Solving Inverse Kinematics of a 7-DOF Manipulator Using Convolutional Neural Network

Hassan Ashraf Elkholy[1,2(✉)], Abdalla Saber Shahin[1,2(✉)], Abdelaziz Wasfy Shaarawy[2(✉)], Hagar Marzouk[1,2(✉)], and Mahmoud Elsamanty[1,2,3(✉)]

[1] Smart Engineering Systems Research Center (SESC), Nile University, Shaikh Zayed City 12588, Giza, Egypt
{h.ashraf,a.saber,h.marzouk,melsamanty}@nu.edu.eg
[2] School of Engineering and Applied Sciences, Nile University Campus, Sheikh Zayed District, Juhayna Square, 6th of October City 12588, Giza, Egypt
a.wasfy@nu.edu.eg
[3] Faculty of Engineering at Shoubra, Benha University, Cairo, Egypt
mahmoud.alsamanty@feng.bu.edu.eg

Abstract. This paper presents a way to solve inverse kinematics of a 7-DOF manipulator using artificial neural networks. The manipulator consists of a 6-DOF articulated arm installed on a linear guide system to increase the workspace of the robot. The purpose of this paper is to provide an alternative to the traditional and complicated way to solve inverse kinematics by using artificial neural networks. The training data is generated from MATLAB after obtaining the DH parameters and workspace of the manipulator. Then, it was fed to the convolutional neural architecture to obtain a model for the manipulator. The input of the CNN is the end effector desired pose, and the outputs are the position angles of each joint. Two different architectures of artificial neural networks are compared to decide the most efficient architecture that produces the most accurate descriptive model of the manipulator.

Keyword: 7-DOF manipulator, Inverse kinematics, Convolutional Neural Network (CNN), Artificial Neural Network (ANN)

1 Introduction and Literature Survey

Nowadays, serial robotic manipulators are being used in many applications especially in the industrial sector. In general, they consist of several rigid links that are connected together by the means of rotational or prismatic joints. Increasing the number of degrees of freedom (DOF) results in more motion flexibility and more complex inverse kinematics problem (IK) [3]. The traditional method of controlling serial robotic manipulators is done by sending the desired coordinates of the end effector to the controller unit after solving its IK problem. For higher

A.-E. Hassanien et al. (Eds.): AICV 2020, AISC 1153, pp. 343–352, 2020.
https://doi.org/10.1007/978-3-030-44289-7_32

order of DOF, the Jacobean inverse kinematics algorithm can be used [13,14]. However, IK problem is difficult to be solved analytically because the solution will have high redundancy and many solutions [7]. Nowadays, Artificial Neural Network (ANN) is commonly used to solve the IK problems with higher number of DOF because of its advantages such as the ability of self-learning as well as parallel processing [16]. It is also able to fit non-linear relationships between inputs and outputs with a very good accuracy [5]. Neural networks depend on the input-output relationship specified by the training data, unlike traditional methods of control [3]. This paper is related to previous work in the aspect of using ANN to solve the IK problem of the manipulator [5–9,12]. According to Almusawi et al. [1], a new artificial neural network was implemented on a 6-DOF Denso robotic manipulator to increase the ability of the neural network to estimate the output joint angles [1]. Moreover, a study proposed multi-layered perceptron neural network architecture that has six sub-neural networks and back-propagation algorithm applied on 2-DOF manipulator [4]. In this study, a neural network was used to solve the problem of matrix inversion by iterating on the joint position directly [4]. Furthermore, according to Toshani et al. [13], a neural network based on a numerical method and one based on ANN were used to solve the inverse kinematics problem of a 7-DOF robot manipulator. The aim of this paper is to provides an ANN to solve the IK of a proposed 7-DOF manipulator. Figure 1 shows the whole stages that this paper went through. First, the 3D model of the proposed robotic arm was designed on SolidWorks software, then the DH parameters were taken and fed into the kinematics equations of this robotic arm to get the workspace which was validated using Moveit environment and SolidWorks. Afterwards, these obtained data of the workspace was fed into two different types of ANN on matlab. The paper is organized as the following: kinematic model of the roboticator, data obtaining and processing, artificial neural network architecture, simulation results and discussion, CNN validation, and conclusion.

2 Kinematics Model of the 7-DOF Manipulator

The system used in the paper is a 7-DOF manipulator consisting of six revolute joints mounted on a prismatic joint as shown in Fig. 2. The DH parameters table relates each joint with its previous joint to finally come up with the relation between the world frame and the end effector frame [11]. This is done by multiplying the 4×4 transformation matrices of the seven joints which yields very complex equations. One single transformation matrix of these equations is shown in Eq. (1).

$$T_i = \begin{bmatrix} cos(\Theta_i) & -sin(\Theta_i) \times cos(\alpha_i) & sin(\Theta_i) \times sin(\alpha_i) & ai \times cos(\Theta_i) \\ sin(\Theta_i) & cos(\Theta_i) \times cos(\alpha_i) & -cos(\Theta_i) \times sin(\alpha_i) & ai \times sin(\Theta_i) \\ 0 & sin(\alpha_i) & cos(\alpha_i) & di \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

where i ranges between 1 to 7. In order to obtain the DH parameters shown in Table 1 accurately, joint limits were defined and assigned according to its own

hardware and mechanical limitations. To validate these DH parameters, a model of the robot was created in MATLAB using the robotics toolbox as shown in Fig. 3(a). The workspace was validated by applying random angle joints of the workspace in the CAD model in SolidWorks and comparing the pose of the manipulator in the two models. This comparison resulted in a 0.01 mm error in the manipulator's pose between the two models.

Robot Operating System (ROS) is a widely used framework for robotics motion planning and inverse kinematics solving [10]. Moveit is a navigation software in ROS to enhance the process of trajectory planning and help creating a collision-free environment [2]. The 7-DOF manipulator is parsed in moveit environment to further validate the manipulator's workspace generated by MATLAB as shown in Fig. 3(b). In order to make the robot ready to be used in moveit environment, a Unified Robot Description Format (URDF) of the robot in its home position was created [10] as shown in Fig. 2. Figure 4 shows the real hardware of the 7-DOF serial manipulator. Figure 5 illustrates the front and side views of the workspace of the robot.

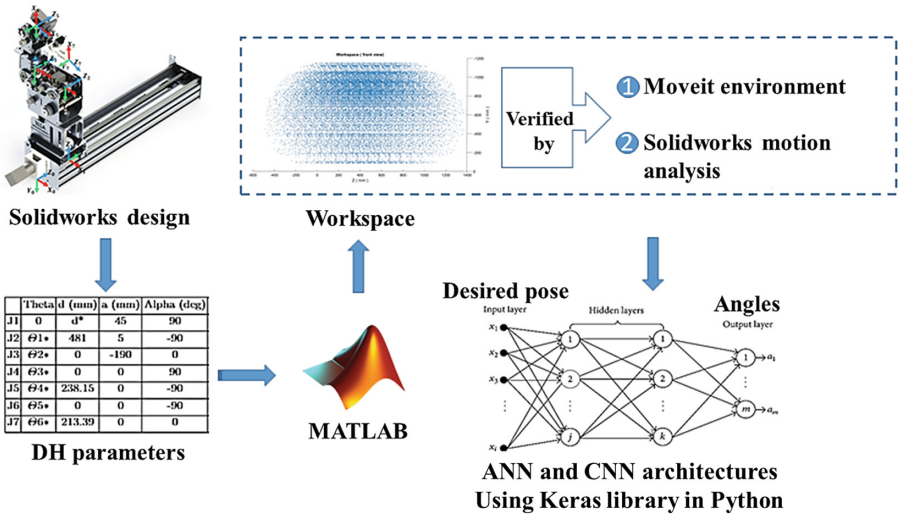
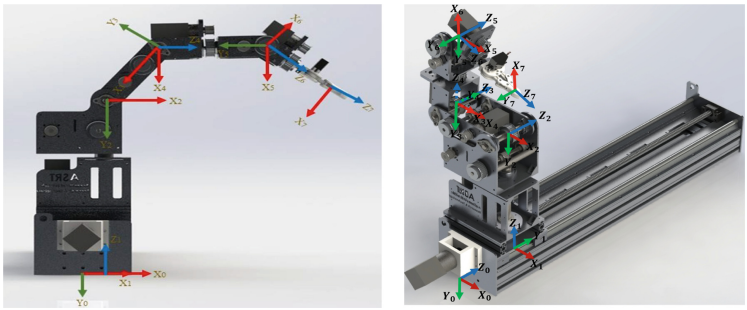
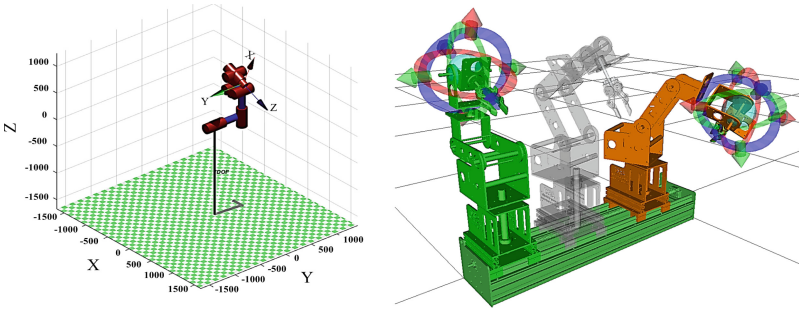


Fig. 1. Process of collecting training data



(a) Side view of the manipulator (b) Isometric view of the manipulator in home position

Fig. 2. 7-DOF serial manipulator with joints frames



(a) MATLAB model describing the direction of rotation of each joint (b) Motion Analysis of the manipulator on moveit

Fig. 3. Matlab model and moveit analysis

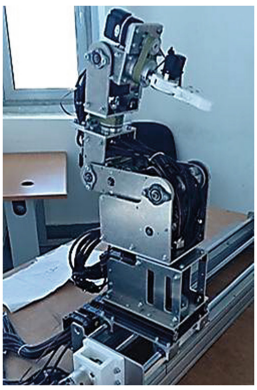
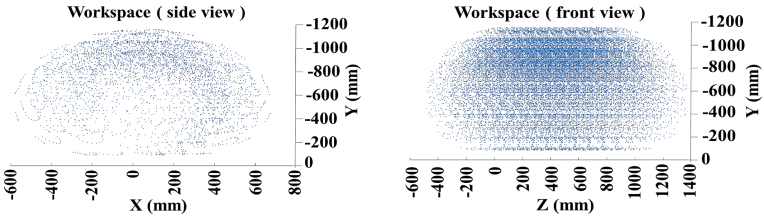


Fig. 4. The real hardware



(a) Workspace side view (b) Workspace front view

Fig. 5. Workspace of the manipulator

Table 1. The DH parameters of the manipulator

	Theta	d (mm)	a (mm)	Alpha (deg)	Joints in home position
J1	0	d*	45	90	97.5 mm
J2	$\Theta 1*$	481	5	−90	0°
J3	$\Theta 2*$	0	−190	0	−0.4°
J4	$\Theta 3*$	0	0	90	4.36°
J5	$\Theta 4*$	238.15	0	−90	0°
J6	$\Theta 5*$	0	0	−90	−70.44°
J7	$\Theta 6*$	213.39	0	0	0°

3 Data Obtaining and Processing

After obtaining and validating the data from Matlab, it was fed into the neural network to start training the model. Range of data values was large; thus, it was scaled down to produce data values that are close to each other, so the weights of the artificial neural network are adjusted quickly, and the model converges fast. Two types of scaling were used: normalization and standardization. Sometimes normalization or standardization of data is required to make the training process much easier [15]. Standardization of the data produced more accurate model than normalization shown in Eq. (2), and so was preferred upon normalization. Only the output data was standardized as shown in Eq. (3):

$$Normalized\ Data = \frac{Data - Min}{Max - Min} \tag{2}$$

$$Standardized\ Data = \frac{Data - Mean}{Standard\ deviation} \tag{3}$$

4 Artificial Neural Network Architecture

Two different types of ANN architectures are used in this paper. The first architecture is a Multi-Layer Perceptron (MLP) ANN with 7 hidden layers and "linear" activation function. The second one consists of the same previous architecture with replacing the first two layers with two 1D CNN layers. Each ANN

architecture is evaluated in two aspects: loss and accuracy. The architecture that reaches minimum loss and maximum accuracy is chosen to be the best neural architecture that is capable of generating the most descriptive model for the manipulator. The two architectures were constructed using keras a neural network library in python. About 70% of the data obtained was fed to each neural net, while 15% were used as validation data, and other 15% were used as testing data. Figure 6 illustrates the architecture of the neural network with CNN and the number of neurons in each layer. The activation functions in all layers were "Tanh", the optimizer function was 'Adam' and the loss function was mean square error. The learning rate during the training was 1×10^{-3}.

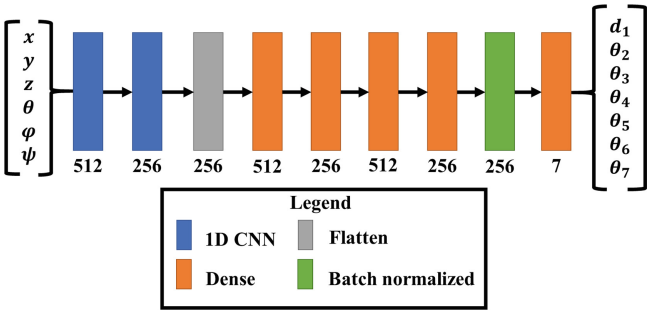


Fig. 6. CNN architecture

5 Simulation Results and Discussion

As mentioned before, the accuracy of the CNN with and without data scaling is illustrated in Fig. 7 where two methods of data scaling were used: normalization and standardization. The accuracy of the standardized training data was found to be 92.16% and that of the normalized training data was 91.68%. Whereas the accuracy of non-scaled training data was only 85%. The network that produces the best model for the serial manipulator is shown in Fig. 8. The results of this ANN is compared with results of another ANN that consists only of MLP 2NN with 7 hidden layers. The accuracy and loss of the training data in the CNN-based network are 92.16% and 0.346 respectively. However, they were 91.6% and 0.027 respectively in the MLP-based network as shown in Fig. 9. Both models were generated after 100 epochs with a batch size of 10000 and a learning rate of 1×10^{-3}.

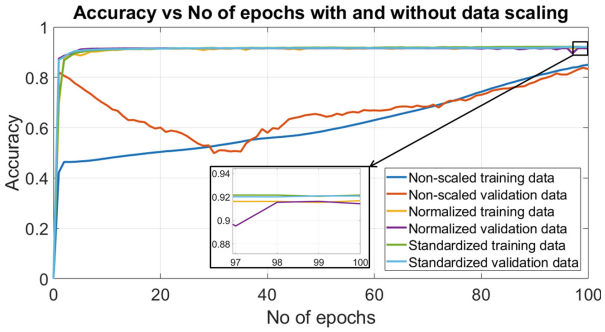
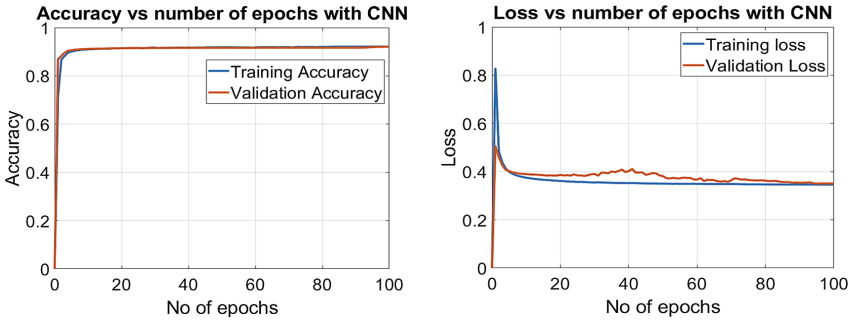


Fig. 7. Comparison between results of data with and without scaling



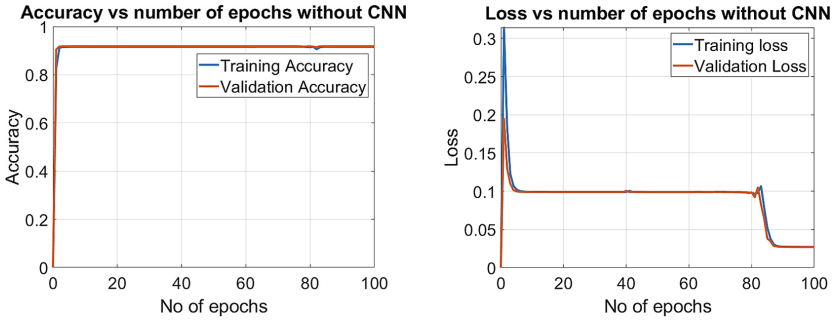
(a) Accuracy with CNN vs number of epochs

(b) loss with CNN vs number of epochs

Fig. 8. Accuracy and loss of the neural network with CNN

6 CNN Validation

To validate the accuracy of the predicted data from the CNN, a path of 4 points was fed to the CNN as the desired poses to obtain the required angles. The predicted angles were fed to the forward kinematics model in MATLAB to visualize the position of the end effector. Afterwards, a comparison between the generated positions and the desired positions is done as shown in Fig. 10. According to Fig. 10, the two paths are compared in four-point path, and the error between each two points in the x-axis and y-axis is shown in Fig. 11. The values of the error range between 14.8 mm to 89.3 mm in the x-axis, and 7 mm to 75 mm in the y-axis proving the 92.16% accuracy of the network.



(a) Accuracy without CNN vs number of epochs

(b) loss without CNN vs number of epochs

Fig. 9. Accuracy and loss of the neural network without CNN

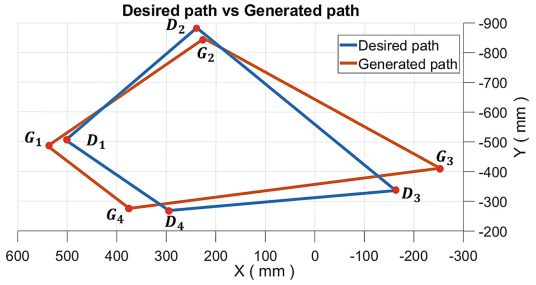


Fig. 10. Desired path vs generated path from the CNN

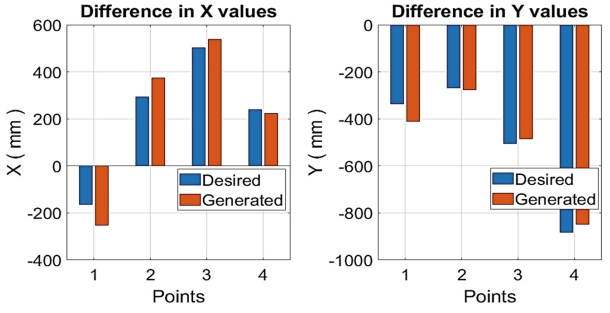


Fig. 11. Error values between the desired path and generated path

7 Conclusion

In this paper, the kinematic analysis of a 7-DOF manipulator, consisting of one prismatic and six revolute joints, using artificial neural networks is presented. The manipulator is designed on SolidWorks and simulated in moveit environment to extract the data of its workspace. The data is reformed and manipulated to

be fed into the neural network architecture to be trained to produce a strong model to be later used in order to control the manipulator. Therefore, various combinations of neural network architectures were evaluated and compared. The most accurate architecture combined CNN connected to traditional ANN with dense layers, and it produced an accuracy of 92.16%. For future work, more complex ANN, that combines different neural architectures, will be used with much more layers and neurons. After obtaining the model from the artificial neural network, it will be used to control the real hardware system.

Acknowledgment. The authors would like to thank Eng. Bahaa Ashraf, Eng. Ahmed Tarek, Eng. Mahmoud Bakr, Eng. Lotfy Monir from Nile University for their design of the 7-DOF serial manipulator, and Eng. MennaAllah Soliman, teaching assistant in Mechatronics Engineering department at Nile University, for her great help and support in this work.

References

1. 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**, 1–10 (2016). https://doi.org/10.1155/2016/5720163
2. Chitta, S., Sucan, I., Cousins, S.: Moveit! [ros topics]. IEEE Robot. Autom. Mag. **19**(1), 18–19 (2012). https://doi.org/10.1109/mra.2011.2181749
3. Dash, K., Choudhury, B., Khuntia, A., Biswal, B.: A neural network based inverse kinematic problem. In: 2011 IEEE Recent Advances in Intelligent Computational Systems (2011). https://doi.org/10.1109/raics.2011.6069357
4. Daya, B., Khawandi, S., Akoum, M.: Applying neural network architecture for inverse kinematics problem in robotics. J. Softw. Eng. Appl. **03**(03), 230–239 (2010). https://doi.org/10.4236/jsea.2010.33028
5. Duka, A.V.: Neural network based inverse kinematics solution for trajectory tracking of a robotic arm. Procedia Technol. **12**, 20–27 (2014). https://doi.org/10.1016/j.protcy.2013.12.451
6. Farzan, S., DeSouza, G.N.: From D-H to inverse kinematics: a fast numerical solution for general robotic manipulators using parallel processing. In: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (2013). https://doi.org/10.1109/iros.2013.6696709
7. Hasan, A.T., Hamouda, A., Ismail, N., Al-Assadi, H.: An adaptive-learning algorithm to solve the inverse kinematics problem of a 6 D.O.F serial robot manipulator. Adv. Eng. Softw. **37**(7), 432–438 (2006). https://doi.org/10.1016/j.advengsoft.2005.09.010
8. Köker, R.: A genetic algorithm approach to a neural-network-based inverse kinematics solution of robotic manipulators based on error minimization. Inf. Sci. **222**, 528–543 (2013). https://doi.org/10.1016/j.ins.2012.07.051
9. Nagata, F., Kishimoto, S., Kurita, S., Otsuka, A., Watanabe, K.: Neural network-based inverse kinematics for an industrial robot and its learning method. In: The Proceedings of the 4th International Conference on Industrial Application Engineering 2016 (2016). https://doi.org/10.12792/iciae2016.015
10. Qian, W., Xia, Z., Xiong, J., Gan, Y., Guo, Y., Weng, S., Deng, H., Hu, Y., Zhang, J.: Manipulation task simulation using ROS and Gazebo. In: 2014 IEEE International Conference on Robotics and Biomimetics (ROBIO 2014) (2014). https://doi.org/10.1109/robio.2014.7090732

11. Singh, A., Singla, A., Soni, S.: D-H parameters augmented with dummy frames for serial manipulators containing spatial links. In: The 23rd IEEE International Symposium on Robot and Human Interactive Communication (2014). https://doi.org/10.1109/roman.2014.6926379
12. Srisuk, P., Sento, A., Kitjaidure, Y.: Inverse kinematics solution using neural networks from forward kinematics equations. In: 2017 9th International Conference on Knowledge and Smart Technology (KST) (2017). https://doi.org/10.1109/kst.2017.7886084
13. Toshani, H., Farrokhi, M.: Kinematic control of a seven DOF robot manipulator with joint limits and obstacle avoidance using neural networks. In: The 2nd International Conference on Control, Instrumentation and Automation (2011). https://doi.org/10.1109/icciautom.2011.6356794
14. Wang, J., Li, Y., Zhao, X.: Inverse kinematics and control of a 7-DOF redundant manipulator based on the closed-loop algorithm. Int. J. Adv. Robot. Syst. **7**(4), 37 (2010). https://doi.org/10.5772/10495
15. Wilamowski, B.: Neural network architectures and learning algorithms. IEEE Ind. Electron. Mag. **3**(4), 56–63 (2009). https://doi.org/10.1109/mie.2009.934790
16. Zhou, Y., Tang, W., Zhang, J.: Algorithm for multi-joint redundant robot inverse kinematics based on the Bayesian - BP neural network. In: 2008 International Conference on Intelligent Computation Technology and Automation (ICICTA) (2008). https://doi.org/10.1109/icicta.2008.406