Received: June 16, 2023. Revised: July 26, 2023.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

An Improved Performance of the Designed Robotic Motion Control for NAO Robot Arms Using Hybrid Neural Network-Jacobian

Arif Nugroho[1] **Eko Mulyanto Yuniarno[1,2]** **Mauridhi Hery Purnomo[1,2,3]***

[1]*Department of Electrical Engineering, Institut Teknologi Sepuluh Nopember (ITS), Surabaya, Indonesia*
[2]*Department of Computer Engineering, Institut Teknologi Sepuluh Nopember (ITS), Surabaya, Indonesia*
[3]*University Center of Excellence on Artificial Intelligence for Healthcare and Society (UCE AIHeS), Indonesia*
* Corresponding author's Email: hery@ee.its.ac.id

Abstract: Inverse kinematics plays a significant role in the robotic motion control. The flexible way to formulate the inverse kinematics can be achieved by using a neural network. The drawback of the neural network-based inverse kinematics is that it has no feedback mechanism to compensate for the remaining error. To improve its performance, further development should be conducted. In the design of the robotic motion control for the NAO robot arms, this paper proposed a new approach that combines the neural network-based inverse kinematics with the Jacobian. This combination yields a closed-loop control system. This control system utilizes the neural network as the feedforward controller and the Jacobian as the feedback controller. In particular, the neural network-based inverse kinematics has a function to estimate a set of required joint angles for the joint actuators, and the Jacobian function is to compensate for the remaining error of the neural network-based inverse kinematics in the estimation of joint angles. By using this proposed approach, we obtained more accurate joint angles for controlling the joint actuators. The comparison result showed that the averaged MSE for the particle swarm optimization (PSO) was 3.47 x 10^{-3} rad, 1.19 x 10^{-3} rad for the neural network, and 3.72 x 10^{-5} rad for the proposed approach. The performance comparison result indicated that our proposed approach has a lower averaged MSE than the other ones. Accordingly, the result of this research confirmed that our proposed approach can provide more accurate joint angles for controlling the joint actuators such that the robot's end-effector can be driven along the desired path in the cartesian space.

Keywords: Robotic motion control, NAO robot arms, Inverse kinematics, Neural network, Jacobian, Closed-loop control system.

1. Introduction

The articulated robots have received widespread attention in modern robot applications. One reason is that the articulated robots contain more degrees of freedom (DOF) than any other robot such that they have more flexibility in the movement. The presence of multiple joints in the articulated robots produces a number of degrees of freedom. The other reason for using the articulated robots is due to their versatility. The articulated robots have been extensively applied for several industrial applications, such as welding [1], painting [2], and assembling [3]. It is important to note, the articulated robots are not limited only to industrial robots but also to other robots that contain multiple joints, including the bio-inspired robot [4], surgery robot [5], humanoid robot [6], and so on.

NAO robot is the kind of articulated robot that physically resembles the human body structure due to containing a head, two arms, and two legs. This robot contains multiple revolute joints, all of which are represented by servo motors. These servo motors are so-called joint actuators. Physically, this robot is available in two models: NAO H25 and NAO H21. The main difference between the two is the number of joints and the type of end-effector on their arms. For each arm, the NAO H25 has five joints and a prehensile hand as the end-effector, while the NAO H21 has four joints and a non-prehensile hand as the end-effector [7, 8]. Accordingly, the robotic arms of the NAO H25 have more degrees of freedom than

those of the NAO H21. For the other body parts, the number of joints and the type of end-effector are the same. For that reason, this research will be focused on the robotic arms of the NAO H25.

To design the robotic motion control, we require to formulate robot kinematics. The robot kinematics relates to the transformation from the joint space to the cartesian space and vice versa [9]. Related to the robot kinematics, there are two kinematics models: forward and inverse kinematics [10]. The forward kinematics is concerned with the mapping from the joint space, where the control action is executed, to the cartesian space, which is the robot's workspace. The forward kinematics is specifically utilized to find the robot's end-effector position and orientation based on a specific set of joint variables. In revolute joints, the joint variables are expressed in the joint angles. In contrast to the forward kinematics, the inverse kinematics deals with the mapping from the cartesian space to the joint space. The inverse kinematics has a function to determine a set of joint angles based on the given position and orientation of the robot's end-effector. The joint angles resulted from the inverse kinematics are required to control the joint actuators. The inverse kinematics has high nonlinearity, so it is more complex than the forward kinematics.

In spite of its complexity, the inverse kinematics has an important role in the robotic motion control. It should be noted that most tasks to be performed by robotic arms are specified in the cartesian space, whereas the control action is carried out in the joint space. That is why we need the inverse kinematics. For instance, in the manipulation task [11, 12], the robotic arms are supposed to drive their end-effector to attain the desired objects and then move them to another location in the cartesian space. In this task, the desired targets are defined in the cartesian space, and the inverse kinematics is required to find a set of appropriate joint angles to control the joint actuators so that the robot's end-effector can be directed to the desired targets. In addition to the manipulation task, path tracking is another task that requires the inverse kinematics role. In the task of path tracking [13, 14], the robotic arms should be able to move their end-effector along a predetermined path in the cartesian space. This path is typically formed by a sequence of desired target points in the cartesian space. In this task, the inverse kinematics is also utilized to figure out the required joint angles for controlling the joint actuators in order that the robot's end-effector can be driven to follow the desired path in the cartesian space. Based on the applications, it is clear that the inverse kinematics plays a vital role in the planning and execution of robotic motions. The analytical and computational methods are the two techniques that can be applied to solve the inverse kinematics [15].

The analytical method is principally divided into geometric and algebraic methods [16]. In this study [17], the geometric method was proposed as means of obtaining the inverse kinematics solution of the NAO robot arms. Each of the robotic arms consists of four revolute joints. The way to obtain the inverse kinematics solution is initiated by the decomposition of the spatial geometry of the NAO robot arms into multiple plane geometries. From these decomposed plane geometries, the trigonometric equations can be formulated to figure out the joint angles for the joint actuators. This method is typically used for simple robot structures, such as the planar robots whose joint axes are parallel [18, 19] and the spatial robots with a few connected links and joints [17]. Hence, the research [20, 21] proposed the algebraic method for solving the inverse kinematics of the NAO robot arms, each of which has five revolute joints. In this case, the process of solving the inverse kinematics is started with the formulation of forward kinematics. The result of forward kinematics is a transformation matrix whose elements are trigonometric equations. The inverse kinematics solution can be obtained by extracting and simplifying trigonometric equations. In the other research [22, 23], to avoid the difficulty in solving inverse kinematics, they did not involve all the joints and preferred to sacrifice one or more joints in the robot. Unfortunately, this can reduce the number of degrees of freedom in the robot.

Even though the analytical methods have been widely applied to solve the inverse kinematics, there are remaining problems with the analytical methods. One such remaining problem is multiplicity. This is because the analytical methods can produce multiple solutions in terms of joint angles. The system should be able to choose a single correct one for each joint. The mistake of choosing the correct joint angles for the joints causes the robot unable to perform desired motions. In addition to the multiplicity problem, the analytical methods have strict requirements related to the robot structure, so they are only applicable to a certain robot structure. In accordance with Pieper's criterion [24], the inverse kinematics solution can be achieved by the analytical methods when the robotic arms have a spherical wrist whose three consecutive revolute joint axes intersect at the same point. Most industrial robotic arms, such as Denso VP6242 [25], Kuka KR Agilus [26], and Motoman GP180 [27], conform to that criterion. However, the structure of the NAO robot arms does not conform to Pieper's criterion. To meet the criterion, the research [20, 21] modified the kinematic chain in the NAO robot arms by means of shifting the last joint to the preceding

joint position without any change to their physical hardware. Otherwise, the analytical solution for the inverse kinematics of the NAO robot arms is very difficult to achieve.

In comparison with the analytical methods, the computational methods are more flexible for solving the inverse kinematics problem because they are not dependent on robot structures. One computational method to solve the inverse kinematics is Jacobian. In this method, the relation between the joint space and cartesian space is represented by the Jacobian matrix [28]. The inverse kinematics solution can be then obtained by the inverse of the Jacobian matrix whose operation is conducted until the loss function approximates zero. Some previous research used the Jacobian to solve the inverse kinematics of the NAO robot arms [29-31]. Besides, the alternative method for solving the inverse kinematics is metaheuristic algorithms. Some metaheuristic algorithms, such as the artificial bee colony [32], and firefly algorithm [33], particle swarm optimization [34-36], have been applied for solving the inverse kinematics problem. These algorithms require some random initialization to guide their search to obtain an optimal solution. The optimal solution for the inverse kinematics can be achieved by minimizing their objective function. Similar to the previous one, the process of obtaining the inverse kinematics solution is iteratively carried out until their objective function is close to zero. In addition to the aforementioned methods, the inverse kinematics problem can be also solved by the neural network. The way to build the neural network-based inverse kinematics is by the training process, which is the process of teaching the neural network with the dataset. The neural network is such a data-driven modeling technique that it is flexible for modeling the inverse kinematics. Because of its flexibility and learning ability, the neural network can handle the problems of the inverse kinematics, starting from the simple robots [37, 38] to the robots with complex structures [39-41]. The inverse kinematics solution resulted from the neural network is expressed in the neural network architecture that defines the mapping from the cartesian space to the joint space.

However, the computational methods mentioned above have the remaining problems. In the Jacobian, the inverse kinematics solution can be only achieved when the Jacobian matrix is invertible. The Jacobian matrix is invertible when it is a square matrix whose determinant is not zero. The research conducted by [29-31] used the pseudoinverse technique as means of obtaining the inverse kinematics solution, but this technique has no mechanism to handle the problem of singularity. In the singular state, the inverse of the Jacobian matrix does not exist and thus the obtained inverse kinematics solution becomes unacceptable. In this state, the robot can lose one or more degrees of freedom in the workspace [42]. Consequently, the robot is not able to move its end-effector to a certain direction in the cartesian workspace. Meanwhile, the drawback of the metaheuristic algorithms deals with their vulnerability to the starting point. Note that the metaheuristic algorithms are stochastic optimization that requires some random initializations to obtain an optimal solution. It means that their convergence to the optimal solution highly depends on the initial guess and if the initial guess is not appropriate, these algorithms may not find the best solution. Moreover, these metaheuristic algorithms are also vulnerable to premature convergence, which is a state when their objective function becomes trapped in local minima [43]. Because of this condition, the obtained inverse kinematics solution is not acceptable. Different from the previous methods, the neural network can learn from the data through the training process. Once the training is done, the trained neural network requires no iterative processes to find the inverse kinematics solution. In addition, the problem such as singularity and multiplicity does not exist in the neural network. That is why, in the design of robotic motion control, many researchers [37-41] preferred to use the neural network. It is important to note, the neural network-based inverse kinematics structure is feedforward so it is classified as the open-loop control system. The shortcoming of this control system has no feedback mechanism to compensate for the remaining error.

In this paper, we focus on the design of robotic motion control for the NAO robot arms. The main contributions of this paper are listed as follows:

(1) The robotic motion control that we designed in this research adheres to the closed-loop control system,
(2) In the design of the robotic motion control, we proposed a novel approach by using the neural network-based inverse kinematics combined with the Jacobian,
(3) The forward kinematics of the NAO robot arms that we formulated in this research can be used as means of creating the dataset for training the neural network-based inverse kinematics,
(4) The neural network-based inverse kinematics that acts as the feedforward controller can yield a set of joint angles for the joint actuators, while the Jacobian that acts as the feedback controller can helpfully decrease the remaining error of the neural network-based inverse kinematics,
(5) Our designed robotic motion control can result in more accurate joint angles for controlling the joint actuators such that the robot's end-effector can be directed to the desired targets.

Table 1. The joint operational ranges

No	Joint Name	Range (rad)
1	LShoulder Pitch	-2.0857 to 2.0857
2	LShoulder Roll	-0.3142 to 1.3265
3	LElbow Yaw	-2.0857 to 2.0857
4	LElbow Roll	-1.5446 to -0.0349
5	LWrist Yaw	-1.8238 to 1.8238
6	RShoulder Pitch	-2.0857 to 2.0857
7	RShoulder Roll	-1.3265 to 0.3142
8	RElbow Yaw	-2.0857 to 2.0857
9	RElbow Roll	0.0349 to 1.5446
10	RWrist Yaw	-1.8238 to 1.8238

The rest of this paper is organized as follows: section 2 provides the forward kinematics model of the NAO robot arms. Section 3 proposes the design of robotic motion control. The results and discussion are presented in section 4, and section 5 contains the conclusion and direction for future research.

2. Forward kinematics of NAO robot arms

In this section, the discussion will be focused on forward kinematics. The forward kinematics studies the transformation from the joint space, in which the control action is carried out, to the cartesian space, which is the robot's workspace. The joint actuators that operate in the joint space can be controlled to move the robot's end-effector in the cartesian space. As shown in Table 1, the joint actuators in the NAO robot arms have different operational ranges, all of which operate in radians [7]. By controlling the joint actuators, the robot's end-effector can be directed to various locations in the cartesian space. In addition to being controllable, these joint actuators are also equipped with embedded sensors to read their joint angular positions. By using the forward kinematics, the information about the joint angular positions can be then utilized to find out the robot's end-effector locations in the cartesian space [44]. Furthermore, the forward kinematics is commonly utilized as the prerequisite in solving the inverse kinematics. The forward kinematics plays a crucial role, so it must be formulated correctly.

The forward kinematics is typically represented by a single transformation matrix. It can be obtained by multiplying a set of transformation matrices, each of which is the result of the transformation between two adjacent coordinate frames. This transformation is started from the base coordinate frame to the end-effector. The torso located in the center of the NAO robot body is defined as the base coordinate frame. To denote the transformation between two adjacent coordinate frames, we use Denavit-Hartenberg (DH) parameters. These parameters are a_{i-1}, α_{i-1}, d_i, and θ_i. The a_{i-1} parameter denotes the distance between z_{i-1} axis and z_i axis measured along x_{i-1} axis, while the α_{i-1} parameter is the angle between z_{i-1} axis and z_i axis measured about x_{i-1} axis. The d_i parameter represents the distance between x_{i-1} axis and x_i axis measured along z_i axis, and then the θ_i parameter is the angle between x_{i-1} axis and x_i axis measured about z_i axis. By using these parameters, the general transformation matrix equation resulted from the transformation between two neighboring coordinate frames can be defined as [42]

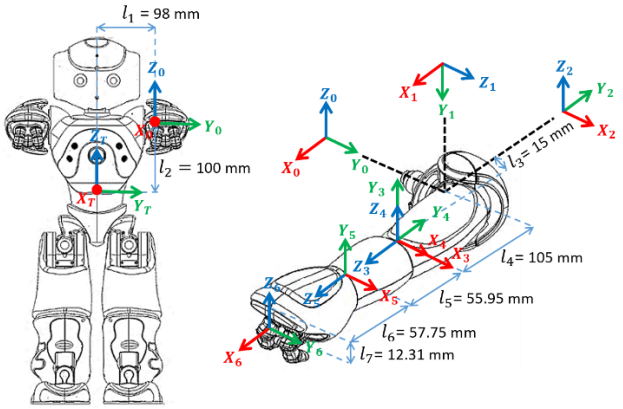


Figure. 1 Coordinate frames in the NAO's left arm

$$T_i^{i-1} = R_X(\alpha_{i-1})D_X(a_{i-1})R_Z(\theta_i)D_Z(d_i) \quad (1)$$

$$T_i^{i-1} = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i\, c\alpha_{i-1} & c\theta_i\, c\alpha_{i-1} & -s\alpha_{i-1} & -d_i\, s\alpha_{i-1} \\ s\theta_i\, s\alpha_{i-1} & c\theta_i\, s\alpha_{i-1} & c\alpha_{i-1} & d_i\, c\alpha_{i-1} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where s and c denote sin and cos, respectively.

2.1 Forward kinematics of NAO's left arm

The NAO's left arm is physically composed of a set of rigid links connected by five joint actuators. The connected rigid links by the joint actuators form a kinematic chain with the NAO's left hand as the end-effector. To formulate the forward kinematics of the NAO's left arm, we require to initially attach a set of coordinate frames from the torso to the end-effector. The attachment of these coordinate frames is performed at the zero posture: the standing NAO robot with straight legs and arms pointing forward. All of the coordinate frames that we attached to the NAO's left arm can be seen in Fig. 1.

Based on Fig. 1, the coordinate frame located in the torso has three axes, where the x-axis is pointing forward, the y-axis is pointing to the left, and the z-axis is vertical. Then to reach its adjacent coordinate frame, the current coordinate frame located in the torso should be translated along the shoulder offset in

Table 2. DH parameters for the NAO's left arm

Frame	$\boldsymbol{\theta_i}$	$\boldsymbol{\alpha_{i-1}}$	$\boldsymbol{a_{i-1}}$	$\boldsymbol{d_i}$
LShoulder Pitch	θ_1	$-\pi/2$	0	0
LShoulder Roll	$\theta_2 + \pi/2$	$\pi/2$	0	0
LElbow Yaw	θ_3	$\pi/2$	l_3	l_4
LElbow Roll	θ_4	$-\pi/2$	0	0
LWrist Yaw	θ_5	$\pi/2$	0	l_5

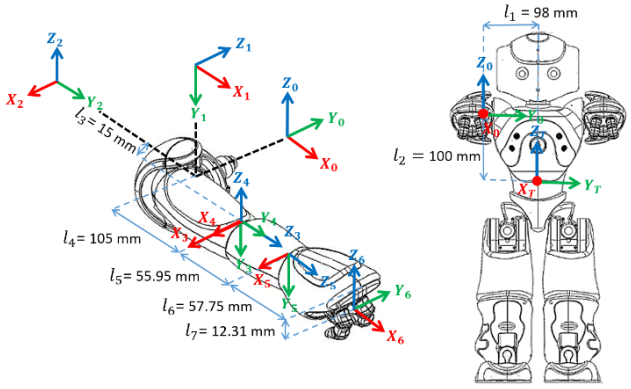


Figure. 2 Coordinate frames in the NAO's right arm

the direction of the y_0-axis (l_1), and then followed by the translation along the shoulder offset in the direction of the z_0-axis (l_2). This transformation can be represented as

$$T_0^{Ts} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & l_1 \\ 0 & 0 & 1 & l_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

The transformation should be then continued to the other coordinate frames, from the shoulder pitch to the wrist yaw. Such transformations produce a set of DH parameters, as presented in Table 2.

By plugging the DH parameters into the general transformation matrix Eq. (1), we can obtain a set of transformation matrices as follows:

$$T_1^0 = \begin{bmatrix} cos\,\theta_1 & -sin\,\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -sin\,\theta_1 & -cos\,\theta_1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

$$T_2^1 = \begin{bmatrix} -sin\,\theta_2 & -cos\,\theta_2 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ cos\,\theta_2 & -sin\,\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

$$T_3^2 = \begin{bmatrix} cos\,\theta_3 & -sin\,\theta_3 & 0 & l_3 \\ 0 & 0 & -1 & -l_4 \\ sin\,\theta_3 & cos\,\theta_3 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

$$T_4^3 = \begin{bmatrix} cos\,\theta_4 & -sin\,\theta_4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -sin\,\theta_4 & -cos\,\theta_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6}$$

$$T_5^4 = \begin{bmatrix} cos\,\theta_5 & -sin\,\theta_5 & 0 & 0 \\ 0 & 0 & -1 & -l_5 \\ sin\,\theta_5 & cos\,\theta_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

To reach the left hand, the coordinate frame located in the wrist yaw must be initially aligned by using a clockwise rotation on the x_5-axis by $\pi/2$ and a clockwise rotation on the z_5-axis by $\pi/2$. This aligned coordinate frame should be then translated along the hand offset in the direction of the x_6-axis (l_6), and then followed by a translation along the hand offset in the opposite direction of the z_6-axis (l_7). This transformations can be represented as

$$T_{LH}^5 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -l_7 \\ 1 & 0 & 0 & l_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{8}$$

The forward kinematics of the NAO's left arm can be eventually obtained by multiplying a set of the transformation matrices.

$$T_{LH}^{Ts} = T_0^{Ts}\, T_1^0\, T_2^1\, T_3^2\, T_4^3\, T_5^4\, T_{LH}^5 \tag{9}$$

2.2 Forward kinematics of NAO's right arm

The NAO's right arm contains serial rigid links connected by five joint actuators. This configuration forms a kinematic chain with the NAO's right hand as the end-effector. The kinematics structure of the right arm in a humanoid robot is typically symmetric to that of the left arm. But, the kinematics structure of the NAO robot arms is not fully symmetric since the physical arrangement between the left shoulder pitch and right shoulder pitch joints is not reflected to each other. To formulate the forward kinematics of the NAO's right arm, the first step is to attach a set of coordinate frames from the torso to the end-effector, as depicted in Fig. 2.

The coordinate frame attachment is started from the torso. To reach its neighboring coordinate frame, the coordinate frame in the torso must be translated along the shoulder offset in the opposite direction of the y_0-axis (l_1) and then followed by a translation along the shoulder offset in the direction of the z_0-axis (l_2). This transformation can be represented as

Table 3. DH parameters for the NAO's right arm

Frame	$\boldsymbol{\theta_i}$	$\boldsymbol{\alpha_{i-1}}$	$\boldsymbol{a_{i-1}}$	$\boldsymbol{d_i}$
RShoulder Pitch	θ_1	$-\pi/2$	0	0
RShoulder Roll	θ_2 $-\pi/2$	$\pi/2$	0	0
RElbow Yaw	θ_3	$-\pi/2$	l_3	l_4
RElbow Roll	θ_4	$\pi/2$	0	0
RWrist Yaw	θ_5	$-\pi/2$	0	l_5

$$T_0^{Ts} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -l_1 \\ 0 & 0 & 1 & l_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{10}$$

Then the sequential transformation between two adjacent coordinate frames from the shoulder pitch to the wrist yaw can yield a set of DH parameters, as shown in Table 3.

By feeding the DH parameters into the general transformation matrix Eq. (1), we can generate a set of transformation matrices as follows:

$$T_1^0 = \begin{bmatrix} cos\,\theta_1 & -sin\,\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -sin\,\theta_1 & -cos\,\theta_1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{11}$$

$$T_2^1 = \begin{bmatrix} sin\,\theta_2 & cos\,\theta_2 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ -cos\,\theta_2 & sin\,\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{12}$$

$$T_3^2 = \begin{bmatrix} cos\,\theta_3 & -sin\,\theta_3 & 0 & l_3 \\ 0 & 0 & 1 & l_4 \\ -sin\,\theta_3 & -cos\,\theta_3 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{13}$$

$$T_4^3 = \begin{bmatrix} cos\,\theta_4 & -sin\,\theta_4 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ sin\,\theta_4 & cos\,\theta_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{14}$$

$$T_5^4 = \begin{bmatrix} cos\,\theta_5 & -sin\,\theta_5 & 0 & 0 \\ 0 & 0 & 1 & l_5 \\ -sin\,\theta_5 & -cos\,\theta_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{15}$$

To reach the right hand, the current coordinate frame located in the right wrist yaw must be initially aligned by using a counterclockwise rotation on the x_5-axis by $\pi/2$ and a counterclockwise rotation on the z_5-axis by $\pi/2$. This aligned coordinate frame should be then translated along the hand offset in the direction of the x_6-axis (l_6) and followed by the translation along the hand offset in the opposite direction of the z_6-axis (l_7). Such transformations can be represented as

$$T_{RH}^5 = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & l_7 \\ 1 & 0 & 0 & l_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{16}$$

The forward kinematics of the NAO's right arm can be generated by the multiplication of a set of the transformation matrices.

$$T_{RH}^{Ts} = T_0^{Ts}\, T_1^0\, T_2^1\, T_3^2\, T_4^3\, T_5^4\, T_{RH}^5 \tag{17}$$

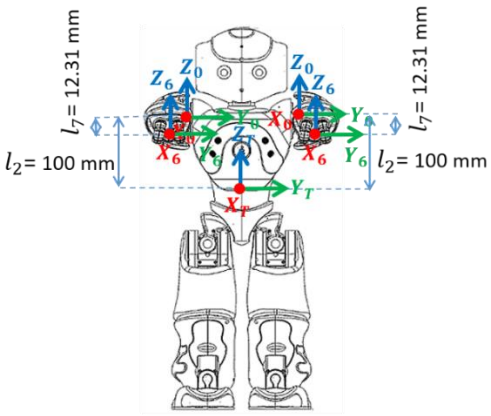
2.3 Forward kinematics verification

Verification needs to be performed to reveal the correctness of the forward kinematics of the NAO's arms. As mentioned earlier, the result of the forward kinematics is represented in a transformation matrix. This transformation matrix can be decomposed into the translation matrix and rotation matrix. The 3×1 translation matrix denotes the end-effector position, and the 3×3 rotation matrix denotes the end-effector orientation. The end-effector position is defined as the position of the NAO's hands with respect to the torso, and the end-effector orientation is referred to as the orientation of the NAO's hands frame relative to the torso's coordinate frame.

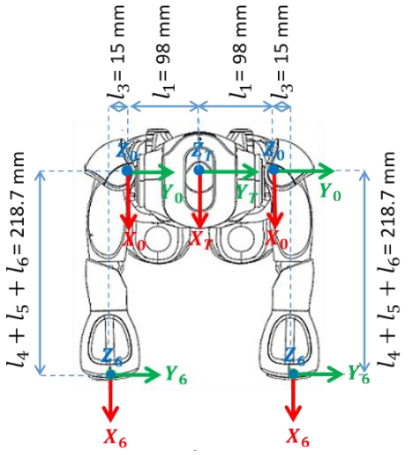
As previously mentioned, the attachment of the coordinate frames is performed at the zero posture. It is such a posture when all of the joint angles in the NAO robot are zero radians. By plugging these joint angles into the forward kinematics Eq. (9) and Eq. (17), we can obtain the transformation matrix whose elements contain the position and orientation of the end-effector. In this verification, the transformation matrix resulted from the forward kinematics should yield the end-effector position and orientation that corresponds to the NAO zero posture (see Fig. 3).

The verification result reveals that the position of the end-effector in the transformation matrix Eq. (18-19) is equal to the position of the end-effector in the NAO zero posture. Additionally, the rotational identity matrix indicates that the coordinate frame of the NAO's hands aligns with the torso's coordinate frame. It indicates that the forward kinematics of the NAO robot arms has been formulated correctly.

$$T_{LH}^{Ts} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & P_x \\ r_{21} & r_{22} & r_{23} & P_y \\ r_{31} & r_{32} & r_{33} & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 218.7 \\ 0 & 1 & 0 & 113 \\ 0 & 0 & 1 & 87.69 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{18}$$



(a)



(b)

Figure. 3 NAO zero posture: (a) front view, (b) top view

$$T_{RH}^{Ts} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & P_x \\ r_{21} & r_{22} & r_{23} & P_y \\ r_{31} & r_{32} & r_{33} & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 218.7 \\ 0 & 1 & 0 & -113 \\ 0 & 0 & 1 & 87.69 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (19)$$

3. The design of robotic motion control

3.1 Neural network-based inverse kinematics

The neural network has the ability to learn from the data, so its application can be addressed on any problem, including the inverse kinematics problem. In this research, the inverse kinematics problem will be solved by the neural network. To build the neural network-based inverse kinematics, we require a set of data. The dataset used in this research originates from a number of robotic poses. It should be known that these robotic poses are generated from various configurations of the joints in the NAO robot arms. Therefore, by extracting these robotic poses, we can obtain the data containing a set of the joint angles: shoulder pitch (θ_1), shoulder roll (θ_2), elbow yaw (θ_3), elbow roll (θ_4), and wrist yaw (θ_5).

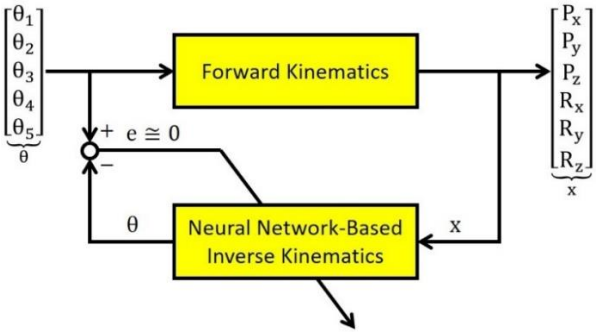


Figure. 4 Neural network-based inverse kinematics

As depicted in Fig. 4, the process of generating the dataset can be then continued by plugging a set of the joint angles into the forward kinematics Eq. (9) and Eq. (17). As a result, we obtain the data that contains the end-effector position and orientation in the cartesian workspace. This data is represented in the transformation matrix, in which the end-effector position is denoted as $[P_x, P_y, P_z]$ and the end-effector orientation is denoted as $\{[r_{11}, r_{12}, r_{13}]$, $[r_{21}, r_{22}, r_{23}]$, $[r_{31}, r_{32}, r_{33}]\}$. However, among nine rotation matrix elements, only three elements are independent [45]. It means that the rotation matrix gives a redundant description for the end-effector orientation. Instead, we used the rotation vectors $[R_x, R_y, R_z]$ to describe the end-effector orientation. To convert the rotation matrix to the rotation vectors, we use the following equations [46]

$$\phi = \arccos\left(\frac{r_{11}+r_{22}+r_{33}-1}{2}\right) \quad (20)$$

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = \frac{1}{2\sin\phi}\begin{bmatrix} r_{32}-r_{23} \\ r_{13}-r_{31} \\ r_{21}-r_{12} \end{bmatrix} \quad (21)$$

$$\begin{bmatrix} R_x \\ R_y \\ R_z \end{bmatrix} = \phi\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \quad (22)$$

The dataset that we generated to build the neural network-based inverse kinematics consists of 10000 input-output pairs. The input parameters are the end-effector position and orientation $[P_x, P_y, P_z, R_x, R_y, R_z]$, while the output parameters are a set of joint angles $[\theta_1, \theta_2, \theta_3, \theta_4, \theta_5]$. This dataset will be split into three separate datasets: training dataset, validation dataset, and testing dataset. In this case, 60% of the data will be allocated for the training dataset, 20% for the validation dataset, and the remaining 20% for the testing dataset. Each dataset has a different function. The training dataset is utilized for training the neural network. The validation dataset is used to validate the neural network performance during the training

process. The obtained information from this process helps us in tuning and choosing the neural network's hyperparameters. The testing dataset is another set of data used to test how well the performance of the trained neural network. Contrary to the training and validation datasets, the testing dataset contains data that it has never seen during the training process. These datasets must be normalized by using Eq. (23) before being fed into the neural network.

$$z = \frac{x-\mu}{\sigma} \tag{23}$$

where z denotes the normalized data, x is the feature data, μ is the mean of the feature data, and σ is the standard deviation.

Based on the input and output parameters in the dataset, the neural network-based inverse kinematics will contain six neurons in the input layer and five neurons in the output layer. In between the input and output layers, there are hidden layers. Through the training process, we can adjust the neural network's hyperparameters to find the appropriate number of hidden layers and their neurons. During the training process, the training dataset and validation dataset should be fed into the neural network. The training should be conducted until both the training loss and validation loss functions converge to zero [47].

The neural network architecture for the inverse kinematics is a fully connected neural network. Note that the neural network-based inverse kinematics has such a feedforward structure that it is classified as feedforward controller. Fig. 5 shows the relationship between the input and output in this control system. The control system whose outputs is not involved in the control action is known as the open-loop control system [48]. The drawback of this control system is that it has no feedback mechanism to overcome the remaining error in the estimation of joint angles.

3.2 Neural network-based inverse kinematics combined with Jacobian

For achieving better performance, we designed a robotic motion control that conforms to the closed-loop control system. As shown in Fig. 6, this robotic motion control is established by the neural network-based inverse kinematics combined with Jacobian. The neural network-based inverse kinematics acts as the feedforward controller, while the Jacobian acts as the feedback controller. In particular, the neural network-based inverse kinematics has a function to generate a set of joint angles (θ) based on the given position and orientation of the end-effector (x). The inclusion of the Jacobian in the design of the robotic

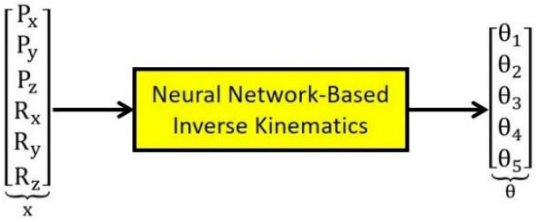


Figure. 5 Open-loop control system

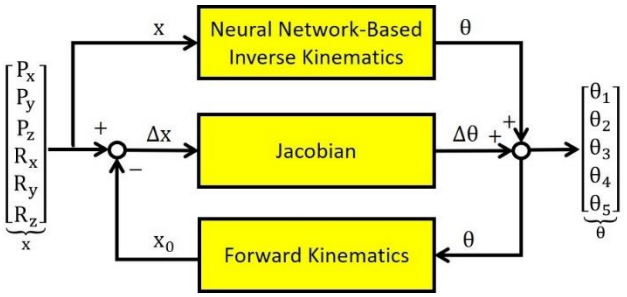


Figure. 6 Closed-loop control system

motion control is to minimize the remaining error of the neural network-based inverse kinematics.

In this way, we can generate more accurate joint angles for controlling the joint actuators. Note that this control system is principally made up of two different controllers, so this can be classified as a hybrid control system.

Contrary to the open-loop control system, this control system has a feedback signal. This feedback signal contains data from the output that is sent back to be compared with the reference input. To provide the feedback signal, a set of the joint angles resulted from the neural network-based inverse kinematics should be initially fed into the forward kinematics to obtain the actual position and orientation of the end-effector (x_0). Then by comparing it with the desired position and orientation of the end-effector (x), we can find its difference ($\Delta x = x - x_0$). This difference will be then utilized to obtain the corrective action. This corrective action is represented in terms of the displacement of the joint angles ($\Delta\theta$). Note that the relationship between the displacement of the joint angles ($\Delta\theta$) and the displacement of the end-effector position and orientation (Δx) can be defined as

$$\Delta x = J\,\Delta\theta \tag{24}$$

$$\begin{bmatrix} \Delta P_x \\ \Delta P_y \\ \Delta P_z \\ \Delta R_x \\ \Delta R_y \\ \Delta R_z \end{bmatrix} = \begin{bmatrix} \frac{\partial P_x}{\partial \theta_1} & \frac{\partial P_x}{\partial \theta_2} & \frac{\partial P_x}{\partial \theta_3} & \frac{\partial P_x}{\partial \theta_4} & \frac{\partial P_x}{\partial \theta_5} \\ \frac{\partial P_y}{\partial \theta_1} & \frac{\partial P_y}{\partial \theta_2} & \frac{\partial P_y}{\partial \theta_3} & \frac{\partial P_y}{\partial \theta_4} & \frac{\partial P_y}{\partial \theta_5} \\ \frac{\partial P_z}{\partial \theta_1} & \frac{\partial P_z}{\partial \theta_2} & \frac{\partial P_z}{\partial \theta_3} & \frac{\partial P_z}{\partial \theta_4} & \frac{\partial P_z}{\partial \theta_5} \\ \frac{\partial R_x}{\partial \theta_1} & \frac{\partial R_x}{\partial \theta_2} & \frac{\partial R_x}{\partial \theta_3} & \frac{\partial R_x}{\partial \theta_4} & \frac{\partial R_x}{\partial \theta_5} \\ \frac{\partial R_y}{\partial \theta_1} & \frac{\partial R_y}{\partial \theta_2} & \frac{\partial R_y}{\partial \theta_3} & \frac{\partial R_y}{\partial \theta_4} & \frac{\partial R_y}{\partial \theta_5} \\ \frac{\partial R_z}{\partial \theta_1} & \frac{\partial R_z}{\partial \theta_2} & \frac{\partial R_z}{\partial \theta_3} & \frac{\partial R_z}{\partial \theta_4} & \frac{\partial R_z}{\partial \theta_5} \end{bmatrix} \begin{bmatrix} \Delta\theta_1 \\ \Delta\theta_2 \\ \Delta\theta_3 \\ \Delta\theta_4 \\ \Delta\theta_5 \end{bmatrix}$$

Thereafter, the corrective action can be obtained by means of calculating the displacement of the joint angles ($\Delta\theta$). This can be conducted by inverting the Jacobian matrix. To avoid the singularity, we use a technique called damped least squares (DLS) [49], as in the following equation

$$\Delta\theta = J^T(JJ^T + \lambda^2 I)^{-1}\Delta x \tag{25}$$

where J denotes the Jacobian matrix, I is the identity matrix, J^T is the Jacobian matrix transpose, and λ is a non-zero damping constant.

Eventually, the required joint angles for driving the robot's end-effector to the desired targets can be obtained by

$$\theta = \theta + \Delta\theta \tag{26}$$

4. Result and discussion

4.1 Neural network architecture for inverse kinematics

It is important to note that the dataset to build a neural network-based inverse kinematics comprises the input-output pairs. In this case, the input data is the position and orientation of the end-effector in the cartesian space, while the output data is a set of joint angles whose operation is in the joint space. Due to containing the input-output pairs, this is classified as supervised learning. The objective of this learning is to find the neural network architecture that defines the relationship model between the input and output. This relationship model is called the neural network-based inverse kinematics. The experiment has been conducted to find the neural network-based inverse kinematics.

From the experimental result, we can obtain the neural network architecture for inverse kinematics, as depicted in Fig. 7. In this architecture, this neural network-based inverse kinematics has six neurons in the input layer and five neurons in the output layer. Between the input and output layers, there are three hidden layers, each of which contains 48 neurons, 120 neurons, and 60 neurons. In the last two hidden layers, we used the dropout whose value is 0.05. The activation function we used for the hidden layers is sigmoid and for the output layer is linear activation. The linear activation is employed in the output layer because this is a regression problem whose outputs are continuous values, not discrete labels as in the classification problem. In addition, the other neural network's hyperparameters can be seen in Table 4.

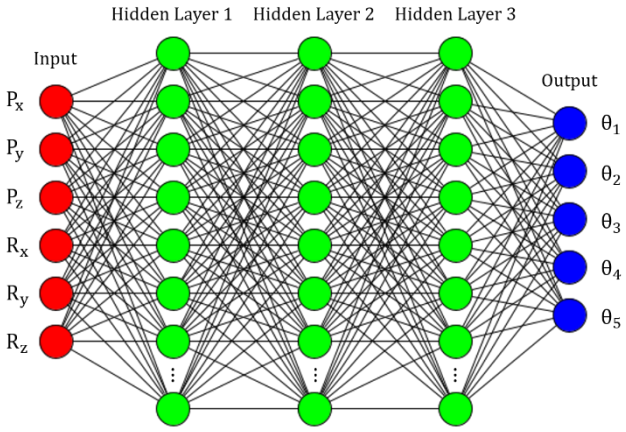


Figure. 7 Neural network architecture for inverse kinematics

Table 4. Hyperparameters

Hyperparameters	Value
Learning rate	0.001
Epoch	2000
Optimizer	Adam
Loss function	MSE

4.2 The validation of the trained neural network-based inverse kinematics

The neural network-based inverse kinematics is a fundamental aspect in the design of robotic motion control, so its performance must be validated. In this validation, we will check whether this trained neural network suffers from overfitting. Overfitting is such a condition where the neural network performs well on the training data but cannot perform satisfactorily on other data. It is because the neural network tends to memorize the training data instead of learning from it. Consequently, the neural network is unable to give accurate predictions and cannot perform well on new data. Overfitting can be identified during the training process. It can be conducted by comparing the training loss and the validation loss. The trained neural network is said not to be overfitting when the

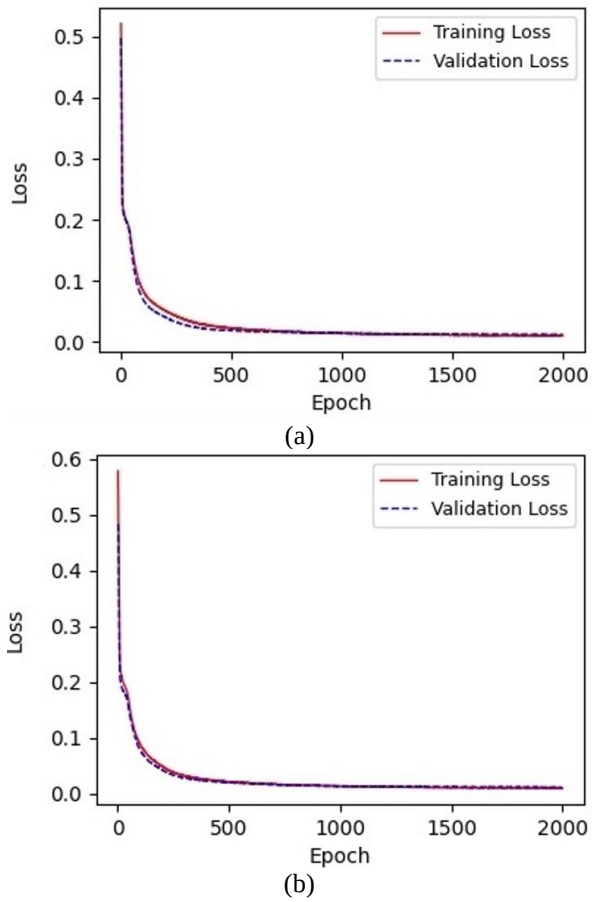


Figure. 8 Neural network validation: (a) NAO's left arm, and (b) NAO's right arm

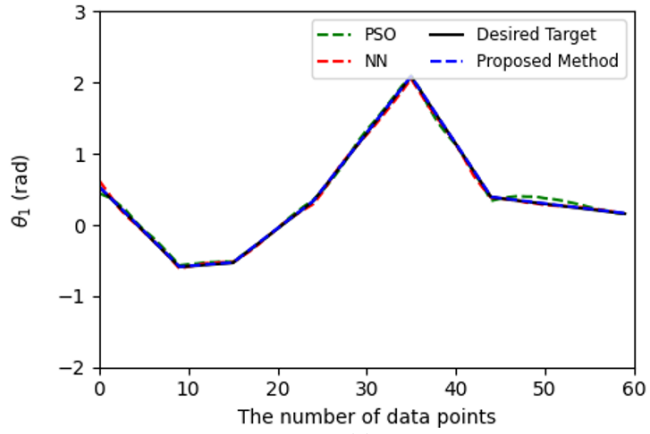
training loss and the validation loss converge to zero [47]. Fig. 8 shows that both the training loss and the validation loss synchronously converge to zero. This means that our trained neural network-based inverse kinematics does not suffer from overfitting.

4.3 The performance comparison of three different approaches

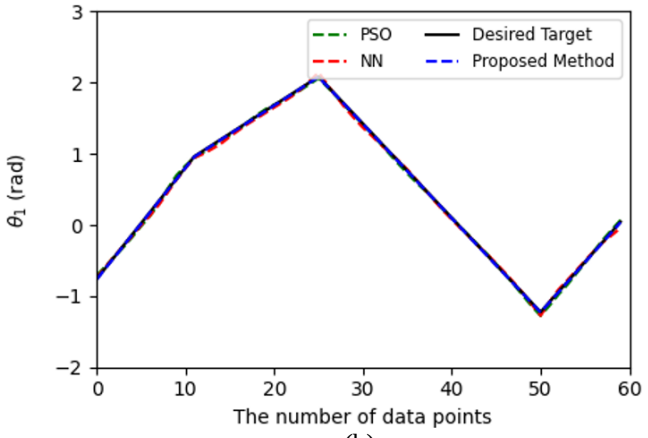
In this research, the performance of the designed robotic motion control by three different approaches will be compared in terms of providing a set of joint angles for the joint actuators. The first approach is particle swarm optimization (PSO), as conducted by this research [34-36]. Particle swarm optimization is one such metaheuristic algorithm that is powerful and easy to implement since it has fewer parameters compared to the other metaheuristic algorithms. In this approach, a swarm is a collection of particles in a search space. Each particle in a swarm represents a candidate solution, in this case, a set of joint angles. Initially, these joint angles are randomly determined for each particle. By feeding the joint angles into the forward kinematics, we can calculate the actual end-effector locations of each particle. Each particle has a cost function calculated via the distance between the desired and actual end-effector locations. Once we have the cost function of all the particles, we can then sort them and find the minimum one among all the particles. This process should be conducted until the distance between the desired and actual end-effector locations converges to zero. As explained in [36], the particle with the lowest cost function is the optimal solution for the inverse kinematics.

Meanwhile, the second approach is the neural network-based inverse kinematics. Because of its learning ability, most research used this approach [37-41]. Different from this approach, the approach we proposed in this research is the neural network-based inverse kinematics combined with Jacobian. To demonstrate their performance, a set of desired targets to be reached by the robot's end-effector are given in the cartesian space, as depicted in Fig. 10. This data is taken from a few data collections in the testing dataset. In this demonstration, the designed robotic motion control by the three approaches is supposed to generate a set of joint angles required to place the robot's end-effector to the desired targets. The joint paths resulted from the continuous motion from one set of joint angles to another can be seen in Fig. 9. The performance comparison results showed that the designed robotic motion control by using our proposed approach can provide more accurate joint angles than the other ones. Therefore, the joint paths obtained from the continuous motion from one set of joint angles to another are closer to the desired joint paths.

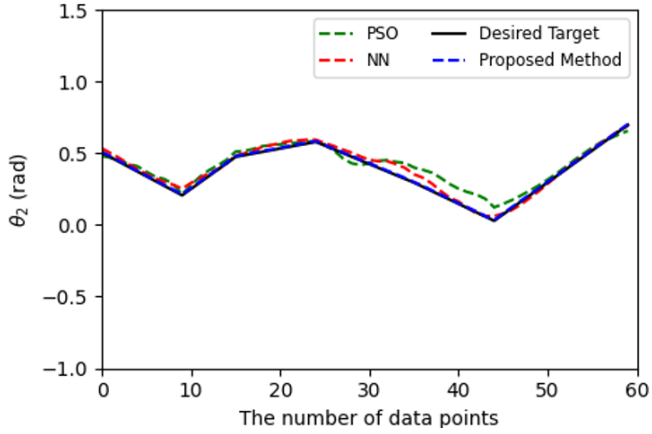
In addition to comparing their joint paths, we also measure their mean squared error (MSE). It is necessary to reveal which robotic motion control has the lowest error in the estimation of the joint angles. Table 5 shows the performance comparison between three different approaches. Based on the comparison results, the averaged MSE for the particle swarm optimization was 3.47×10^{-3} rad and 1.19×10^{-3} rad for the neural network. It indicates that the neural network has better performance than particle swarm optimization because it has a lower averaged MSE. The particle swarm optimization is such a stochastic optimization that needs some random initializations, so its convergence to the desired solution is highly dependent on the initial guess. This approach cannot find an optimal solution when the initial guess is not appropriately selected. That is why most researchers preferred to use the neural network for solving the inverse kinematics problem. Nevertheless, compared with the neural network-based inverse kinematics, our proposed approach has a lower averaged MSE (3.72×10^{-5}). This is because our proposed approach has the feedback mechanism to compensate for the remaining error. The presence of the Jacobian in our


PSO
NN
Desired Target
Proposed Method
θ_1 (rad)
The number of data points

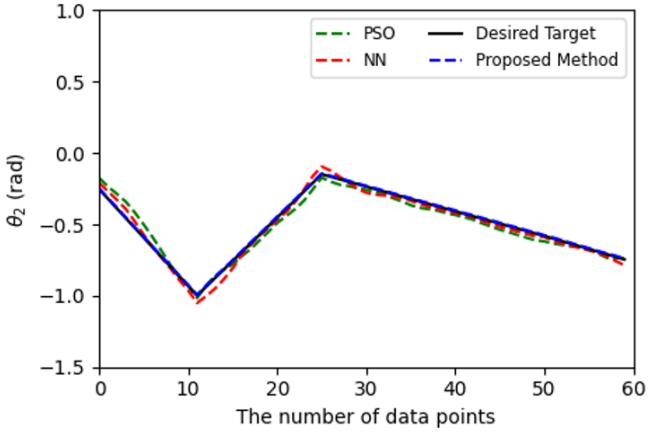
(a)


PSO
NN
Desired Target
Proposed Method
θ_1 (rad)
The number of data points

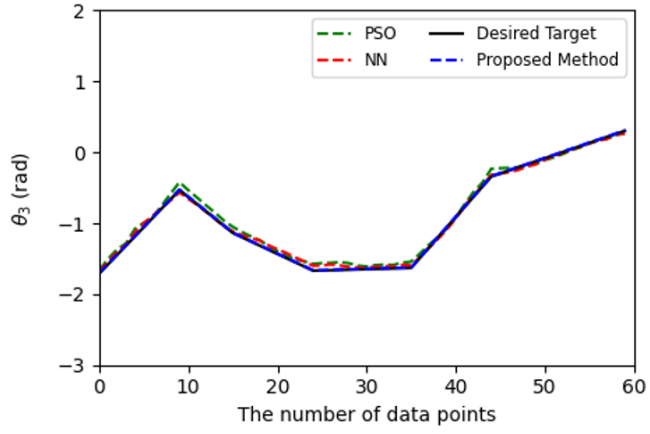
(b)


PSO
NN
Desired Target
Proposed Method
θ_2 (rad)
The number of data points

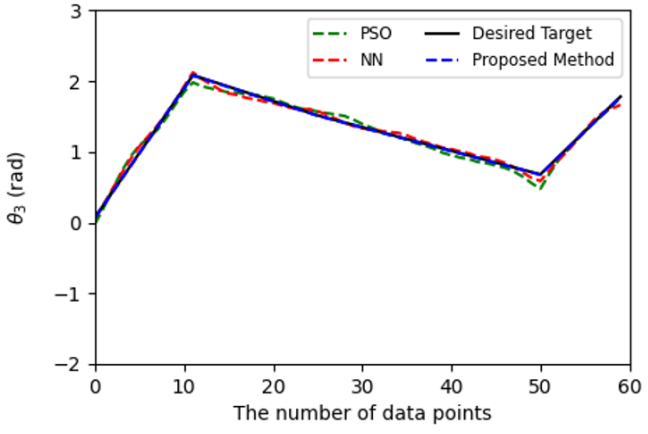
(c)


PSO
NN
Desired Target
Proposed Method
θ_2 (rad)
The number of data points

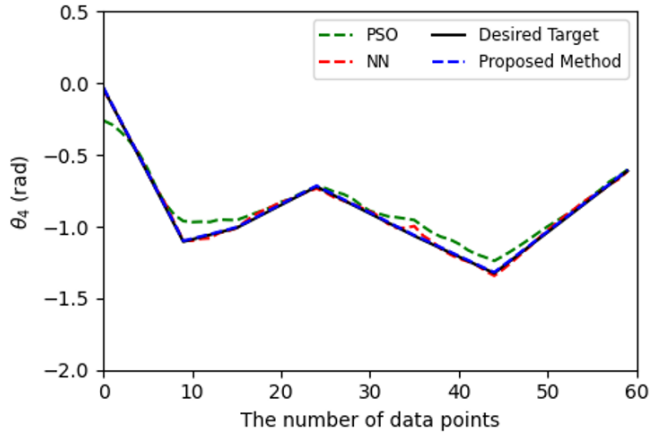
(d)


PSO
NN
Desired Target
Proposed Method
θ_3 (rad)
The number of data points

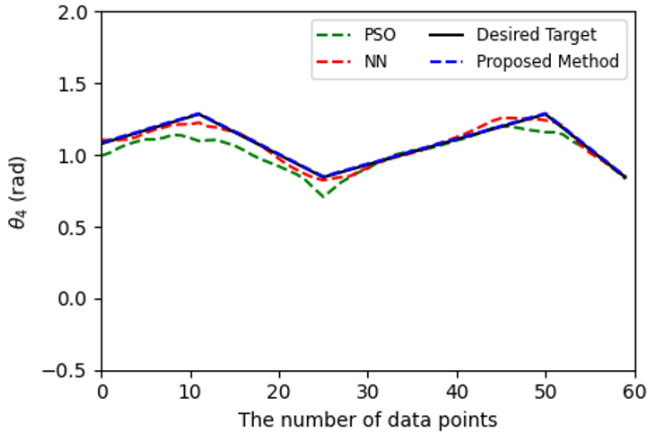
(e)


PSO
NN
Desired Target
Proposed Method
θ_3 (rad)
The number of data points

(f)


PSO
NN
Desired Target
Proposed Method
θ_4 (rad)
The number of data points

(g)


PSO
NN
Desired Target
Proposed Method
θ_4 (rad)
The number of data points

(h)

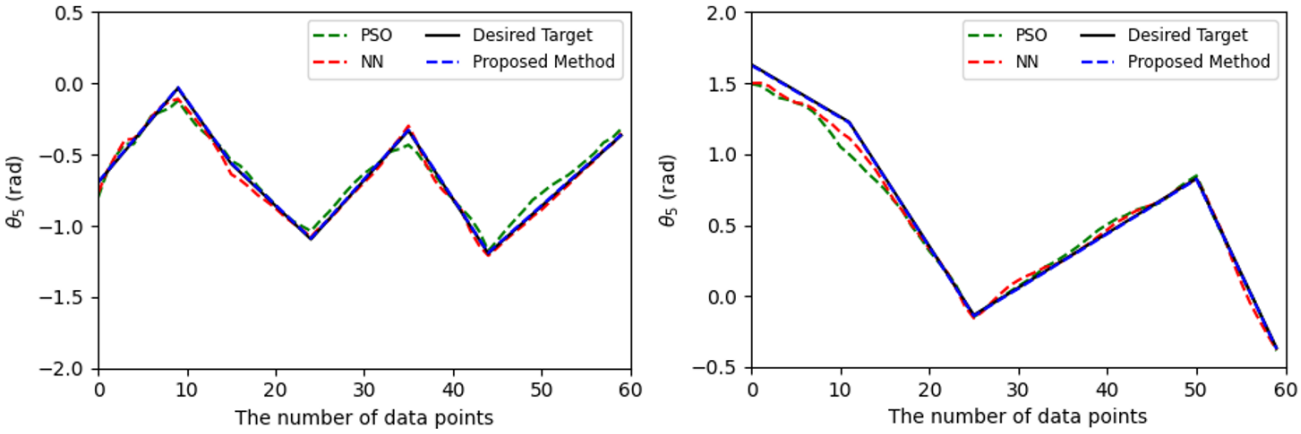


Figure. 9 Joint paths: (a) the left shoulder pitch, (b) the right shoulder pitch, (c) the left shoulder roll, (d) the right shoulder roll, (e) the left elbow yaw, (f) the right elbow yaw, (g) the left elbow roll, (h) the right elbow roll, (i) the left wrist yaw, and (j) the right wrist yaw

Table 5. Performance comparison result

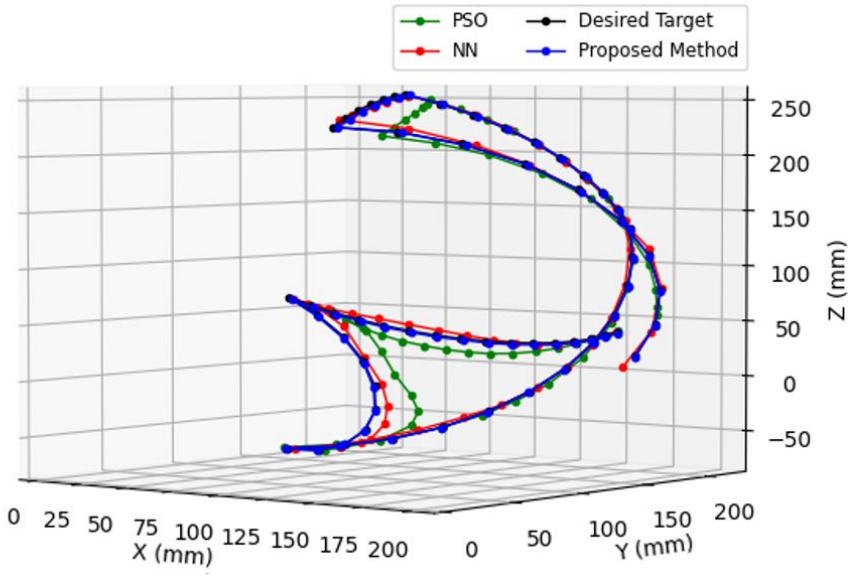
No	Joint Name	MSE (rad)		
		PSO [36]	**Neural Network** [41]	**Proposed Approach**
1.	LShoulder Pitch	1.52×10^{-3}	6.59×10^{-4}	3.14×10^{-5}
2.	LShoulder Roll	3.41×10^{-3}	9.44×10^{-4}	2.88×10^{-5}
3.	LElbow Yaw	4.03×10^{-3}	1.42×10^{-3}	4.55×10^{-5}
4.	LElbow Roll	4.59×10^{-3}	2.60×10^{-4}	3.23×10^{-5}
5.	LWrist Yaw	2.72×10^{-3}	1.03×10^{-3}	4.61×10^{-5}
6.	RShoulder Pitch	7.05×10^{-4}	9.93×10^{-4}	3.17×10^{-5}
7.	RShoulder Roll	1.99×10^{-3}	9.91×10^{-4}	3.04×10^{-5}
8.	RElbow Yaw	4.53×10^{-3}	2.25×10^{-3}	4.72×10^{-5}
9.	RElbow Roll	5.52×10^{-3}	7.56×10^{-4}	3.18×10^{-5}
10.	RWrist Yaw	5.72×10^{-3}	2.68×10^{-3}	4.65×10^{-5}
The Averaged MSE		**3.47×10^{-3}**	**1.19×10^{-3}**	**$3.72\ 10^{-5}$**

proposed approach is to reduce the remaining error of the neural network. As a result, we obtained more accurate joint angles for the joint actuators. From Table 5, the performance comparison results stated that, among three different approaches, our proposed approach has the smallest averaged MSE.

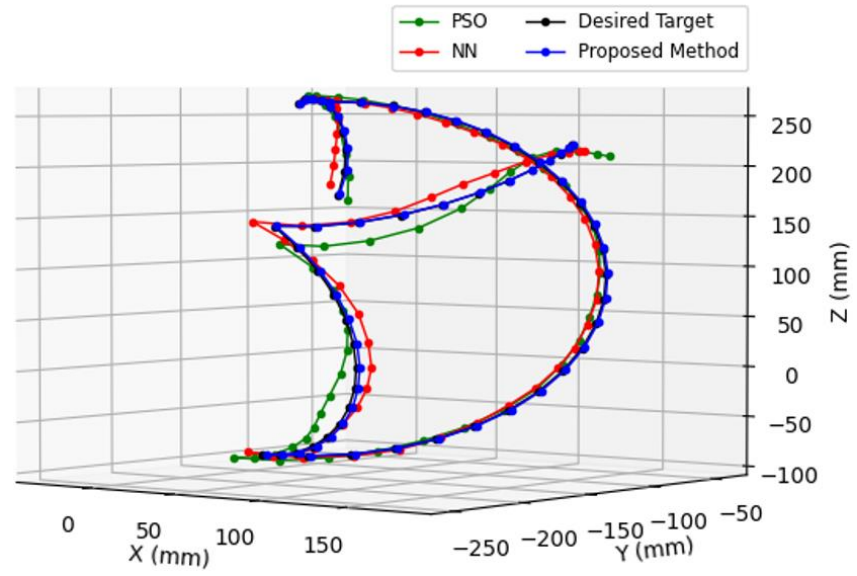
4.4 The effectiveness of the proposed approach

In this research, the demonstration was carried out with three different approaches. In the previous section, we demonstrated how the designed robotic motion control by the three different approaches can generate the joint paths as a result of the continuous motion from one set of the joint angles to another. In this section, we will demonstrate whether a set of the obtained joint angles can place the robot's end-effector to the desired targets. It should be noted that a series of the desired targets to be reached by the robot's end-effector can form the cartesian paths, as illustrated in Fig. 10. In a simple way, to illustrate the robot's end-effector movement in achieving the desired targets, we can feed a set of these obtained joint angles into the forward kinematics Eq. (9) and Eq. (17). By doing so, we can obtain the data points containing the end-effector locations in the cartesian space. By connecting these data points sequentially, we can visualize the cartesian path resulted from the robot's end-effector movement. Then, by comparing the cartesian paths obtained from the three different approaches, we can notice how well they perform. Through this demonstration, we can find out which one performs better.

From the simulation results in Fig. 10, although the designed robotic motion control by the particle swarm optimization (PSO) and the neural network-based inverse kinematics can generate a set of the joint angles to drive the robot's end-effector in the cartesian space, the cartesian paths resulted from the movement of the robot's end-effector still deviated from the desired path. It is because the required joint angles to drive the robot's end-effector are not so close to the desired targets. Therefore, for obtaining better performance, we designed the robotic motion control by using the neural network-based inverse kinematics integrated with Jacobian. The inclusion of



(a)



(b)

Figure. 10 Cartesian path resulted from the end-effector movement: (a) the left arm and (b) the right arm

the Jacobian in the design of the robotic motion control can helpfully minimize the remaining error of the neural network-based inverse kinematics, thus producing more accurate joint angles. As a result, the cartesian path resulted from the movement of the robot's end-effector is very close to the desired path. In addition to possessing a lower averaged MSE, the cartesian path obtained from our proposed approach is closer to the desired target path. It confirmed the effectiveness of our proposed approach.

4 Conclusion

This paper proposed a new approach to enhance the performance of the designed robotic motion control for the NAO robot arms. In the design of the robotic motion control, the approach we proposed in this research is the neural network-based inverse kinematics combined with the Jacobian. In this case, the neural network-based inverse kinematics plays as the feedforward controller whose function is to estimate a set of joint angles, whereas the Jacobian that acts as the feedback controller is responsible for reducing the remaining error of the neural network-based inverse kinematics in the estimation of the joint angles. In this way, we can obtain more precise joint angles for controlling the joint actuators such that the robot's end-effector can be driven along the desired path in the cartesian space. According to the performance comparison results, among the three different approaches, our proposed approach has a better performance due to the fact that it has a lower averaged MSE in the estimation of the joint angles. For further research, we will design and develop the

robotic motion control for cooperative multi-agent systems.

Conflicts of interest

The authors declare no conflict of interest.

Author contributions

Arif Nugroho: conceptualization, methodology, formal analysis, data curation, writing–original draft preparation and editing. Eko Mulyanto Yuniarno: methodology, formal analysis, validation, writing–review. Mauridhi Hery Purnomo: conceptualization, validation, supervision, writing–review.

Acknowledgments

We would like to sincerely express our gratitude to the Ministry of Research, Technology, and Higher Education of the Republic of Indonesia for providing financial support through the PMDSU (Pendidikan Magister menuju Doktor untuk Sarjana Unggul) scholarship.

References

[1] X. Chen, A. G. Dharmawan, S. Foong, and G. S. Soh, "Seam tracking of large pipe structures for an agile robotic welding system mounted on scaffold structures", *Robot. Comput.-Integr. Manuf.*, Vol. 50, pp. 242–255, 2018, doi: 10.1016/j.rcim.2017.09.018.

[2] B. Zhang, J. Wu, L. Wang, and Z. Yu, "Accurate dynamic modeling and control parameters design of an industrial hybrid spray-painting robot", *Robot. Comput.-Integr. Manuf.*, Vol. 63, p. 101923, 2020, doi: 10.1016/j.rcim.2019.101923.

[3] F. Li, Q. Jiang, W. Quan, S. Cai, R. Song, and Y. Li, "Manipulation Skill Acquisition for Robotic Assembly Based on Multi-Modal Information Description", *IEEE Access*, Vol. 8, pp. 6282–6294, 2020, doi: 10.1109/ACCESS.2019.2934174.

[4] A. Raj and A. Thakur, "Dynamically feasible trajectory planning for Anguilliform-inspired robots in the presence of steady ambient flow", *Robot. Auton. Syst.*, Vol. 118, pp. 144–158, 2019, doi: 10.1016/j.robot.2019.05.001.

[5] H. J. Shin, H. K. Yoo, J. H. Lee, S. R. Lee, K. Jeong, and H. S. Moon, "Robotic single-port surgery using the da Vinci SP® surgical system for benign gynecologic disease: A preliminary report", *Taiwan. J. Obstet. Gynecol.*, Vol. 59, No. 2, pp. 243–247, 2020, doi: 10.1016/j.tjog.2020.01.012.

[6] D. Kulic, G. Venture, K. Yamane, E. Demircan, I. Mizuuchi, and K. Mombaur, "Anthropomorphic Movement Analysis and Synthesis: A Survey of Methods and Applications", *IEEE Trans. Robot.*, Vol. 32, No. 4, pp. 776–795, 2016, doi: 10.1109/TRO.2016.2587744.

[7] A. Robotics, "H25 - Joints - V3.3 — Aldebaran Software 2.1.0.18 documentation", 2014, https://fileadmin.cs.lth.se/robot/nao/doc/family/nao_h25/joints_h25_v33.html#h25-joints-v33 (accessed Aug. 16, 2021).

[8] A. Robotics, "H21 - Joints - V3.3 — Aldebaran Software 2.1.0.18 documentation", 2014, https://fileadmin.cs.lth.se/robot/nao/doc/family/nao_h21/joints_h21_v33.html#h21-joints-v33 (accessed Aug. 16, 2021).

[9] N. Kofinas, "Forward and Inverse Kinematics for the NAO Humanoid Robot", *Technical University of Crete, Greece*, 2012. [Online]. Available: https://www.cs.umd.edu/~nkofinas/Projects/KofinasThesis.pdf

[10] S. Kucuk and Z. Bingul, "Robot Kinematics: Forward and Inverse Kinematics", in *Industrial Robotics: Theory, Modelling and Control*, S. Cubero, Ed., Pro Literatur Verlag, Germany / ARS, Austria, 2006, doi: 10.5772/5015.

[11] M. Dahari and J. D. Tan, "Forward and inverse kinematics model for robotic welding process using KR-16KS KUKA robot", in *2011 Fourth International Conference on Modeling, Simulation and Applied Optimization*, Kuala Lumpur, Malaysia, pp. 1–6, 2011. doi: 10.1109/ICMSAO.2011.5775598.

[12] S. Qiu and M. R. Kermani, "Inverse Kinematics of High Dimensional Robotic Arm-Hand Systems for Precision Grasping", *J. Intell. Robot. Syst.*, Vol. 101, No. 4, p. 70, 2021, doi: 10.1007/s10846-021-01349-7.

[13] A. R. J. Almusawi, L. C. Dülger, and S. Kapucu, "A New Artificial Neural Network Approach in Solving Inverse Kinematics of Robotic Arm (Denso VP6242)", *Comput. Intell. Neurosci.*, Vol. 2016, pp. 1–10, 2016, doi: 10.1155/2016/5720163.

[14] A. V. S. S. Somasundar and G. Yedukondalu, "Robotic path planning and simulation by jacobian inverse for industrial applications", *Procedia Comput. Sci.*, Vol. 133, pp. 338–347, 2018, doi: 10.1016/j.procs.2018.07.042.

[15] A. T. Hasan and H. M. A. A. A. Assadi, "Performance Prediction Network for Serial Manipulators Inverse Kinematics Solution Passing Through Singular Configurations", *Int.*

J. Adv. Robot. Syst., Vol. 7, No. 4, p. 36, 2010, doi: 10.5772/10492.

[16] O. M. Omisore, S. Han, L. Ren, N. Zhang, K. Ivanov, A. Elazab, and L. Wang, "Non-iterative geometric approach for inverse kinematics of redundant lead-module in a radiosurgical snake-like robot", *Biomed. Eng. OnLine*, Vol. 16, No. 1, p. 93, 2017, doi: 10.1186/s12938-017-0383-2.

[17] T. Zhu, Q. Zhao, W. Wan, and Z. Xia, "Robust Regression-Based Motion Perception for Online Imitation on Humanoid Robot", *Int. J. Soc. Robot.*, Vol. 9, No. 5, pp. 705–725, 2017, doi: 10.1007/s12369-017-0416-9.

[18] R. V. N. Kumar and R. Sreenivasulu, "Inverse Kinematics (IK) Solution of a Robotic Manipulator using PYTHON", *J. Mechatron. Robot.*, Vol. 3, No. 1, pp. 542–551, 2019, doi: 10.3844/jmrsp.2019.542.551.

[19] L. Sardana, M. K. Sutar, and P. M. Pathak, "A geometric approach for inverse kinematics of a 4-link redundant In-Vivo robot for biopsy", *Robot. Auton. Syst.*, Vol. 61, No. 12, pp. 1306–1313, Dec. 2013, doi: 10.1016/j.robot.2013.09.001.

[20] S. A. Nugroho, A. S. Prihatmanto, and A. S. Rohman, "Design and implementation of kinematics model and trajectory planning for NAO humanoid robot in a tic-tac-toe board game", In: *Proc. of 2014 IEEE 4th International Conference on System Engineering and Technology (ICSET)*, Bandung, Indonesia: IEEE, pp. 1–7, 2014, doi: 10.1109/ICSEngT.2014.7111783.

[21] N. Kofinas, E. Orfanoudakis, and M. G. Lagoudakis, "Complete Analytical Forward and Inverse Kinematics for the NAO Humanoid Robot", *J. Intell. Robot. Syst.*, Vol. 77, No. 2, pp. 251–264, 2015, doi: 10.1007/s10846-013-0015-4.

[22] M. Zhang, J. Chen, X. Wei, and D. Zhang, "Work chain-based inverse kinematics of robot to imitate human motion with Kinect", *ETRI J.*, Vol. 40, No. 4, pp. 511–521, 2018, doi: 10.4218/etrij.2018-0057.

[23] A. K. Singh, P. Chakraborty, and G. C. Nandi, "Sketch drawing by NAO humanoid robot", In: *Proc. of TENCON 2015 - 2015 IEEE Region 10 Conference*, Macao, pp. 1–6, 2015, doi: 10.1109/TENCON.2015.7373001.

[24] Y. Wang, C. Zhao, X. Wang, P. Zhang, P. Li, and H. Liu, "Inverse Kinematics of a 7-DOF Spraying Robot with 4R 3-DOF Non-spherical Wrist", *J. Intell. Robot. Syst.*, Vol. 101, No. 4, p. 68, 2021, doi: 10.1007/s10846-021-01338-w.

[25] D. Robotics, "Denso VP6242 Specification", [Online]. Available: https://www.densorobotics.com/products/5-6-axis/vp-series/

[26] K. Robotics, "Kuka KR AGILUS Specification", [Online]. Available: https://www.kuka.com/en-de/products/robot-systems/industrial-robots/kr-agilus

[27] Y. M. Robotics, "Yaskawa Motoman GP180 Specification", [Online]. Available: https://www.motoman.com/en-us/products/robots/industrial/assembly-handling/gp-series/gp180

[28] R. N. Jazar, "Theory of Applied Robotics: Kinematics, Dynamics, and Control", *Second Edition. Boston, MA: Springer US*, 2010, doi: 10.1007/978-1-4419-1750-8.

[29] S. Mukherjee, D. Paramkusam, and S. K. Dwivedy, "Inverse Kinematics of a NAO Humanoid Robot using Kinect to Track and Imitate Human Motion", In: *Proc. of International Conference on Robotics, Automation, Control and Embedded Systems*, India, p. 7, 2015.

[30] A. K. Singh, N. Baranwal, and G. C. Nandi, "Development of a self reliant humanoid robot for sketch drawing", *Multimed. Tools Appl.*, Vol. 76, No. 18, pp. 18847–18870, 2017, doi: 10.1007/s11042-017-4358-x.

[31] M. Alibeigi, "Inverse Kinematics Based Human Mimicking System using Skeletal Tracking Technology", *J Intell Robot Syst*, p. 19, 2017.

[32] L. Zhang and N. Xiao, "A novel artificial bee colony algorithm for inverse kinematics calculation of 7-DOF serial manipulators", *Soft Comput.*, Vol. 23, No. 10, pp. 3269–3277, May 2019, doi: 10.1007/s00500-017-2975-y.

[33] S. Dereli and R. Köker, "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.*, Vol. 28, No. 5, pp. 601–613, 2020, doi: 10.1080/17415977.2019.1602124.

[34] H. Deng and C. Xie, "An improved particle swarm optimization algorithm for inverse kinematics solution of multi-DOF serial robotic manipulators", *Soft Comput.*, Vol. 25, No. 21, pp. 13695–13708, Nov. 2021, doi: 10.1007/s00500-021-06007-6.

[35] L. Yiyang, J. Xi, B. Hongfei, W. Zhining, and S. Liangliang, "A General Robot Inverse Kinematics Solution Method Based on Improved PSO Algorithm", *IEEE Access*, Vol. 9,

pp. 32341–32350, 2021, doi: 10.1109/ACCESS.2021.3059714.

[36] Ö. Ekrem and B. Aksoy, "Trajectory planning for a 6-axis robotic arm with particle swarm optimization algorithm", *Eng. Appl. Artif. Intell.*, Vol. 122, p. 106099, 2023, doi: 10.1016/j.engappai.2023.106099.

[37] A. N. Sharkawy and S. S. Khairullah, "Forward and Inverse Kinematics Solution of A 3-DOF Articulated Robotic Manipulator Using Artificial Neural Network", *Int. J. Robot. Control Syst.*, Vol. 3, No. 2, pp. 330–353, 2023, doi: 10.31763/ijrcs.v3i2.1017.

[38] Y. Bai, M. Luo, and F. Pang, "An Algorithm for Solving Robot Inverse Kinematics Based on FOA Optimized BP Neural Network", *Appl. Sci.*, Vol. 11, No. 15, p. 7129, 2021, doi: 10.3390/app11157129.

[39] J. Lu, T. Zou, and X. Jiang, "A Neural Network Based Approach to Inverse Kinematics Problem for General Six-Axis Robots", *Sensors*, Vol. 22, No. 22, p. 8909, 2022, doi: 10.3390/s22228909.

[40] V. Kramar, O. Kramar, and A. Kabanov, "An Artificial Neural Network Approach for Solving Inverse Kinematics Problem for an Anthropomorphic Manipulator of Robot SAR-401", *Machines*, Vol. 10, No. 4, p. 241, 2022, doi: 10.3390/machines10040241.

[41] C. Lee and D. An, "AI-Based Posture Control Algorithm for a 7-DOF Robot Manipulator", *Machines*, Vol. 10, No. 8, p. 651, 2022, doi: 10.3390/machines10080651.

[42] J. J. Craig, *Introduction to Robotics: Mechanics and Control*, 3rd international ed. England: Pearson Higher Education, 2014.

[43] I. Boussaïd, J. Lepagnot, and P. Siarry, "A survey on optimization metaheuristics", *Inf. Sci.*, Vol. 237, pp. 82–117, 2013, doi: 10.1016/j.ins.2013.02.041.

[44] A. Nugroho, E. M. Yuniarno, and M. H. Purnomo, "Cooperative Multi-agent for The End-Effector Position of Robotic Arm Based on Consensus and PID Controller", In: *Proc. of 2019 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA)*, Tianjin, China, pp. 1–6, 2019, doi: 10.1109/CIVEMSA45640.2019.9071621.

[45] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, "Robotics: Modelling, Planning, and Control", *Advanced Textbooks in Control and Signal Processing*, 2008.

[46] "Axis–angle representation - Wikipedia", https://en.wikipedia.org/wiki/Axis%E2%80%93angle_representation#Rotation_vector (accessed Aug. 22, 2022).

[47] F. Chollet, "Deep learning with Python", *Shelter Island, New York: Manning Publications Co*, 2018.

[48] K. Ogata, "Modern Control Engineering", *5th Edition. Prentice Hall*, 2009.

[49] O. M. Omisore, S. Han, L. Ren, A. Elazab, L. Hui, T. Abdelhamid, N. A. Azeez, and L. Wang, "Deeply-learnt damped least-squares (DL-DLS) method for inverse kinematics of snake-like robots", *Neural Netw.*, Vol. 107, pp. 34–47, 2018, doi: 10.1016/j.neunet.2018.06.018.