaluate each particle's position using the fitness function as shown in Equation (19):

$$Z = f(x, y) = \sin x^2 + \sin y^2 + \sin x \sin y \quad (19)$$

2. Update Personal Best: Evaluate positions of vectors B . If a particle's current position is better than its previous best position, update to the new position. Locate the best particle based on its previous best location.
3. Update Velocities: Update all particles' velocities based on Equation (20), considering inertia weight (W), cognitive constant (c_1), social constant (c_2), and random numbers (U_1, U_2).

$$V_i^{t+1} = WV_i^t + c_1 U_1^t (P_{b_i}^t - P_i^t) + c_2 U_2^t (g_b^t - P_i^t) \quad (20)$$

4. Update Positions: Move all particles to their new positions based on Equation (20), considering velocity and current position.
5. Iteration: Repeat steps 2–4 until stopping criteria are met.

This iterative process continues until the stopping criteria are fulfilled, optimizing the objective function (f) over the population of particles (A).

3.3. Integration of PSO and ANN

Previous studies have shown that while PSO can achieve high accuracy results, it suffers from time-consuming issues when applied to real-time problems [20–23]. In contrast, Artificial Neural Networks (ANNs) offer shorter online computation times for solving the IK of RM, albeit with longer offline computation (training) times [12]. However, the accuracy of ANN solutions heavily relies on training databases and is currently inferior to PSO. Hence, a combined approach using both PSO and ANN is proposed to solve the IK of RM in real time, aiming to leverage the strengths of both methods.

3.3.1. Implementation of ANN for Inverse Kinematic Problem

The structure of an Artificial Neural Network (ANN) comprises nodes connected by links, representing neuron relationships. In a feed-forward ANN, there are no feedback links. This

architecture includes an input layer, hidden layers, and an output layer. Neurons within the hidden and output layers are interconnected via adaptive weights, calibrated during training with input–output data. Each neuron has an activation function, typically within the range $[-1, 1]$, with common functions being tangent sigmoid and logarithmic sigmoid. The difference between the desired and actual output is the error, minimized by updating neurons based on predefined rules. To train the ANN, a dataset containing inputs and corresponding outputs is required, often determined experimentally according to the system.

Using ANN to solve IK or RM explicitly is challenging due to the unmatched quantities between inputs and outputs. In the case of a 4-DOF RM, with two inputs and three outputs, multiple solutions exist. Constraints, such as end-effector orientation, are necessary to achieve explicit solutions.

Complex constraints, like singularity and obstacle avoidance, make finding explicit solutions difficult. To address this, the position coordinates of the end-effector and auxiliary virtual functions (constraints) are used to train the ANN. Various input data, including FK and constraint equations, are used to train the ANN system. Different constraints, such as the summation of orientation or joint angles, diversify inputs to the ANN, improving its performance.

The implementation procedure for solving IK of a 4-DOF RM using ANN involves:

1. Initializing joint angles within the permitted range.
2. Using FK equations to obtain end-effector coordinates (X, Y, Z) .
3. Employing random joint angles in constraint equations to obtain input constraints.
4. Setting $[X, Y, Z, C]$ as inputs and $[\theta_1, \theta_2, \theta_3, \theta_4]$ as outputs to the ANN.
5. Designing the ANN structure, including hidden layers, neurons, and training functions.
6. Training, testing, and validating the ANN.
7. Defining a valid target trajectory and preparing input data for the ANN.
8. Applying inputs to the ANN to obtain joint angles as outputs.

Steps 1–4 build a database for ANN inputs and outputs. While time-consuming, this process is necessary, as directly applying FK equations may not ensure precise end-effector coordinates are derived from the selected joint angles.

3.3.2. Combination of PSO and ANN

The combination of PSO and ANN provides a powerful solution for real-time IK of RM. PSO calculates inputs and outputs for the ANN to build a database, facilitating training, testing, and validation. Here is the procedure for building an ANN database using PSO for a 4-DOF RM IK solution (Figure 3a):

1. Initialize random values for each joint angle within the permitted range.
2. Use PSO to obtain the end-effector coordinates (X, Y, Z) as inputs for the ANN.
3. Utilize random joint angles in constraint equations to derive the input constraints for the ANN.
4. Set $[X, Y, Z, C]$ as inputs and $[\theta_1, \theta_2, \theta_3, \theta_4]$ as outputs for the ANN.

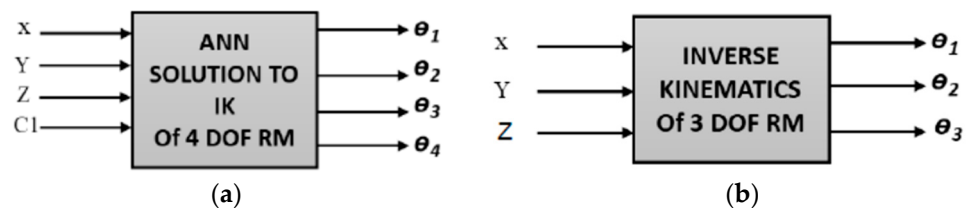


Figure 3. (a) ANN solution to IK of 4-DOF RM [26]; (b) IK solution to 3-DOF manipulator [23].

For a 3-link rescue manipulator (3-DOF) operating in a 3-dimensional space, the IK inputs are the end-effector coordinates $[X, Y, Z]$, and the outputs are the three joint angles. Since the inputs and outputs are equal, auxiliary virtual functions (constraints), used in the 4-link RM IK solution, are unnecessary. Step 3 can be skipped.

4. Results and Analysis

The focus of this project is rescue robot manipulators, which execute various tasks in dangerous environments. The manipulators need to quickly solve problems with minimal error and adapt to new environments. Since these environments are unpredictable and manipulators cannot rely on previous routes, they must solve problems in real time without prior learning. While real-time

solutions require high-tech computing, they offer the advantage of addressing diverse problems with high accuracy.

In this section, the results of solving the IK problem for a 3-DOF manipulator is presented using PSO. Programmed simulations aim to generate optimal paths using PSO in different configurations. Four configurations with various iterations and swarm populations are designed. Simulation results from the first path in each scenario are compared to determine the most effective scenario for PSO. This optimal configuration is then used to find optimal paths for the other scenarios.

Each path is defined in a 3D Cartesian coordinate system and divided into 40 points to make the paths discrete and flexible. When applying PSO for IK, all 40 points are solved simultaneously to ensure accuracy.

Table 1 presents these combinations along with their mean square errors. Configuration 3 exhibits the lowest error compared to the other scenarios. Hence, we will utilize the swarm population and iteration numbers from configuration 3 for subsequent paths.

Table 1. Comparison of different scenarios for path 1 [23].

Configuration	Population of Swarms	Iteration	Run Time	Error (MSE)
1	50	100	1 min, 10 s	0.001
2	75	200	1 min, 40 s	0.0003981
3	100	300	2 min, 10 s	0.0002511
4	150	400	4 min, 37 s	0.0007814

The runtime varies, but the error stays low, which proves that PSO is robust across different paths. Comparing the results of the PSO in configuration 3, where its error was 0.0002511 within 50 iterations in 2 min and 10 s, the ANN method achieved lower accuracy results in shorter runtimes.

Table 2 shows the performance of the inverse kinematics with PSO in different paths under the best configuration, configuration 3. The selected path is in a 2D workspace.

Table 2. Comparison of different scenarios in configuration 3 [23].

Paths	Population of Swarms	Iteration	Run Time	Error (MSE)
1	100	300	2 min, 10 s	0.0002511
2	100	300	17 min, 9 s	0.0004964
3	100	300	6 min, 2 s	0.0007153
4	100	300	9 min, 4 s	0.0009080
5	100	300	18 min, 4 s	0.000811

The proposed PSO-ANN approach was compared with the ANN method. Due to the time-intensive nature of the PSO calculations, only 200 random variables were generated for each joint angle, and these calculations were completed offline. The train-test split ratio was 70/30. Both PSO and ANN contribute to improving result accuracy, enabling the PSO-ANN approach to achieve a lower error compared to PSO alone, by approximately 0.0002511, as shown in Table 3. Moreover, with a database size of only 200, the running time of the ANN can be less than 1 s, demonstrating its efficiency in real-time applications.

Table 3. Comparison of PSO-ANN and ANN [23].

	Iterations	Run Time	Error (MSE)
PSO-ANN	50	2 min, 10 s	0.0002511
ANN	1000	36.10 s	0.045589

Author Contributions: Conceptualization, P.M. and W.P.; methodology, P.M.; software, P.M.; validation, P.M., X.F. and W.P.; resources, P.M.; data curation, P.M.; writing—original draft preparation, P.M.; writing—review and editing, X.F. and W.P.; supervision, W.P.; project administration, W.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: No new data were created.

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

References

1. “Manipulator (Device)”. En.Wikipedia.Org. 2020. Available online: [https://en.wikipedia.org/wiki/Manipulator_\(device\)](https://en.wikipedia.org/wiki/Manipulator_(device)) (accessed on 21 February 2020).
2. Chirikjian, G.S. *Theory and Applications of Hyper-Redundant Robotic Manipulators*; California Institute of Technology: Pasadena, CA, USA, 1992.
3. Sciavicco, L.; Siciliano, B. A solution algorithm to the inverse kinematic problem for redundant manipulators. *IEEE J. Robot. Autom.* **1988**, *4*, 403–410. [\[CrossRef\]](#)
4. Murray, R.M.; Li, Z.; Sastry, S.S. *A Mathematical Introduction to Robotic Manipulation*; CRC Press: Boca Raton, FL, USA, 2017.
5. Wang, L.C.; Chen, C.C. A combined optimization method for solving the inverse kinematics problems of mechanical manipulators. *IEEE Trans. Robot. Autom.* **1991**, *7*, 489–499. [\[CrossRef\]](#)
6. Fu, Z.; Yang, W.; Yang, Z. Solution of inverse kinematics for 6R robot manipulators with offset wrist based on geometric algebra. *J. Mech. Robot.* **2013**, *5*, 031010. [\[CrossRef\]](#) [\[PubMed\]](#)
7. Alavandar, S.; Nigam, M.J. Neuro-fuzzy based approach for inverse kinematics solution of industrial robot manipulators. *Int. J. Comput. Commun. Control* **2008**, *3*, 224–234. [\[CrossRef\]](#)
8. Daya, B.; Khawandi, S.; Akoum, M. Applying neural network architecture for inverse kinematics problem in robotics. *J. Softw. Eng. Appl.* **2010**, *3*, 230. [\[CrossRef\]](#)
9. Dyuksinx, P.; G  radin, M. *An Introduction to Robotics: Mechanical Aspects*; University of Li  ge: Li  ge, Belgium, 2004.
10. G  mez, S.; S  nchez, G.; Zarama, J.; Ramos, M.C.; Alc  ntar, J.E.; Torres, J.; N    ez, A.; Santana, S.; N    era, F.; Lopez, J.A. Design of a 4-DOF robot manipulator with optimized algorithm for inverse kinematics. *Int. J. Mech. Mechatron. Eng.* **2015**, *9*, 929–934.
11. Aghajarian, M.; Kiani, K. Inverse kinematics solution of PUMA 560 robot arm using ANFIS. In Proceedings of the 2011 8th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI), Incheon, Republic of Korea, 23–26 November 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 574–578.
12. Tejomurtula, S.; Kak, S. Inverse kinematics in robotics using neural networks. *Inf. Sci.* **1999**, *116*, 147–164. [\[CrossRef\]](#)
13. Ming, Y.; Guizhang, L.; Jiangeng, L. An inverse kinematics solution for manipulators based on fuzzy logic. In Proceedings of the 2001 International Conferences on Info-Tech and Info-Net. Proceedings (Cat. No. 01EX479), Beijing, China, 29 October–1 November 2001; IEEE: Piscataway, NJ, USA, 2001; Volume 4, pp. 400–404.
14. Batista, J.; Souza, D.; Dos Reis, L.; Barbosa, A.; Ara  jo, R. Dynamic model and inverse kinematic identification of a 3-DOF manipulator using RLSPSO. *Sensors* **2020**, *20*, 416. [\[CrossRef\]](#) [\[PubMed\]](#)
15. Zhao, J.; Xu, T.; Fang, Q.; Xie, Y.; Zhu, Y. A synthetic inverse kinematic algorithm for 7-DOF redundant manipulator. In Proceedings of the 2018 IEEE International Conference on Real-Time Computing and Robotics (RCAR), Kandima, Maldives, 1–5 August 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 112–117.
16. Dereli, S.; K  ker, R. Calculation of the inverse kinematics solution of the 7-DOF redundant robot manipulator by the firefly algorithm and statistical analysis of the results in terms of speed and accuracy. *Inverse Probl. Sci. Eng.* **2020**, *28*, 601–613. [\[CrossRef\]](#)
17. Song, R.; Chen, X.; Shen, C.; Zhang, H. Modeling FOG drift using back-propagation neural network optimized by artificial fish swarm algorithm. *J. Sens.* **2014**, *2014*, 273043. [\[CrossRef\]](#)
18. Oh, S.Y.; Orin, D.; Bach, M. An inverse kinematic solution for kinematically redundant robot manipulators. *J. Robot. Syst.* **1984**, *1*, 235–249. [\[CrossRef\]](#)
19. Zhao, J.; Zhao, L.; Liu, H. Motion planning of hyper-redundant manipulators based on ant colony optimization. In Proceedings of the 2016 IEEE International Conference on Robotics and Biomimetics (ROBIO), Qingdao, China, 3–7 December 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 1250–1255.
20. Liu, Y.; Xi, J.; Bai, H.; Wang, Z.; Sun, L. A general robot inverse kinematics solution method based on improved PSO algorithm. *IEEE Access* **2021**, *9*, 32341–32350.
21. Starke, S.; Hendrich, N.; Magg, S.; Zhang, J. An efficient hybridization of genetic algorithms and particle swarm optimization for inverse kinematics. In Proceedings of the 2016 IEEE International Conference on Robotics and Biomimetics (ROBIO), Qingdao, China, 3–7 December 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 1782–1789.
22. Dereli, S.; K  ker, R. IW-PSO approach to the inverse kinematics problem solution of a 7-DOF serial robot manipulator. *Sigma J. Eng. Nat. Sci.* **2018**, *36*, 77–85.
23. Monfred, P. Computation of Inverse Kinematics of Redundant Manipulator Using Particle Swarm Optimization Algorithm and Its Combination with Artificial Neural Networks. Master’s Thesis, University of Regina, Regina, SK, Canada, 2023.
24. Mohammed, A.A.; Sunar, M. Kinematics modeling of a 4-DOF robotic arm. In Proceedings of the 2015 International Conference on Control, Automation and Robotics, Singapore, 20–22 May 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 87–91.

25. Available online: <https://www.analyticsvidhya.com/blog/2021/10/an-introduction-to-particle-swarm-optimization-algorithm/> (accessed on 23 September 2024).
26. Habibkhah, S. Artificial Neural Networks for the Computation of the Inverse Kinematics of Redundant Manipulators. Master's Thesis, University of Regina, Regina, SK, Canada, 2020.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.