robotics

MDPI

Article

Inverse Kinematic Control of a Delta Robot Using Neural Networks in Real-Time

Akram Gholami, Taymaz Homayouni, Reza Ehsani * and Jian-Qiao Sun

Department of Mechanical Engineering, University of California, Merced, CA 95343, USA; agholamipareh@ucmerced.edu (A.G.); thomayouni@ucmerced.edu (T.H.); jsun3@ucmerced.edu (J.-Q.S.)
* Correspondence; rehsani@ucmerced.edu

Abstract: This paper presents an inverse kinematic controller using neural networks for trajectory controlling of a delta robot in real-time. The developed control scheme is purely data-driven and does not require prior knowledge of the delta robot kinematics. Moreover, it can adapt to the changes in the kinematics of the robot. For developing the controller, the kinematic model of the delta robot is estimated by using neural networks. Then, the trained neural networks are configured as a controller in the system. The parameters of the neural networks are updated while the robot follows a path to adaptively compensate for modeling uncertainties and external disturbances of the control system. One of the main contributions of this paper is to show that updating the parameters of neural networks offers a smaller tracking error in inverse kinematic control of a delta robot with consideration of joint backlash. Different simulations and experiments are conducted to verify the proposed controller. The results show that in the presence of external disturbance, the error in trajectory tracking is bounded, and the negative effect of joint backlash in trajectory tracking is reduced. The developed method provides a new approach to the inverse kinematic control of a delta robot.

Keywords: inverse kinematic control; trajectory tracking; neural networks; delta robot

Citation: Gholami, A.; Homayouni, T.; Ehsani, R.; Sun, J.-Q. Inverse Kinematic Control of a Delta Robot Using Neural Networks in Real-Time. *Robotics* **2021**, *10*, 115. https://doi.org/10.3390/robotics10040115

Academic Editor: Xinjun Liu

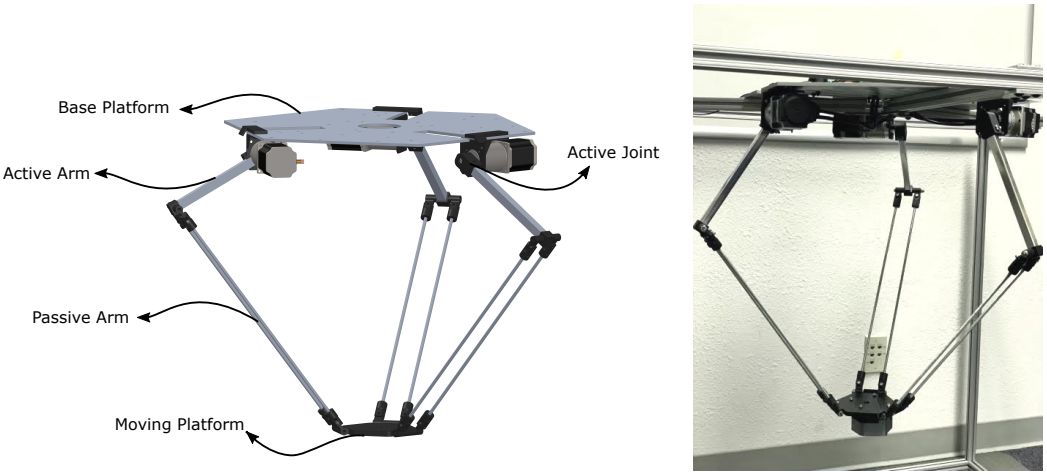
Received: 19 August 2021
Accepted: 13 October 2021
Published: 16 October 2021

1. Introduction

The study of robot kinematics and dynamics is crucial for robot design, control, and simulation; however, kinematic and dynamic modeling can be computationally expensive and time consuming [1,2]. The kinematic model of a robot can be derived analytically based on the physics and structure of the robot. The mathematical process for computing the joint coordinates for a given set of end-effector coordinates is called inverse kinematics and involves solving a set of non-linear and complex equations [3].

Recent studies on neural networks and deep learning have proven that robotic control can benefit from this method, and neural networks can replace the complicated mathematical modeling of some systems [2,4–6]. Neural networks can learn from data. They can identify non-linearity between input and output data, which makes them good candidates for estimating the mathematical modeling of a system [7]. For instance, Yang et al. [8] successfully estimated the inverse kinematics of a redundant seven degree of freedom (DOF) manipulator based on a radial basis function neural network. In another study, Zhou et al. [9] developed a neural network method to estimate the kinematic pose from an image with prior knowledge of the geometric model of the object.

A delta robot is a parallel manipulator with three DOF in which the base platform of the robot is connected to a moving platform by three parallel identical kinematic chains, as shown in Figure 1. Each chain consists of an arm connected by a revolute joint to a four-bar parallelogram link. A rotary motor actuates the arm of the delta robot on the base platform. The four-bar parallelogram links restrict the movement of the moving platform along the three Cartesian axes without rotation [10–15].



(**a**) Schematic view (**b**) Mechanical structure

Figure 1. Delta robot.

Over the years, many controllers have been developed to control delta robots. Le and Kang [16] implemented an adaptive tracking controller for parallel robots. By utilizing a fully tuned radial basis function in the controller system, they aimed to compensate for the uncertainties in modeling a parallel robot and external disturbances in the control system. However, a dynamics model of the robot is required to develop the controller. Hernandez et al. [17] applied a Proportional-Derivative (PD) controller with a feed-forward artificial B-Spline neural network (BSNN). The main advantage of this controller is that it can work with little knowledge of the dynamics parameters of the robot. Xu and Li [18] estimated the forward kinematics of a parallel manipulator with neural networks. Neural networks are utilized as an observer to approximate the current position of the end-effector in the control algorithm. Uzunovic et al. [19] investigated the control of a parallel robot with neural networks-based inverse kinematics. They used the neural networks for kinematics estimation and converted the desired trajectory to the desired actuated joints angle. The estimated actuated joints angles were used as a reference input to the controller system. Mohammed and Li [20] proposed using neural networks to solve the optimization problem in kinematic control of the parallel Stewart platform. Zhang et al. [21] proposed a fuzzy control method for delta robot motion control in which the prior knowledge of the robot is not required.

The accuracy of controllers developed based on the mathematical models depends on the accuracy of these models and the sensors in the control loop [22]. However, the mathematical modeling of a robot can be different from the actual modeling after manufacturing and assembly or because of joint wear and backlash after long operation [19].

This project aims to design a control system that does not depend on the kinematic and dynamic modeling of the delta robot and can adapt to the changes in the system. To achieve this goal, first, the kinematics of the delta robot is estimated by using neural networks. Then, the neural networks are configured in our system as a controller to achieve optimal trajectory tracking. Our specific contributions include the following:

- Development of a neural network model that learns the kinematics of a delta robot;
- Demonstration of the effect of updating the parameters of neural networks in inverse kinematics control of a delta robot with consideration of joint backlash.

The remainder of this paper is organized as follows. Previous related work is presented in Section 2. Section 3 describes the architecture of the neural networks in order to approximate the kinematic modeling of the delta robot, and an overview of the configuration of the neural networks as a controller in the system is given. The simulation and experimental results of the proposed method for the different trajectories tracking are presented in Section 4, and the feasibility of the method is verified in the presence

of external disturbance. Moreover, in Section 4.3, the developed method is compared with recently published algorithms. Finally, in Section 5, we present our conclusions and future work.

2. Related Work

In recent years, many studies have focused on using neural networks to solve the kinematics of robots [7,23–25]. These research studies intended to estimate the kinematics of different serial manipulators. Based on the studies, this method can be applied to any general serial manipulator. Moreover, reasonable accuracy can be achieved along any desired path. As an instance, Aggarwal et al. [23] used neural networks to find the inverse kinematics solution of a PUMA 560 robot. They reported the error percentage of 4.930, 7.292, and 3.738 along the $X-$, $Y-$, and $Z-$ coordinates. In a different study, Almusawi et al. [25] showed that including the current position of the end-effector in the inverse kinematics estimation of a Denso VP6242 robot reduces the error percentage to less than 0.5 in Cartesian space.

Thus, it would be of interest to investigate if neural networks can be employed to approximate the kinematics of a parallel manipulator. Furthermore, it is still unclear how well this idea can be generalized to a system in the presence of disturbance and uncertainty. In addition, the capability of neural networks in reducing the negative impact of joints backlash in a system is still unknown.

3. Material and Methods

Radial Function Basis (RFB) and Multilayer Perceptron (MLP) are the two main neural networks used in robot model estimation. Despite the greater accuracy of the RFB in kinematics estimation, RFB is more computationally expensive [26]. Training an offline MLP neural network is more straightforward; however, the trained neural networks parameters cannot be adjusted after implementation onto the manipulator. Therefore, online training is needed in order to adjust to the changes in the robot model. [27].

3.1. Delta Robot Kinematics Estimation Using Neural Networks

In many applications, the robot moves on a predefined trajectory along sequential points. Including the current position of the robot in neural networks has beneficial outcomes in the estimation of joint angles to move to the next desired position [25]. To approximate the kinematic modeling of the delta robot, sufficient data for representing the workspace of the robot (a total of 2000 pairs of data) were collected based on randomly moving the arms, recording the joints, and moving platform positions. The neural network is trained with a Levenberg–Marquardt backpropagation, a popular curve-fitting algorithm. The inputs to the neural networks are the current position of the moving platform $P[k] = [X_k, Y_k, Z_k]^T$ together with the current angular position of each arm $\theta[k] = [\theta_{1k}, \theta_{2k}, \theta_{3k}]^T$ and the position of the moving platform at next time step $P[k+1] = [X_{k+1}, Y_{k+1}, Z_{k+1}]^T$. The output of the neural networks is the estimated change in angular position of each arm $\Delta\widehat{\theta}[k] = [\Delta\theta_1, \Delta\theta_2, \Delta\theta_3]^T$. In other words, the neural networks predict the input to the delta robot in order to move the end-effector to the desired point. The output of the neural networks can be represented by Equation (1):

$$\Delta\widehat{\theta}[k] = f(P[k], P[k+1], \theta[k]) \tag{1}$$

where k represents the time step. The neural network model is trained to minimize the mean squared error of the neural network prediction versus the true data, as shown in Equation (2):

$$J = \frac{1}{n}\sum_{j=1}^{n}(\Delta\theta[j] - \Delta\widehat{\theta}[j])^2 \tag{2}$$

where $\Delta\theta$ is the actual data, and $\Delta\widehat{\theta}$ is the neural networks output.

The grid search algorithm is used to find the optimum number of hidden layers and the neurons of the hidden layers and the activation function of the neural networks. While straightforward, this technique is relatively inefficient since many networks need to be trained before an acceptable one is found [28]. The neural networks parameters are selected based on having the lowest mean square error while keeping the training time less than a second to ensure real-time control is achievable. The grid search algorithm is performed in two steps. First, the search algorithm takes the below parameters as inputs:

- Number of hidden layers: [1, 2, 3];
- Number of neurons in each layer: $[1, 2, 4, 8, 16, 32, 64, 128]$;
- Activation functions: {Sigmoid, Hyperbolic tangent, Softmax, Linear}.

A total of 20 networks were trained for each combination of inputs in the grid search. The neural networks are trained in the Matlab environment using a deep learning Toolbox. The average of neural networks mean square error and the average of neural network training time for each combination are calculated. A neural network with three hidden layers of 8, 16, and 16 neurons, with sigmoid as an activation function, provides the lowest mean square error with minimum training time, where the mean square error is 0.7007 and the time of training is 0.5497 s.

For the second step, a fine search around the selected number of neurons in each layer is conducted. In this case, a neural network (Figure 2) with an input layer of 9 neurons, three hidden layers of 8, 20, and 15 neurons respectively, with sigmoid as an activation function, and an output layer of 3 linear neurons is selected. The mean square error is 0.5658, and the time of training is 0.6893 seconds.

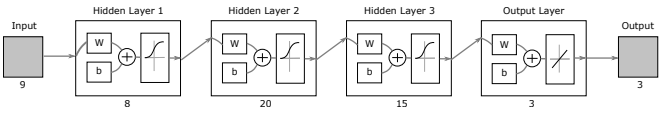


Figure 2. The structure of the neural networks in delta robot kinematics estimation.

3.2. Structure of Control System

The trained neural network is then implemented to predict the actuated joint angles of the robot in real-time. As shown in Figure 3, the block diagram of the robot trajectory tracking control consists of a trained neural network block, an online neural network training block, a driving unit, and the robot. The input to the neural network is the desired trajectory and the feedback of the current active joint angles and the current position of the moving platform. Take into account that the desired trajectory is acting as the future position input for the neural network. In the online neural network training block, while the robot is working, at each time step, the angular position of the arms and the moving platform position are sampled online. As a new data point is sampled, it is inserted in the initial 2000 data sets collected for neural network training. Adding more data points increases the size of the training set; thus, the time of training increases as well. Hence, it is necessary to remove another data point from the initial data set. Adding and removing data points in the data set can result in a biased training set for the neural network. To prevent this, the initial data set, which represents the delta robot workspace, is divided into several segments. As a new data point is sampled, based on the current position of the end-effector (or the first three features of the data point), the new sampled data point is inserted in the corresponding segment, and we randomly deleted another data point from that segment. After collecting ten new data points, the neural network will be retrained, and the neural network weights and biases in the trained neural network block are updated.

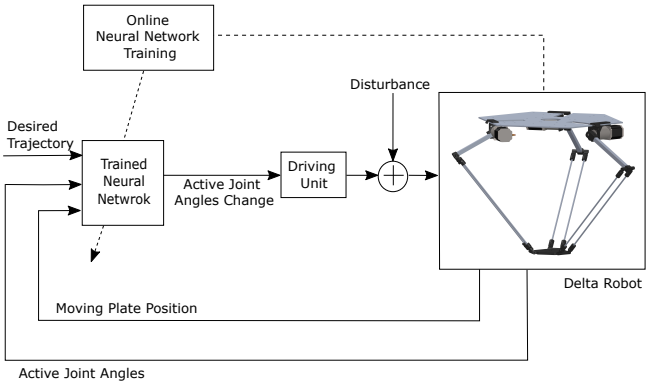


Figure 3. Block diagram of trajectory tracking control of delta robot using a neural network in real-time.

We divided the workspace of the delta robot into nine segments (Figure 4). The workspace is split using the three symmetry planes of the workspace and two equally spaced planes in $Z-$ direction. Dividing the workspace into very small segments increases the processing time of data replacement.

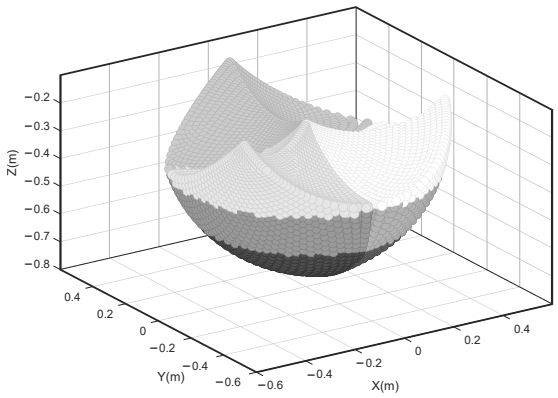


Figure 4. The nine segmentations of the delta robot's workspace. Each color represents one segment.

To accelerate the retraining process, the initial weights and biases in the neural network training will be set to the previous weights and biases; therefore, the weights and biases will be optimized faster, making the algorithm practical for real-time training processes. Toshani and Farrokhi [29] employed this algorithm for updating the parameters of neural networks in real-time in their proposed method as well. Furthermore, they ranked this algorithm as a low computational time technique. The overview of the algorithm for the neural network online training is shown in Algorithm 1.

4. Simulation and Experimental Results

Experimental and simulation results are presented and discussed in this section to validate the effectiveness of the proposed inverse kinematic controller on a delta robot with the parameters summarized in Table 1. Simulation tests are performed in MATLAB/Simulink environment by using the Simscape Multibody Toolbox. An external disturbance, i.e., white noise with the power of noise samples equal to 0.01 is added to the system to evaluate the robustness of the algorithm. The experimental tests are carried out on the delta robot (Figure 1a) actuated by geared Nema 23 closed-loop stepper motors with holding torque of 30 Nm. The VL53L1X, a Time-of-Flight (ToF) laser-ranging sensor, is used to measure the position of the moving platform in Cartesian space. Three laser-ranging sensors are attached to the end-effector facing the $X-$, $Y-$, and $Z-$ directions. Each sensor measures the distance to the boards attached to the frame of the delta robot.

Algorithm 1: Overview of the algorithm for the neural network online training

```
input  : Neural network weights and biases trained offline
output : New weights and biases
while robot is running do
    for i ← 1 to 10 do
        Collect moving platform position;
        Collect active points angles;
        Find the segment which the data point belongs to;
        Replace new data in the corresponding segment randomly;
        Save the new data set;
    end
    Initialize the neural network weights and biases;
    Retrain the neural network with new data set;
    Save the new weights and biases;
end
```

Table 1. Parameters of the delta robot.

Link	Dimension (m)
Active arm length	0.25
Passive arm length	0.5
Base platform radius	0.225
Moving platform radius	0.075

Two different paths are defined to show and verify the performance of the proposed neural network in inverse kinematic modeling and trajectory tracking. For the first reference trajectory, a spiral is defined by the following equations (Equation (3)):

$$\begin{cases} X = 0.2\cos(\alpha) \\ Y = 0.2\sin(\alpha) \\ Z = (-0.2\alpha)/(4\pi - 0.4) \end{cases} \tag{3}$$

where α varies between 0 and 4π. The delta robot starts from a steady state with a random initial position.

For the second trajectory, we evaluate a non-smooth trajectory tracking using a square with an edge length of 0.4 m and $Z = -0.5 + 0.1\cos(\alpha)$, where α varies between 0 and 2π. Following a square is different from tracking a smooth circular motion because the desired square path is non-smooth at the four corners, which challenges the proposed controller system on its real-time performance.

The accuracy of the controller is measured by calculating the Euclidean distance error, as represented in Equation (4):

$$e = \sqrt{e_X^2 + e_Y^2 + e_Z^2} \tag{4}$$

where e_X, e_Y, and e_Z are the error in $X-$, $Y-$, and $Z-$ directions, respectively.

4.1. Joint Backlash and Kinematics Estimation Performance

In order to evaluate the performance of estimated delta robot kinematics, the neural network trained to estimate the kinematics of the robot in Section 3.1 is used as the controller in trajectory tracking. The parameters of the neural network do not change during the path following. Furthermore, a joint backlash of 2 degrees is added to one of the delta robot active joints to examine its effect on estimated delta robot kinematics. The joint backlash is modeled in the simulation by using the backlash block in Simulink.

Figure 5 shows the Euclidean distance error in the spiral tracking simulation. It is evident that, if the joint backlash is not considered in the structure of the robot, the neural network trained to estimate the kinematics of the delta robot, works fine in trajectory tracking, and the tracking error is bounded. By using the same control configuration, the tracking error is higher when the joint backlash of two degrees is added to the delta robot. This is expected because the neural network is trained with no backlash in the system.

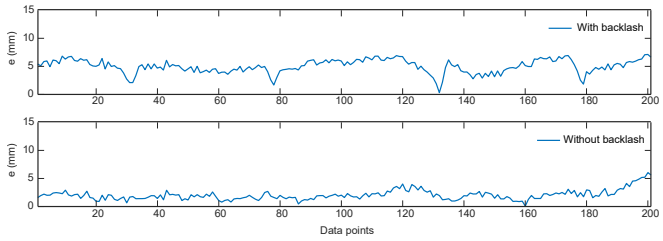


Figure 5. Simulation result of trajectory tracking errors for a spiral path using a neural network.

Figure 6 demonstrates the Euclidean distance error in square trajectory tracking. Similarly to the spiral path following, it can be observed that the neural network trained to estimate the kinematics of the delta robot works more precisely when the joint backlash is not in the system. The peak spotted in tracking error could be because enough data points are not collected in that part of the working space for training the neural network kinematics estimator. Therefore, collecting more data might be helpful to reduce the overshoot in trajectory tracking.

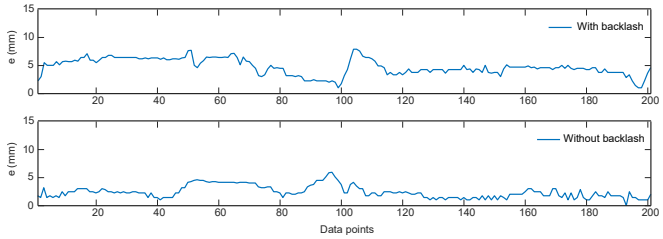


Figure 6. Simulation result of trajectory tracking errors for a square path using a neural network.

4.2. Tracking Performance

This part presents the simulation and experimental results of the neural network online training algorithm during path following and when adapting to the joint backlash.

The results of the spiral tracking are illustrated in Figure 7. Figure 8 shows the simulation and experimental tracking error in $X-$, $Y-$, and $Z-$ directions and the Euclidean distance error.

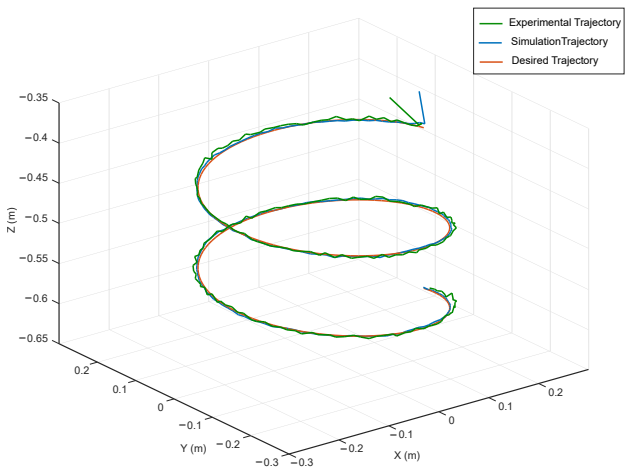
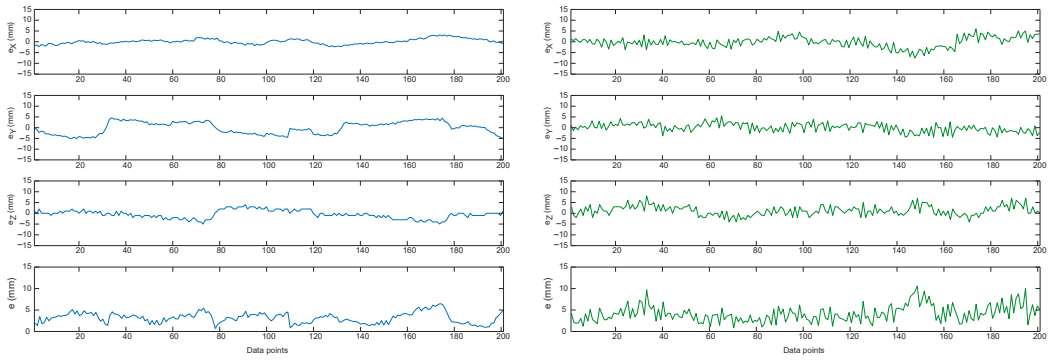


Figure 7. Tracking of a spiral path using a neural network in real-time.

In Figure 8, it is observed that the position errors in all directions are restricted to a small vicinity of the origin. The maximum Euclidean distance errors in the simulation and the experiment are 6.79 mm and 10.56 mm, respectively. The reason behind this difference is that, in the experimental setup, multiple joints backlash might exist. The peaks observed in the figure could be because the neural network is not trained very well during the online training, but this is temporary. From the presented result, it can be observed that the proposed controller achieves better tracking performance compared to using a neural network with fixed weights and biases.



(**a**) Simulation results. (**b**) Experimental results.

Figure 8. Trajectory tracking errors of the moving platform in a spiral path using a neural network in real-time.

The square paths following outcomes are demonstrated in Figure 9. It can be observed that the proposed control system does not have any problem following the edges.

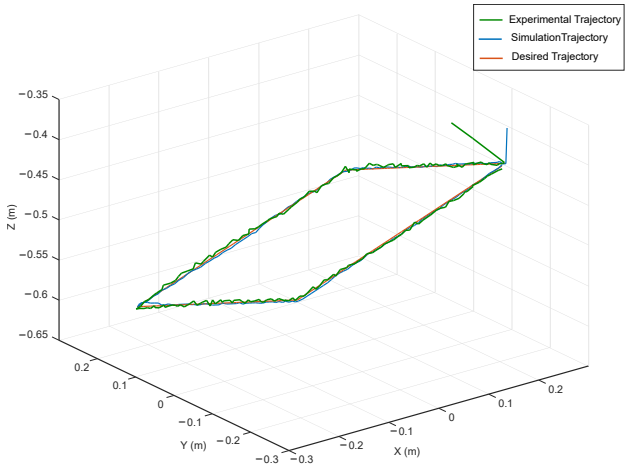
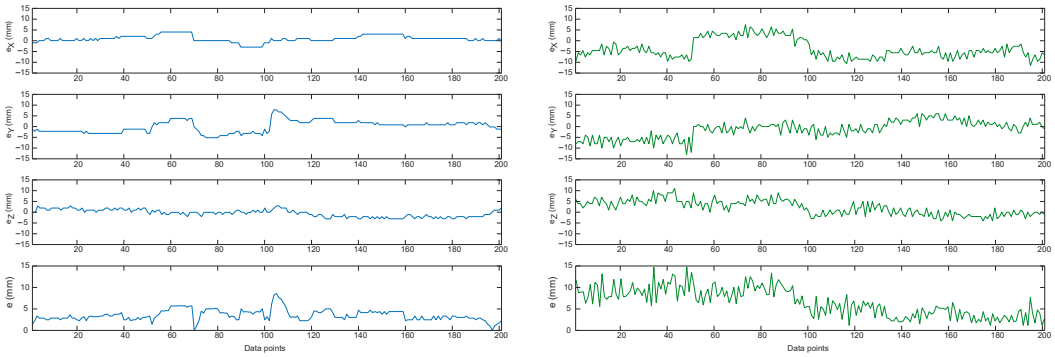


Figure 9. Tracking of a square path using a neural network in real-time.

As represented in Figure 10a, the precision of the trajectory tracking has been improved compared to using a neural network without updating in Figure 6. Similarly to the spiral path investigation, the position errors in all directions are bounded. Maximum Euclidean distance errors are 8.54 mm and 15 mm in simulation and experiment, respectively (Figure 10). The peaks in the tracking error figure could be because the parameters of the neural network are not optimized very well during online training.



(**a**) Simulation results. (**b**) Experimental results.

Figure 10. Trajectory tracking errors of the moving platform in a square path using a neural network in real-time.

In Table 2, the averages of the absolute tracking error in $X-$, $Y-$, and $Z-$ directions and the Euclidean distance error during the entire path are shown. It can be concluded that the act of collecting data and updating the neural network weights and biases has reduced the average of the absolute tracking error for each trajectory tracking when the joint backlash is added to the delta robot. By observing the result of several simulations and experiments, it can be argued that the overall performance of retraining the neural network during the process can enhance the tracking performance.

Table 2. Average of the absolute tracking error.

		$e_x(mm)$	$e_y(mm)$	$e_z(mm)$	$e(mm)$
Spiral path	Without backlash	1.19	0.99	1.08	2.18
	With backlash	1.78	4.17	2.08	5.08
	With backlash and NN update (simulation)	0.98	2.55	1.58	3.48
	With backlash and NN update (experiment)	1.96	1.68	2.54	4.16
Square path	Without backlash	1.34	1.31	0.76	2.40
	With backlash	1.34	3.63	1.83	4.71
	With backlash and NN update (simulation)	1.27	2.38	1.42	3.50
	With backlash and NN update (experiment)	3.43	3.47	3.25	6.79

NN: Neural Networks

If the neural network output sign is the opposite of the correct movement of active joints, it may render the system locally unstable. In the closed-loop system, any significant oscillation cannot be observed; therefore, it can be argued that the neural network has been trained well with the cost function less than 0.56, and it works well in the proposed configuration.

4.3. Comparison

In Table 3, a comparison between the proposed approach and the recently published methods is shown to help demonstrate the contribution of the proposed inverse kinematics control scheme. Zhou et al. [30] developed an adaptive position-force control using recurrent neural networks for redundant manipulators with uncertainties. In this approach, the kinematics modeling of the robot is needed, but the pseudo-inversion calculation is eliminated. Furthermore, the method works well with uncertainty in the modeling and environment. Li et al. [31] designed a dual neural network for manipulator control. The developed control scheme inherits all the dual neural network features. Moreover, it does not need model information and has tolerance to uncertainty. In another study, a recurrent neural network for manipulator control is proposed [32]. This research mainly focuses on using neural networks to reduce the effect of noise in the control system. In [33],

a model-free dual neural network for controlling the end-effector of a parallel robot for trajectory tracking is established. In this approach, the unknown deterministic time-varying parameters of the system can be learned.

Table 3. Comparison among different methods for the control of manipulators.

	This Paper	[30]	[31]	[32]	[33]
Model information	No	Yes	No	Yes	No
Handling model uncertainty	Yes	Yes	Yes	No	Yes
Handling system change	Yes	Yes	-	No	No
Handling noise	Yes	-	Yes	Yes	Yes
Number of adjustable parameters	L	M	L	H	M
Complexity	L	M	M	M	L
Level of solving problem	P	F	V	V	V

L: Low, M: Medium, H: High, P: Position, F: Force, V: Velocity.

5. Conclusions

This paper proposes inverse kinematic control for a delta robot using a neural network algorithm in real-time. The neural network is first trained offline and based on no prior knowledge of the delta robot to approximate the kinematics of the robot. Then, the developed neural network is used as a controller to control the position of the moving platform. As the robot works, a new data stream is used to retrain the neural networks in order to adapt to the changes in the system. An advantage of this system is that it is not necessary to obtain knowledge about the dimension or the structure of the parallel manipulator. Moreover, the trained neural network parameters can be adjusted after implementation onto the manipulator, which helps adjusting to the robot kinematics changes. The simulation and experimental results show satisfactory trajectory tracking with joints backlash consideration in the delta robot. Moreover, it is evident that the developed system is robust in the presence of disturbance and noise. For future work, we intend to use the proposed inverse kinematics controller in a strawberry harvesting robot. We expect that this algorithm will be successful in controlling the delta robot in pick-up and place applications.

Author Contributions: Conceptualization, A.G. and R.E.; funding acquisition, R.E.; investigation, A.G. and T.H.; methodology, A.G. and J.-Q.S.; resources, A.G.; software, A.G.; supervision, R.E.; validation, A.G. and T.H.; visualization, A.G.; writing—original draft preparation, A.G.; writing—review and editing, T.H., R.E., and J.-Q.S. All authors have read and agreed to the published version of the manuscript.

Funding: This material is based upon work supported by the National Science Foundation under Grant No. 924662.

Institutional Review Board Statement: Not applicable

Informed Consent Statement: Not applicable.

Data Availability Statement: Data sharing is not applicable to this article.

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

References

1. Bingul, Z.; Karahan, O. Dynamic Modeling and Simulation of Stewart Platform. In *Serial and Parallel Robot Manipulators*; Kucuk, S., Ed.; IntechOpen: Rijeka, Croatia, 2012; Chapter 2.
2. Ren, H.; Ben-Tzvi, P. Learning inverse kinematics and dynamics of a robotic manipulator using generative adversarial networks. *Robot. Auton. Syst.* **2020**, *124*, 103386.
3. Laribi, M.A.; Romdhane, L.; Zeghloul, S. Analysis and dimensional synthesis of the DELTA robot for a prescribed workspace. *Mech. Mach. Theory* **2007**, *42*, 859–870.

4. Jiang, Y.; Yang, C.; Na, J.; Li, G.; Li, Y.; Zhong, J. A Brief Review of Neural Networks Based Learning and Control and Their Applications for Robots. *Complexity* **2017**, *2017*, 1895897.
5. Sánchez-Sánchez, C.; Izzo, D. Real-time optimal control via Deep Neural Networks: Study on landing problems. *arXiv* **2016**, arXiv:1610.08668.
6. Toquica, J.S.; Oliveira, P.S.; Souza, W.S.; Motta, J.M.S.; Borges, D.L. An analytical and a Deep Learning model for solving the inverse kinematic problem of an industrial parallel robot. *Comput. Ind. Eng.* **2021**, *151*, 106682.
7. Duka, A.V. Neural Network based Inverse Kinematics Solution for Trajectory Tracking of a Robotic Arm. *Procedia Technol.* **2014**, *12*, 20–27.
8. Yang, Y.; Peng, G.; Wang, Y.; Zhang, H. A New Solution for Inverse Kinematics of 7-DOF Manipulator Based on Neural Network. In Proceedings of the 2007 IEEE International Conference on Automation and Logistics, Jinan, China , 18–21 August 2007; pp. 1958–1962.
9. Zhou, X.; Sun, X.; Zhang, W.; Liang, S.; Wei, Y. Deep Kinematic Pose Regression. In *Computer Vision—ECCV 2016 Workshops*; Hua, G., Jégou, H., Eds.; Springer International Publishing: Cham, Switzerland, 2016; pp. 186–201.
10. Brinker, J.; Corves, B.; Takeda, Y. Kinematic and Dynamic Dimensional Synthesis of Extended Delta Parallel Robots. In *Robotics and Mechatronics*; Springer International Publishing: Cham, Switzerland, 2019; pp. 131–143.
11. Nguyen, V.L.; Lin, C.Y.; Kuo, C.H. Gravity compensation design of Delta parallel robots using gear-spring modules. *Mech. Mach. Theory* **2020**, *154*, 104046.
12. Lopez, M.; del Castillo, E.; García, G.R.; Bashir, A. Delta robot: Inverse, direct, and intermediate Jacobians. *Proc. Inst. Mech. Eng. Part C J. Mech. Eng. Sci.* **2006**, *220*, 103–109.
13. Brinker, J.; Corves, B. A Survey on Parallel Robots with Delta-like Architecture. In Proceedings of the 14th IFToMM World Congress, Taipei, Taiwan, 25–30 October 2015.
14. Mottola, G.; Gosselin, C.; Carricato, M. Dynamically feasible motions of a class of purely-translational cable-suspended parallel robots. *Mech. Mach. Theory* **2019**, *132*, 193–206.
15. Tang, L.; Shi, P.; Wu, L.; Wu, X.; Tang, X. Singularity Analysis on a Special Class of Cable-Suspended Parallel Mechanisms With Pairwise Cable Arrangement and Actuation Redundancy. *J. Mech. Des.* **2019**, *142*, 024501.
16. Le, T.D.; Kang, H.J. An adaptive tracking controller for parallel robotic manipulators based on fully tuned radial basic function networks. *Neurocomputing* **2014**, *137*, 12–23.
17. Escorcia-Hernández, J.M.; Aguilar-Sierra, H.; Aguilar-Mejía, O.; Chemori, A.; Arroyo-Núñez, J.H. An Intelligent Compensation Through B-Spline Neural Network for a Delta Parallel Robot. In Proceedings of the 2019 6th International Conference on Control, Decision and Information Technologies (CoDIT), Paris, France, 23–26 April 2019; pp. 361–366.
18. Xu, Q.; Li, Y. A 3-PRS Parallel Manipulator Control Based on Neural Network. In *Advances in Neural Networks—ISNN 2007*; Liu, D., Fei, S., Hou, Z.G., Zhang, H., Sun, C., Eds.; Springer Berlin Heidelberg: Berlin/Heidelberg, Germany, 2007; pp. 757–766.
19. Uzunovic, T.; Golubovic, E.; Baran, E.A.; Sabanovic, A. Configuration space control of a parallel Delta robot with a neural network based inverse kinematics. In Proceedings of the 2013 8th International Conference on Electrical and Electronics Engineering (ELECO), Bursa, Turkey, 8–30 November 2013; pp. 497–501.
20. Mohammed, A.M.; Li, S. Dynamic Neural Networks for Kinematic Redundancy Resolution of Parallel Stewart Platforms. *IEEE Trans. Cybern.* **2016**, *46*, 1538–1550.
21. Zhang, J.; Lian, C.; Gao, R.; Shi, L. 3-Degree-of-Freedom Parallel Robot Control Based Fuzzy Theory. In Proceedings of the 2010 Second International Conference on Intelligent Human-Machine Systems and Cybernetics, Nanjing, China, 26–28 August 2010; Volume 1, pp. 221–224.
22. On-line regression algorithms for learning mechanical models of robots: A survey. *Robot. Auton. Syst.* **2011**, *59*, 1115–1129.
23. Aggarwal, L.; Aggarwal, K.; Urbanic, R.J. Use of Artificial Neural Networks for the Development of an Inverse Kinematic Solution and Visual Identification of Singularity Zone(s). *Procedia CIRP* **2014**, *17*, 812–817.
24. Hasan, A.T.; Ismail, N.; Hamouda, A.M.; Aris, I.; Marhaban, M.H.; Al-Assadi, H.M. Artificial neural network-based kinematics Jacobian solution for serial manipulator passing through singular configurations. *Adv. Eng. Softw.* **2010**, *41*, 359–367.
25. Almusawi, A.R.J.; Dülger, L.C.; Kapucu, S. A New Artificial Neural Network Approach in Solving Inverse Kinematics of Robotic Arm (Denso VP6242). *Comput. Intell. Neurosci.* **2016**, *2016*, 5720163.
26. Zubizarreta, A.; Larrea, M.; Irigoyen, E.; Cabanes, I.; Portillo, E. Real time direct kinematic problem computation of the 3PRS robot using neural networks. *Neurocomputing* **2018**, *271*, 104–114.
27. Jin, L.; Li, S.; Yu, J.; He, J. Robot manipulator control using neural networks: A survey. *Neurocomputing* **2018**, *285*, 23–34.
28. Doukim, C.A.; Dargham, J.A.; Chekima, A. Finding the number of hidden neurons for an MLP neural network using coarse to fine search technique. In Proceedings of the 10th International Conference on Information Science, Signal Processing and their Applications (ISSPA 2010), Kuala Lumpur, Malaysia, 10–13 May 2010; pp. 606–609.
29. Toshani, H.; Farrokhi, M. Real-time inverse kinematics of redundant manipulators using neural networks and quadratic programming: A Lyapunov-based approach. *Robot. Auton. Syst.* **2014**, *62*, 766–781.
30. Zhou, X.; Xu, Z.; Li, S.; Wu, H.; Cheng, T.; Lv, X., RNN Based Adaptive Compliance Control for Robots with Model Uncertainties. In *AI Based Robot Safe Learning and Control*; Springer Singapore: Singapore, 2020; pp. 39–61.

31. Li, S.; Shao, Z.; Guan, Y. A Dynamic Neural Network Approach for Efficient Control of Manipulators. *IEEE Trans. Syst. Man Cybern. Syst.* **2019**, *49*, 932–941.
32. Li, S.; Wang, H.; Rafique, M.U. A Novel Recurrent Neural Network for Manipulator Control With Improved Noise Tolerance. *IEEE Trans. Neural Netw. Learn. Syst.* **2018**, *29*, 1908–1918, doi:10.1109/TNNLS.2017.2672989.
33. Mirza, M.A.; Li, S.; Jin, L. Simultaneous learning and control of parallel Stewart platforms with unknown parameters. *Neurocomputing* **2017**, *266*, 114–122.