TÜBİTAK

Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2023) 31: 342 – 359
© TÜBİTAK
doi:10.55730/1300-0632.3988

Boomerang algorithm based on swarm optimization for inverse kinematics of 6 DOF open chain manipulators

Okan DUYMAZLAR[1], **Dilşad ENGİN**[1,2,*]
[1]Mechatronics Program, Ege Higher Vocational School, Ege University, İzmir, Turkey
[2]Department of Electronics and Automation, Ege Higher Vocational School, Ege University, İzmir, Turkey

Received: 20.05.2022 • **Accepted/Published Online:** 01.02.2023 • **Final Version:** 23.03.2023

Abstract: In this study, a feasible swarm intelligence algorithm is proposed that computes the inverse kinematics solution of 6 degree of freedom (DOF) industrial robot arms, which are frequently used in industrial and medical applications. The proposed algorithm is named as Boomerang algorithm due to its recursive structure. The proposed algorithm aims to reduce the computation time to feasible levels without increasing the position and orientation errors. In order to reduce the computational time in swarm optimization algorithms and increase feasibility, an alternative definition method was used instead of the DH method in defining the robot arm kinematic configuration. The effect of the proposed alternative definition method in reducing the computational time is presented through example inverse kinematic analysis. The proposed algorithm was compared with 3 different particle swarm optimization (PSO) variants that include orientation in the inverse kinematic solution of 6 DOF robot arms. Comparative simulation studies were carried out with 20 randomly selected position and orientation data from the workspaces of PUMA 560 and ABB IRB120 manipulators to measure performance of the algorithms. Using the error and computation time values obtained from the simulation results, the algorithms are compared using the Wilcoxon nonparametric statistical test. When the simulation results are analysed by considering the calculation time, positioning accuracy and solution finding rates, it is seen that the Boomerang algorithm is more feasible than the other PSO variants. Verification of the simulation results, and the physical applications were carried out with the ABB IRB120 6 DOF robot arm. Simulation studies and experimental studies showed that the proposed algorithm may be an efficient method for inverse kinematics of time-critical applications.

Key words: Inverse kinematics, industrial robots, time efficient computing, PSO

1. Introduction

Positioning of open chain serial manipulators is of great importance in robotics, mainly in industrial and medical applications. Independent from force and moment, motion and position analysis of industrial robots are examined within the scope of robot kinematics which has two subbranches as forward and inverse kinematics [1]. In forward kinematics, identified joint variables of the industrial robot are used as inputs to calculate position and orientation of the end effector. Forward kinematics always reaches a single unique solution as opposed to inverse kinematics [2]. Since same position and orientation of end effector can be obtained with different joint values, inverse kinematic analysis convergences to a much more complicated and nonlinear structure with the increase of DOFs. Also inverse kinematic must reach a proper and precise solution for trajectory generation and path planning. Inverse kinematic analysis of robot arms is basically performed with analytical and numerical methods. Analytical methods depend on the robot configuration and use closed-form solutions

*Correspondence: dilsad.engin@ege.edu.tr that are algebraically obtained with the help of 4th or higher order polynomials or geometric functions [1]. However, since the algebraic method is only convenient for specific robot configurations, and the geometric method is inefficient and complex except for planar robots with 3 or less degrees of freedom, they cannot be used flexibly [2–4]. Alternatively, numeric methods mostly rely on the inverse Jacobian of the manipulator which causes singularities, and they are not cost-effective for higher degrees of freedom in computational means [5]. Also numeric algorithms using Jacobian-transpose to compute inverse kinematics may have a low convergence rate [6]. In this article, remainder sections are structured as follows. In Section 1, we explained the basics of robot kinematics and the use of PSO variants and drawbacks in inverse kinematics. In Section 2, we proposed a recursively structured PSO variant algorithm named Boomerang that reduces the computational time and achieves inverse kinematic solutions without increasing positioning and orientation errors of 6 DOF robot arms. Also, explained functioning of the proposed algorithm and its mathematical background with differences from the classical PSO algorithm. In Section 3, Boomerang algorithm and the algorithms to be compared are analyzed and how the parameters are determined is explained. We proposed to use an alternative method to define kinematic structure and to obtain fitness function used to reduce computation time and compared DH based fitness function over elapsed time in subsection 3.1. It was detailed how the selection of the optimum parameters for the Boomerang algorithm was done in subsection 3.2. Determination of the searcher, throw and total iteration parameters used in the Boomerang algorithm is presented. The studies carried out to determine the selection criteria and parameters of the PSO algorithms to be compared were explained in subsection 3.3. In subsection 3.4, selected PSO variants and Boomerang algorithm are compared over same dataset contains 20 sample pose from two different industrial robots. In conducting the comparative analysis, the inverse kinematic analyses of the PUMA560 and ABB IRB120 robot arms were simulated with the same computer hardware. Obtained results from the simulation environment were compared, analyzed with nonparametric Wilcoxon test and the results are interpreted. In Section 4, we presented the physical implementation results of the proposed algorithm on ABB IRB120 to verify the simulation results. In Section 5, the robot arm joint angles are calculated using the Boomerang algorithm for an assembly application requiring concentricity, and the results obtained with the application images are shown and interpreted. Finally, we discussed the obtained results in Section 6 and pointed out the achievements and possible drawbacks of the proposed algorithm in Section 7.

1.1. PSO variants for inverse kinematics

As an alternative to analytical and numerical methods, swarm intelligence-based optimization algorithms which mimic the hunting and searching behaviour of the swarms are also widely used for inverse kinematics and are the subject of research [7]. Optimization algorithms reduce inverse kinematics problem to an objective function and search optimal solution by minimizing the error. PSO algorithm [7] and its variants as PSO algorithm for 7-DOF serial robots [8], a modified artificial bee colony algorithm approach for optimization problem in inverse kinematics of robots [9], a fast successive approximation algorithm for the solution of the inverse position analysis of a general serial robot [10], PSO algorithm for 6-DOF serial robots [11], a soft-computing approach [12], improved PSO algorithms [13, 14] are mostly preferred optimization algorithms for inverse kinematics because of its simplicity and its efficient convergence structure [15]. In addition to inverse kinematics of relatively simply configured planar robots [16]–[20], PSO variants can also be used in the solution of higher degree of freedom industrial robots [8]–[13], [21]–[25]. For PSO variant based inverse kinematics, Euclidean distance in (1), which only takes the end effector position error into consideration, is widely used

[8, 9, 14, 17, 21, 23, 25, 26] as fitness function.

$$EuclideanDistance(\Delta P) = \sqrt{\Delta {P_x}^2 + \Delta {P_y}^2 + \Delta {P_z}^2}. \tag{1}$$

In Euclidean distance formula, ΔP notates the absolute difference between targeted Cartesian space parameters (x, y, and z) and the current parameters which are calculated using forward kinematics calculation over current joint variables. Another common fitness function used in swarm optimization based inverse kinematics [16, 22, 24, 27] is mean square errors (MSE) as in (2), which uses both position and orientation. MSE includes the absolute orientation error between the desired and current orientation of the end effector in terms of angular values.

$$\mathrm{MSE} = \sqrt{\Delta P_x^2 + \Delta P_y^2 + \Delta P_z^2 + \Delta R_x^2 + \Delta R_y^2 + \Delta R_z^2}. \tag{2}$$

For numerical and optimization algorithms, in each iteration, every data set, which contains joint angles, must be tested using forward kinematics. Along with the analytical and numerical methods, swarm optimization algorithms mostly use transformation matrix multiplication for forward kinematics calculations. Denavit-Hartenberg (DH) parameters are used to obtain transformation (T) matrices of each link from base to end effector as a generalized method [23] for forward kinematics as in (3) and (4).

$$\mathrm{T_i} = Rot_z(\theta) \cdot Trans_x(a) \cdot Trans_z(d) \cdot Rot_x(\alpha). \tag{3}$$

$$T_i = \begin{bmatrix} c\theta_i & -s\theta_i \cdot s\alpha_i & s\theta_i \cdot s\alpha_i & a_i \cdot c\theta_i \\ s\theta_i & c\theta_i \cdot c\alpha_i & -c\theta_i \cdot s\alpha_i & a_i \cdot s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

DH notation uses four parameters as a, α, d and θ to define $i.th$ link relative to $i-1.th$ link of serial robots. Parameters represent distance between $\mathrm{Z_i}$ and $\mathrm{Z_{i-1}}$, angular difference between $\mathrm{Z_i}$ and $\mathrm{Z_{i-1}}$ measured on the $\mathrm{X_i}$ axis projection and distance between $\mathrm{X_i}$ and $\mathrm{X_{i-1}}$, angular difference between $\mathrm{Z_i}$ and $\mathrm{Z_{i-1}}$ measured on the $\mathrm{X_i}$ axis projection, respectively. For simplicity, *cos* and *sin* functions also represented with 'c' and 's' letters, respectively. By a sequential multiplication of transformation matrices in (5), homogeneous transformation matrix H, which consist both position vector P in (6) and orientation matrix R in (7) are obtained.

$$H = T_0^n = T_0^1 \cdot T_1^2 \cdot T_2^3 \cdots T_{n-1}^n = \begin{bmatrix} R_{3x3} & P_{3x1} \\ 0_{1x3} & 1 \end{bmatrix} \tag{5}$$

$$P = [P_x \, P_y \, P_z]^T \tag{6}$$

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = \begin{bmatrix} n_x & s_x & a_x \\ n_y & s_y & a_y \\ n_z & s_z & a_z \end{bmatrix} \tag{7}$$

Utilizing DH notation to define kinematic structure of the robot and checking the positioning errors with forward kinematics, which uses matrix multiplication or the obtained final equations, increases the computational load of the recursive or iterative algorithms in each loop. In PSO algorithms, which already have local minimum and early convergence problems [13], an increase in computation time is added as a drawback for inverse kinematic cases. Therefore, most challenging problems for PSO based inverse kinematics algorithms are achieving the fitness function properly and reducing the computational time [26]. Nonlinearity of inverse kinematics problems increases with higher DOF nonplanar robots and orientation becomes more important along with the positioning of the end effector. Since construction of a proper fitness function that includes both position and orientation is challenging, fixed orientation may be used or it may completely be excluded from the fitness function for some applications [15, 18, 23]. However, the computed joint angles may provide an arbitrary access to a point in three-dimensional space when we exclude orientation from the inverse kinematics. The computational time reaches to about 2 seconds where PSO variant computes solutions for both position and orientation [12, 21], and then the PSO variant becomes inefficient for applications where computational time is of importance. PSO variants that use orientation in inverse kinematic analysis has a high computational load but fixing or excluding the orientation to reduce computation time makes PSO variant algorithms unfeasible for most industrial applications.

2. Boomerang algorithm for inverse kinematics

In this study, we proposed an algorithm named as Boomerang algorithm that includes orientation in the inverse kinematic analysis of industrial robots and reduces the computation time to acceptable levels for applications where time efficiency is important.

2.1. An alternative definition of kinematic structures

The robot configurations defined with DH parameters and forward kinematic calculations obtained with matrix multiplications or with expressions containing too many trigonometric functions which are obtained once outside of the loop, are used in each iteration for each unique particle. Consequently, the computational load and time increases. To reduce the computational load, we propose an algorithm and named it as Boomerang algorithm which uses ortho-parallel manipulator with spherical wrist (OPW) parameters [28], an analytical method that restricts the robot arms to a standard offset position. With the OPW method, robot arms up to 6 degrees of freedom with Euler wrist configuration, which is mostly preferred in industrial and medical applications [1, 29] can be defined with only 7 scalar values as

- L1 is vertical distance from base to 2nd joint axis
- L2 is vertical distance between 2nd and 3rd joint axes
- L3 is vertical distance between 3rd and 5th joint axes
- L4 is vertical distance between 5th joint axis and end effector
- O1 is horizontal distance between the base joint and the second joint axis
- O2 is horizontal distance between the 3rd and 4th joint axes
- O3 is horizontal distance between the base joint and the 3rd joint axis measured in the back view of the robot.

Physical equivalents of the dimensions used in the identification of robot configuration are shown on Figure 1 with Kuka KR30 and ABB IRB120 robots.

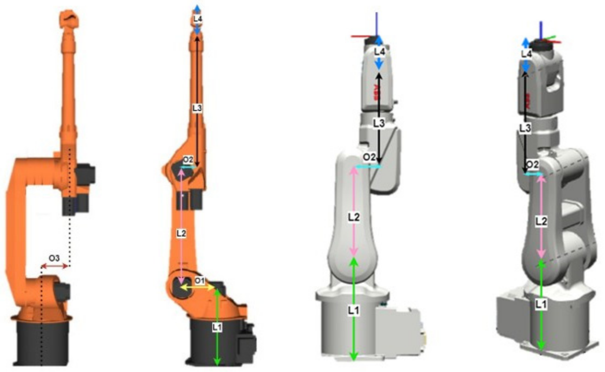


Figure 1. OPW parameters and physical meanings via KUKA KR30 and ABB IRB120 industrial robots.

Current position of the robot is calculated with (8) to (16) for each solution set obtained in iteration. P_x, P_y and P_z are Cartesian space coordinates of the end effector and θ_n is joint angular value of the n_{th} joint.

$$X_k = -O_3 s\theta_1 + c\theta_1 (O_1 + s\mu)\sqrt{{O_2}^2 + {L_3}^2} + L_2 s\theta_2 \tag{8}$$

$$Y_k = O_3 c\theta_1 + s\theta_1 (O_1 + s\mu)\sqrt{{O_2}^2 + {L_3}^2} + L_2 s\theta_2) \tag{9}$$

$$Z_k = L_1 + c(\mu)\sqrt{{O_2}^2 + {L_3}^2} + L_2 c\theta_2), \tag{10}$$

where

$$\mu = \theta_2 + \theta_3 + atan2(O_2, L_3) \tag{11}$$

$$X_j = L_4(s\theta_{23} c\theta_1 c\theta_5 - s\theta_1 s\theta_4 s\theta_5 + c\theta_{23} c\theta_1 c\theta_4 s\theta_5) \tag{12}$$

$$Y_j = L_4(s\theta_{23} c\theta_5 s\theta_1 + c\theta_1 s\theta_4 s\theta_5 + c\theta_{23} c\theta_4 s\theta_1 s\theta_5) \tag{13}$$

$$Z_j = L_4 \cdot (c\theta_{23} \cdot c\theta_5 - s\theta_{23} \cdot c\theta_4 \cdot s\theta_5), \tag{14}$$

where

$$\theta_{23} = \theta_2 + \theta_3 \tag{15}$$

$$P_x = X_k + X_j; P_y = Y_k + Y_j; P_Z = Z_k + Z_j. \tag{16}$$

Equations (17) to (19) are used to derive R matrix in (20) and from the R matrix, orientation of the end effector is calculated with Roll, Pitch and Yaw Euler angles which are shown in (21) to (23), respectively.

$$R_k = \begin{bmatrix} c(\theta_2+\theta_3)c\theta_1 & -s\theta_1 & s(\theta_2+\theta_3)c\theta_1 \\ c(\theta_2+\theta_3)s\theta_1 & c\theta_1 & s(\theta_2+\theta_3)s\theta_1 \\ -s(\theta_2+\theta_3) & 0 & c(\theta_2+\theta_3) \end{bmatrix} \tag{17}$$

$$R_j = \begin{bmatrix} c_4c_5c_6 - s_4s_6 & -c_4c_5s_6 - s_4c_6 & c_4s_5 \\ s_4c_5c_6 + c_4s_6 & -s_4c_5s_6 + c_4c_6 & s_4s_5 \\ -s_5c_6 & s_5s_6 & c_5 \end{bmatrix}, \tag{18}$$

where

$$c_i = c\theta_i \, and \, s_i = s\theta_i \tag{19}$$

$$\mathrm{R} = R_k + R_j \tag{20}$$

$$\varphi = atan2(r_{21}, r_{11}) \tag{21}$$

$$\vartheta = atan2(-r_{31}, \sqrt{{r_{32}}^2 + {r_{33}}^2}) \tag{22}$$

$$\psi = atan2(r_{32}, r_{33}). \tag{23}$$

2.2. Boomerang algorithm structure and differences from PSO variants

Boomerang algorithm is a PSO variant with recursive structure for inverse kinematics of industrial robots that reinitializes angle sets. The proposed algorithm imitates boomerangs that can return to the point where it started as a way of working. In Boomerang algorithm, presented in Figure 2a, the predetermined total number of iterations is divided into subiterations and each subiteration is called a throw. Thus, the throw is repeated in cases where no suitable solution for inverse kinematics can be found with the optimized values of randomly determined starting angle sets.The PSO algorithm is given in Figure 2b for comparison of the two algorithms.

The algorithm uses a similar method with the PSO variant called as RPSO (restart particle swarm optimization) to prevent from premature convergence that consumes time. But RPSO keeps the best particle of former iteration [30] in contradistinction to Boomerang algorithm that eliminates all joint values at each throw. In Boomerang algorithm, there are no limitations for reinitialization of random angle sets except physical joint limits which are specific for the robot configuration.

2.3. Essential points of the proposed algorithm

We used weighted Euclidean norm as in (24) for fitness function which includes both position and orientation. Through empirical observations, weight coefficient of the position error, ΔP was chosen as 60 while weight coefficient of orientation error, ΔRPY was 40.

$$\text{fitness} = \|60 \cdot \Delta P + 40 \cdot \Delta RPY\|_2 \tag{24}$$

For a semirandom change of the joint variables in each iteration, maximum, (25), and minimum, (26), amount of change is determined by taking solution set number into consideration at the beginning of the algorithm. θ_{max} and θ_{min} indicate upper and lower physical constraints of joints and ξ indicates number of discrete solution sets, which contains joint variables needed as much for the degree of freedom of the robot. For a standard PSO algorithm ξ variable corresponds to the number of particles and D corresponds to the particle velocity.

$$D_{max} = (\theta_{max} - \theta_{min})/(5 \cdot \xi) \tag{25}$$

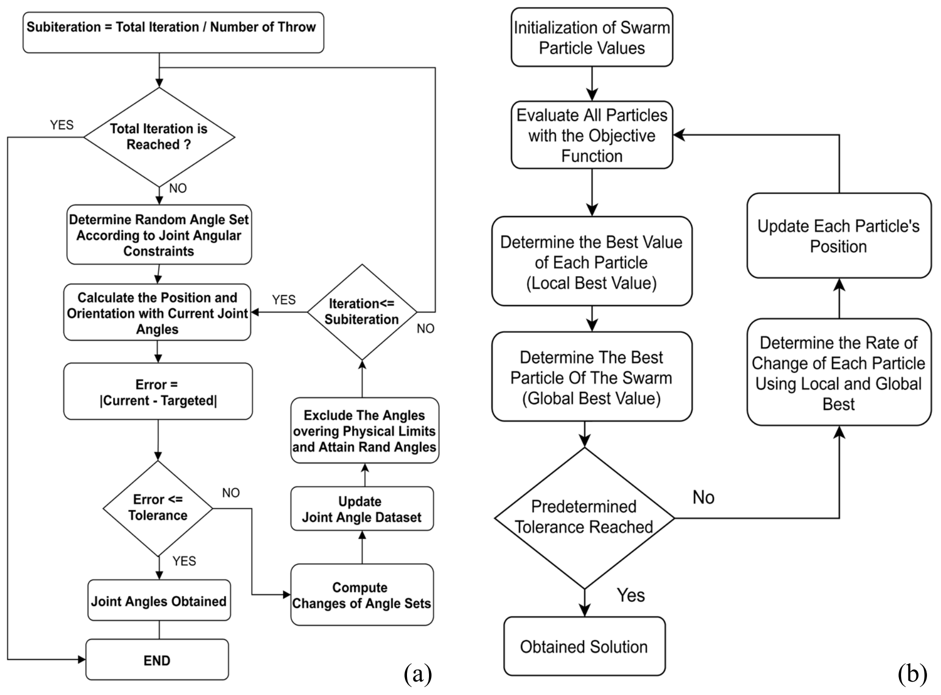

Subiteration = Total Iteration / Number of Throw
Total Iteration is Reached ?
YES
NO
Determine Random Angle Set According to Joint Angular Constraints
Calculate the Position and Orientation with Current Joint Angles
Error = |Current - Targeted|
Error <= Tolerance
NO
YES
Joint Angles Obtained
END
Iteration<= Subiteration
YES
NO
Exclude The Angles overing Physical Limits and Attain Rand Angles
Update Joint Angle Dataset
Compute Changes of Angle Sets
(a)
Initialization of Swarm Particle Values
Evaluate All Particles with the Objective Function
Determine the Best Value of Each Particle (Local Best Value)
Determine The Best Particle Of The Swarm (Global Best Value)
Predetermined Tolerance Reached
No
Yes
Obtained Solution
Update Each Particle's Position
Determine the Rate of Change of Each Particle Using Local and Global Best
(b)

Figure 2. (a) Boomerang algorithm, (b) PSO algorithm flow charts.

$$D_{min} = (\theta_{max} - \theta_{min})/(-5 \cdot \xi) \tag{26}$$

For calculating the amount of change in robot arm joint angles, (27) derived from PSO algorithm is used with a weight constant w, which depends on total iteration number λ in (28). $\Delta\theta$ is the amount of change in angular position, i is the current iteration step, and the random numbers used in the equations are shown with 'rn', the range of values they can take is specified as subscript. Best fitness score obtained by each solution set until the current iteration is named as L_b, and the joint variables that give the most accurate kinematic solution among all solution sets are named as G_b. If the computed angular change amounts are not between the limits determined at the beginning of the algorithm, they are equaled to random limits as presented in (29) and (30).

$$\Delta\theta_{i+1} = w \cdot \Delta\theta_i + rn_{0:2} \cdot (L_b - \theta) + rn_{0:2} \cdot (G_b - \theta) \tag{27}$$

$$w = 1 - \lambda/1000 \tag{28}$$

$$\Delta\theta_{i+1} \leq D_{min} \rightarrow \Delta\theta_{i+1} = D_{min} \tag{29}$$

$$\Delta\theta_{i+1} \geq D_{max} \rightarrow \Delta\theta_{i+1} = D_{max} \tag{30}$$

If the updated joint values in (31) exceed the physical limits of the joint, a random value assignment is made within the physical joint limits as shown in (32) and (33).

$$\theta_{i+1} = \theta_i + \Delta\theta_{i+1} \tag{31}$$

$$\theta_{i+1} \geq \theta_{max} \rightarrow \theta_{i+1} = \theta_{max} \cdot rs_{0:1} \tag{32}$$

$$\theta_{i+1} \leq \theta_{min} \rightarrow \theta_{i+1} = \theta_{min} \cdot rs_{0:1} \tag{33}$$

2.4. Differences between the optimization characteristic of PSO and Boomerang algorithms

The convergence patterns of a PSO and the proposed Boomerang algorithm for the inverse kinematics solution are compared and optimization process for reducing the number of iterations is presented graphically in Figures 3 and 4. In order to graphically illustrate the solution search characteristics of a standard PSO algorithm and the recursive Boomerang algorithm, the inverse kinematic problem of a basic 2 DOF planar robot is solved by both algorithms and the results are shown in Figure 3. This example, which is easy to interpret graphically and to be solved by any optimization algorithm since it does not contain orientation data, is examined. If we focus on a single optimization process as in Figure 4, it is seen that PSO algorithm consumes time in premature convergences while reaching the desired tolerance. In Boomerang algorithm, for a single optimization case, primary purpose for reducing the number of iterations and the computational time is to search for a better starting point by restarting the solution with the thought that current variable values are not suitable.

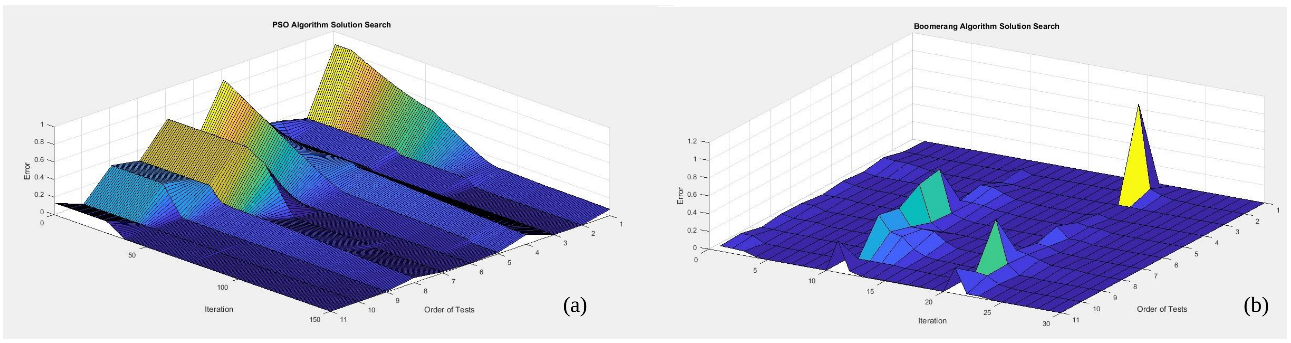


Figure 3. Searching pattern of (a) PSO algorithm, (b) Boomerang algorithm sampled with 11 discrete tests each.

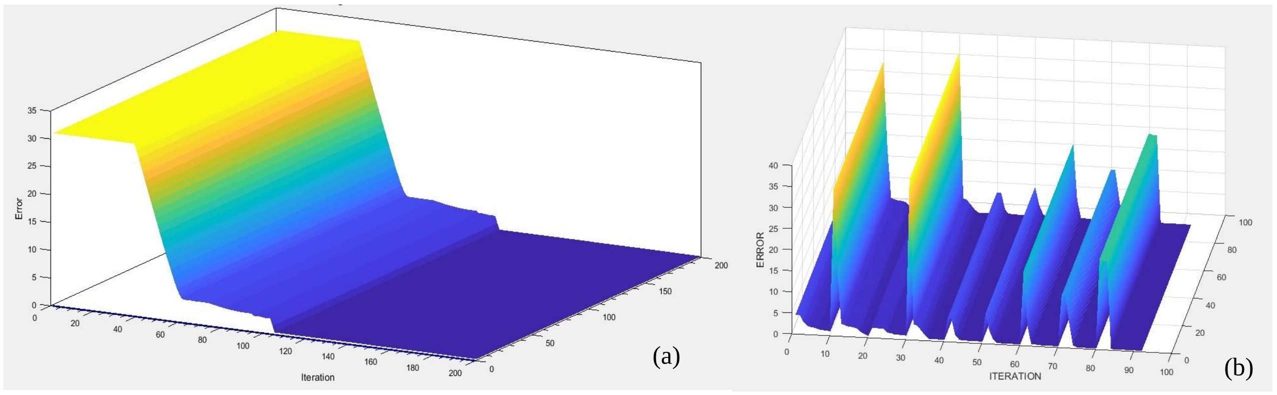


Figure 4. Optimization process of (a) PSO algorithm, (b) Boomerang algorithm.

3. Simulation studies on comparison of algorithms and selection of parameters

As with optimization algorithms in general, the success of particle swarm optimization variants varies depending on the parameters chosen and the fitness function used in the case. For the theoretical and simulation comparison of the Boomerang algorithm, we used three different PSO variants and tested all algorithms with the same position-orientation pair. Since the algorithm we designed is an inverse kinematics algorithm that includes orientation, we gave importance to select PSO variants from the literature for the same case. For this reason, in order to make an objective comparison, the algorithms were implemented and the results were obtained by considering the number of iterations and population sizes used by the authors in each publication. In subsection 3.1, we analyzed OPW method we used in Boomerang algorithm and DH based forward kinematics functions and shared the effects on computational load of algorithms. In subsection 3.2, we decided the optimum parameters for proposed algorithms and shared results and in subsection 3.3 we gave selected PSO variants major points and parameter selection methods. We tested all the algorithms with same dataset and measured the mean absolute errors (MAE) for position and orientation and average computation time for each data and analyzed results with nonparametric Wilcoxon test which is suggested for a fair comparison in the literature [31–33].

3.1. Comparison of OPW and DH based objective functions

For creating the fitness function to be used in the developed algorithm, the effect of defining the manipulator with DH or OPW and as a result, the effect of the forward kinematic functions used to improve the results during optimization on the computational load was examined. Both methods give single solution set for given joint angles but OPW uses less complex equation than DH based forward kinematic equations. And algorithms use obtained forward kinematic equations in each iteration to reduce the positioning error may increase the computation time. Therefore, we investigated whether the OPW method used in the developed algorithm reduces the computational load compared to the DH parameters, since the formulae used in these forward kinematic calculations of the robots up to 6 DOF are less complex with less trigonometric expressions. One hundred randomly selected samples from the workspace were tested using the Boomerang algorithm with the same parameters by only changing the DH and OPW based forward kinematic equations. The graphical results of the comparative analysis given in Figure 5 reveal that a significant advantage is provided in computation times when the objective function defined by OPW is used.

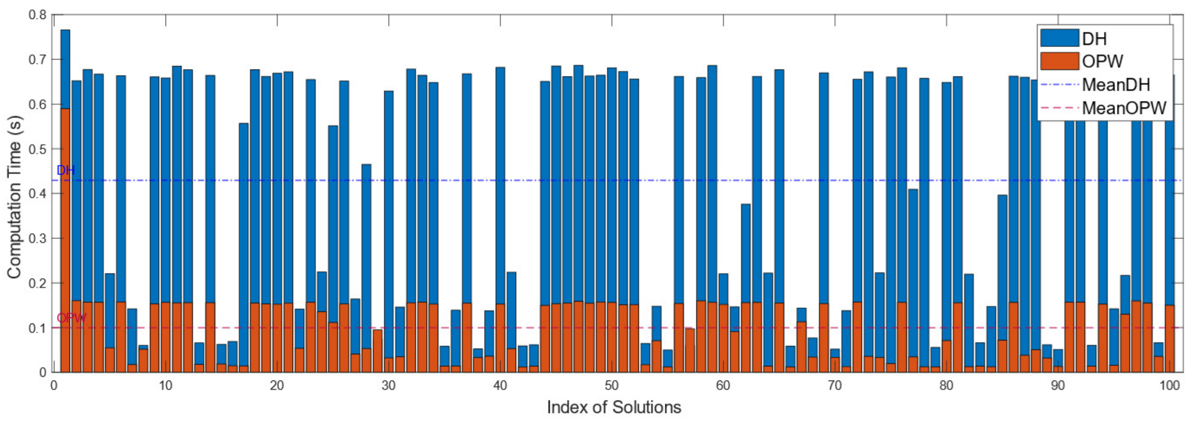


Figure 5. Effects of OPW and DH based fitness functions on computation times.

3.2. Determining the optimum parameters of the Boomerang algorithm

To determine the searcher, throws and total iteration parameters to be used in the Boomerang algorithm, analyses were carried out. As seen in Figure 6, in the inverse kinematics problem, running the whole iteration with a single loop or using the throw number in high amounts increases the computation time. Similarly, the increase in the number of multiple particles, called searcher, which searches for the solution simultaneously in the loop, negatively affects the calculation time when the number of throws is kept constant. To determine the optimum values of the searcher, throws and total iteration parameters of the Boomerang algorithm to minimize the computation time, tests were performed using the position and orientation data of a pose. The aim of the tests is to find the parameters that minimize the computation time. At first step, using only one particle, total iteration, and subiteration (throws) values were obtained from each throw-total iteration value pair that produced solutions with minimum computation times. The points where minimum computation times were obtained from the tests are indicated by the red arrow in Figure 6a. In the next step, using the same position-orientation input, the regions where throw-searcher pairs reached a successful solution with minimum computation time were determined. The results of the second test, where the optimum computation times were obtained, are shown in Figure 6b with red areas. As a result of the intersection of both analyses, the searcher, throw and total iteration parameters were chosen as 10, 8, and 5500, respectively.

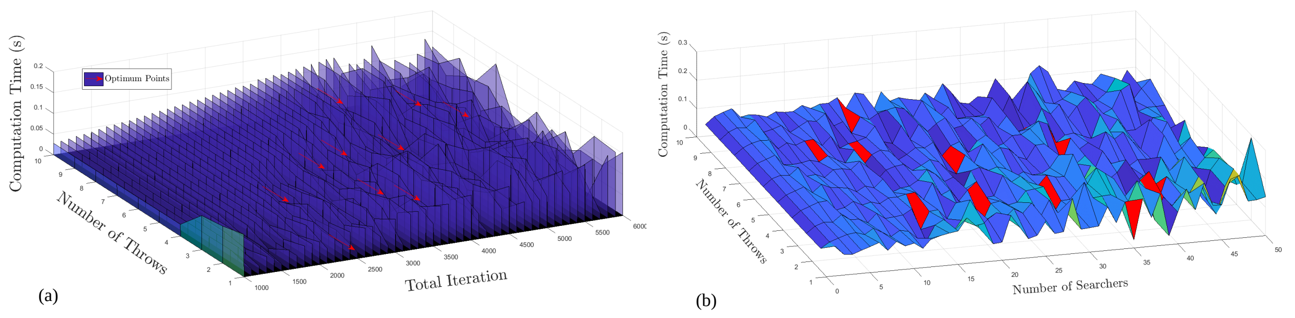


Figure 6. (a) Optimal throws and total iteration parameters based on total computation time, (b) effects of variations of throws and searcher values on computation time and selected local minimum areas.

3.3. Structure of the algorithms to be compared and selection of parameters

Boomerang algorithm has been compared with 3 different PSO variants [12, 22, 24] that have been used for inverse kinematic analysis of 6 DOF robot arms. All the compared algorithms take orientation into consideration for computing the inverse kinematic solution. The computation time and convergence characteristics of particle swarm optimization algorithms vary according to the population size, the objective function and the maximum number of iterations. For this reason, in order to make an objective comparison, the algorithms were implemented and the results were obtained by considering the number of iterations and population sizes used by the authors in each publication. PSO1 variant, developed by Nguyen et al. [22], uses the standard PSO algorithm with the fitness function in (34), that is proposed for a 6 DOF industrial robot. As suggested in the study, the population size was set to 300 and the number of iterations was limited to 500.

$$f_{PSO1} = \sqrt{60(\Delta P_{xyz}) + \Delta R_{xyz}}, \tag{34}$$

where

$$\Delta P_{xyz} = \Delta Px^2 + \Delta Py^2 + \Delta Pz^2 \tag{35}$$

$$\Delta R_{xyz} = \Delta Rx^2 + \Delta Ry^2 + \Delta Rz^2. \tag{36}$$

PSO(2) variant uses a weighted fitness function given in (37) to (40) which is proposed for 6 DOF Puma 560 robot arm by Lopez-Franco et al. [12]. Algorithm terminated as suggested if the algorithm achieves the total number of iterations (1000) or the fitness value reaches a value of tolerance and the population size is set to 30 individuals.

$$P_{error} = \sqrt{\Delta P_{xyz}}/\sqrt{x^2 + y^2 + z^2} \tag{37}$$

$$R_{error} = \sqrt{\Delta R_{xyz}}/\sqrt{Rx^2 + Ry^2 + Rz^2} \tag{38}$$

$$\rho = 0.7 \cdot e^{-R_{error}} + 0.3 \tag{39}$$

$$f_{PSO2} = \rho \cdot P_{error} + (1 - \rho) \cdot R_{error} \tag{40}$$

The PSO(3) variant developed by Alkayyali and Tutunji [24], uses the same fitness function of Lopez–Franco et al.[12], along with the rate of change modification in (41) with total iteration limited to 1000. Particle velocity weight constant denoted as K, was recalculated in each iteration (43). Variables that take random values are denoted by the letters rn, while subscripts indicate the value range in (44) and new joint angles are updated as given in (45).

$$V_i = K \cdot [w \cdot V_{i-1} + rn_i \cdot (L_b - X_{i-1}) + rn_i \cdot (G_b - X_{i-1})], \tag{41}$$

where

$$rn_i = rn_{2:4} \cdot rn_{0:1} \tag{42}$$

$$K_i = 2 \cdot rn_{0:1} / \left| 2 - r_i - \sqrt{{r_i}^2 - 4 \cdot r_i} \right| \tag{43}$$

$$r_i = rn_{0:1} \cdot rn_{2:4} + rn_{0:1} \cdot rn_{2:4} \tag{44}$$

$$NewJointAngles = FormerJointAngles + V_i. \tag{45}$$

3.4. Comparative simulation studies of the algorithms

Six DOF ABB IRB120 and PUMA 560 industrial robot arms with anthropomorphic configurations were used for simulations. By using the open-source Robot Kinematics Simulator (RoKiSim) software tool, ten random position and orientation data for two industrial robots were obtained which are presented in Table 1.

Existence of the inverse kinematic solution is guaranteed by taking the selected positions and orientations from the workspace. Boomerang algorithm and PSO variants were coded as proposed by Alkayyali and Tutunji [24], Lopez-Franco et al. [12], and Nguyen et al. [22], then run in same environment [MATLAB] on same hardware for a fair comparison. For each position and orientation input, all the algorithms were simulated ten times for same data and MAEs obtained as shown in Table 2. Mean values of success rate of reaching

Table 1. Selected positions and orientations for PUMA 560 and ABB IRB120 robots.

	PUMA 560						ABB IRB120					
Data	**X**	**Y**	**Z**	**Roll**	**Pitch**	**Yaw**	**X**	**Y**	**Z**	**Roll**	**Pitch**	**Yaw**
1	431	139.7	489.58	0	0	0	374	0	630	90	0	45
2	920.58	139.7	0	0	90	0	374	0	630	0	90	0
3	618.04	139.7	208.48	0	30	45	239	0	593.82	0	90	0
4	431	139.7	489.58	90	0	45	240	0	590	0	90	45
5	588.86	500.98	73.11	–43,24	58.51	–70.53	264.45	–264.45	630	0	90	45
6	704.32	405.01	84.30	–160	0	180	294.03	270.22	269.67	–126.43	16.40	169.03
7	748.48	435.64	155.40	32.10	73.96	–2.83	–70	0	934	0	0	0
8	765.13	–154.38	252.52	26.99	–7.71	28.65	302	0	558	180	0	180
9	506.31	139.7	–482.99	–180	0	180	302.84	158.28	259.04	80.96	–46.97	178.61
10	300	150	400	0	0	0	0	362.06	395.97	–180	0	165

a solution, computational time in seconds and errors (mm for cartesian space x, y, z positions and degree for $Roll$, $Pitch$, Yaw orientations) were shown as an abstraction of the collected results of Table 2 in Table 3. Concerning the comparison parameters, success rate indicator measures whether the algorithm reached a solution until termination criteria, computational time indicates the elapsed time until termination, and error indicates position-orientation difference between the obtained and desired outputs. Simulations for 20 different position and orientation data showed that the Boomerang algorithm achieves the second-best positioning and minimum orientation error values within the compared PSO variants as depicted in Table 3.

A feasibility function η as in (46) is established using mean absolute errors, calculation time and success rate to measure whether the algorithms are suitable for field applications.

$$\eta = 1 - \frac{Succ.Rate^{-1} + Comp.Time + MAE}{3} \tag{46}$$

Feasibility analysis of the algorithms resulted in 60.13% for PSO1, 73.00% for PSO2, 75.97% for PSO3, and 85.79% for the Boomerang algorithm revealing that Boomerang algorithm is more suitable than the other PSO variants if computational time, position and orientation errors and success rate are correlated.

In order to confirm the results shown and interpreted in Tables 2 and 3 with a nonparametric statistical method, Wilcoxon rank test was used. Wilcoxon nonparametric test is convenient method to measure statistically if the proposed algorithm really differs meaningfully with the compared ones. H1 hypothesis is that there is a meaningful difference between the compared results and H0 is the null hypothesis. A confidence level of 95% (gamma is 0.05) was used and each PSO variant was individually subjected to the Wilcoxon nonparametric test with the Boomerang algorithm using the values in Table 2. The results of the statistical analysis are presented in Table 4. As can be seen from Table 4, similar findings to the previous conclusions are obtained. As the Wilcoxon comparative analysis confirms, there is no significant superiority between the PSO(3) variant and the Boomerang algorithm in positioning accuracy, but in all other parameters the Boomerang algorithm is better.

Table 2. MAEs and computation time of each algorithm's inverse kinematic solutions for the given poses in Table 1.

	MAEs of position (mm)				MAEs of orientation (°)				Mean value of computation time (s)			
Data	**BPSO**	**PSO(1)**	**PSO(2)**	**PSO(3)**	**BPSO**	**PSO(1)**	**PSO(2)**	**PSO(3)**	**BPSO**	**PSO(1)**	**PSO(2)**	**PSO(3)**
1	1.09E-04	1.23E-13	1.12E-01	6.82E-03	3.84E-04	1.21E-14	5.34E+00	9.75E-01	8.48E-01	1.20E+01	9.83E+00	14.32
2	3.36E-02	4.06E+00	5.99E-01	5.36E-02	2.91E-01	7.41E+00	1.30E+00	6.40E-01	2.23E+00	1.02E+01	1.08E+01	12.21
3	7.82E-05	2.18E-02	2.66E-02	1.25E-07	7.38E-04	1.80E+01	7.47E-01	3.46E-01	1.48E-01	1.21E+01	5.30E+00	12.11
4	3.32E-04	1.49E-04	4.96E-02	4.28E-12	6.70E-04	8.88E-02	4.16E+00	4.18E-01	1.38E-01	1.16E+01	4.99E+00	13.27
5	2.18E-04	3.03E-02	1.74E-02	6.16E-14	6.47E-04	1.52E+01	9.65E-01	1.64E-01	1.78E-01	1.05E+01	5.13E+00	13.26
6	4.57E-04	1.36E-01	1.16E-01	5.68E-14	6.53E-04	3.23E+01	2.31E+01	2.12E+00	4.37E-01	4.39E+00	5.22E+00	13.58
7	7.70E-02	3.35E-01	3.03E-02	3.92E-05	1.80E-01	2.52E+01	7.52E+00	5.24E-01	2.85E+00	2.91E+00	5.11E+00	12.06
8	4.66E-04	5.23E-02	1.64E-02	9.47E-15	4.31E-04	5.22E+00	3.44E-01	6.02E-02	6.58E-01	3.05E+00	5.14E+00	14.91
9	2.68E-03	4.98E-03	4.16E-02	9.27E-04	4.19E-03	3.10E-01	4.59E+00	3.53E+00	2.27E+00	2.95E+00	8.52E+00	12.53
10	4.99E-04	8.01E-02	1.12E-01	6.82E-03	3.42E-04	2.51E+01	5.34E+00	9.75E-01	2.26E-01	5.17E+00	5.07E+00	14.08
11	3.50E-03	1.02E+01	3.08E-02	3.92E-07	5.39E-03	3.94E+01	1.26E+00	1.30E-01	3.10E-01	1.38E+01	1.33E+01	13.84
12	1.42E-03	1.87E-01	1.49E-01	9.40E-10	6.86E-03	2.42E+01	6.81E+00	7.29E+00	6.16E-01	1.34E+01	6.60E+00	13.41
13	4.55E-03	2.33E-01	9.46E-01	3.58E-13	3.66E-03	1.48E+01	5.56E-01	7.50E-01	3.79E-01	1.25E+01	1.00E+01	12.51
14	2.21E-01	5.02E-03	4.73E-02	5.20E-07	3.00E-01	5.74E-01	1.59E+01	2.18E+00	1.74E+00	1.43E+01	4.88E+00	14.35
15	9.48E-02	8.23E-02	4.85E-02	4.55E-08	3.05E-01	5.14E+01	3.29E+00	5.90E+00	1.33E+00	1.38E+01	5.13E+00	13.76
16	3.79E-03	2.80E-02	3.05E-02	5.20E-09	6.01E-03	1.39E+01	6.32E+00	5.23E+00	1.10E-01	2.03E+01	5.18E+00	20.31
17	4.08E-03	1.20E+00	1.49E-01	1.60E-06	3.71E-03	1.94E+01	6.81E+00	6.08E+00	2.60E-01	1.43E+01	5.41E+00	14.33
18	1.58E-03	4.28E-02	3.19E-02	1.34E-05	6.99E-03	1.05E+01	5.56E+00	9.98E+00	9.16E-02	1.38E+01	5.18E+00	13.83
19	1.52E-03	4.08E-02	2.63E-02	7.58E-14	6.26E-03	1.34E+01	4.73E+00	1.37E+01	1.56E-01	1.32E+01	5.31E+00	13.21
20	3.28E-03	6.83E-03	3.15E-02	4.62E-14	5.13E-03	6.26E+00	1.69E+01	9.57E+00	6.75E-01	1.34E+01	5.26E+00	13.37

Table 3. Simulation results of the tested algorithms over 20 discrete case.

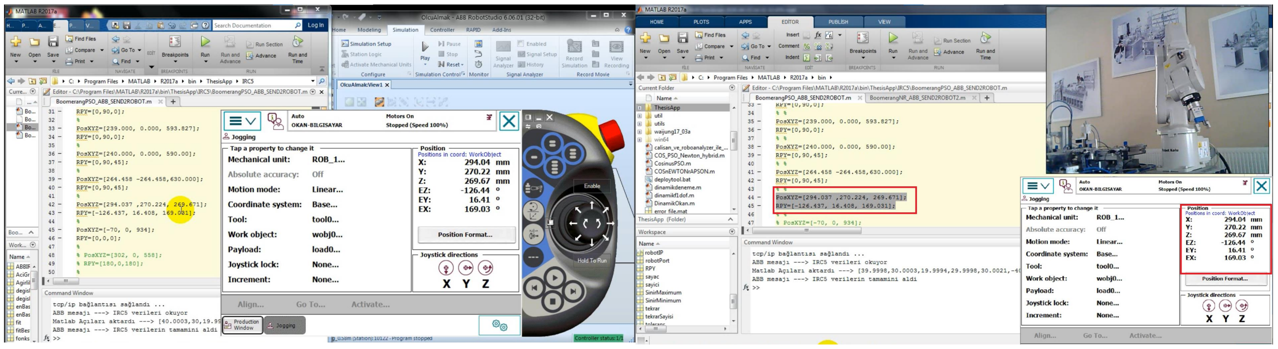
Algorithm	X	Y	Z	Roll	Pitch	Yaw	Position (mm)	Orientation (°)	Success rate	Computation time (s)
PSO1	0.716	1.018	0.015	4.492	5.016	6.031	1.7492	15.5383	100%	9.7207
PSO2	0.197	0.016	0.058	1.022	2.976	4.955	0.2702	8.9538	80%	6.5708
PSO3	0.009	0.002	0.001	0.606	2.319	4.199	0.0101	7.1235	85%	13.7625
Boomerang	0.011	0.027	0.002	0.006	0.065	0.005	0.0408	0.0761	95%	0.7825

Table 4. Nonparametric statistical comparison of the algorithms with $\alpha : 0.05$.

Compared algorithms		Position				Orientation				Computation time			
Algorithm 1	**Algorithm 2**	**R^+**	**R^-**	ρ	**Result**	**R^+**	**R^-**	ρ	**Result**	**R^+**	**R^-**	ρ	**Result**
Boomerang	**PSO (1)**	3	89	1,1368E-03	H1:Boomerang	3	89	5,5788E-05	H1:Boomerang	3	89	4,7846E-05	H1:Boomerang
Boomerang	**PSO (2)**	11	89	6,0396E-03	H1:Boomerang	0	90	9,5367E-07	H1:Boomerang	0	99	9,5367E-07	H1:Boomerang
Boomerang	**PSO (3)**	42	48	9,8520E-01	H0:Null	0	70	9,5367E-07	H1:Boomerang	0	96	9,5367E-07	H1:Boomerang

4. Experimental results

Besides simulation results, success of the Boomerang algorithm was tested on physical systems. Firstly, the positions and orientations used in the analysis of the robot arm with simulation were tested with the physical robot shown in Figure 7. Secondly, the spring element concentric placement, which is a precise assembly operation, was performed with joint angles calculated by the Boomerang algorithm. For both experiments, joint angles calculated by Boomerang algorithm for the desired position and orientation were transferred to the IRC5 robot controller using TCP/IP socket communication.



```
RPY=[0,90,0];
%
PosXYZ=[239.000, 0.000, 593.827];
RPY=[0,90,0];
%
PosXYZ=[240.000, 0.000, 590.00];
RPY=[0,90,45];
%
PosXYZ=[264.458 -264.458,630.000];
RPY=[0,90,45];
%
PosXYZ=[294.037 ,270.224, 269.671];
RPY=[-126.437, 16.408, 169.031];
%
PosXYZ=[-70, 0, 934];
RPY=[0,0,0];
%
% PosXYZ=[302, 0, 558];
% RPY=[180,0,180];
%
```

```
tcp/ip bağlantısı sağlandı ...
ABB mesajı ---> IRC5 verileri okuyor
Matlab Açıları aktardı ---> [39.9998,30.0003,19.9994,29.9998,30.0021,-4
ABB mesajı ---> IRC5 verilerin tamamini aldi
>>
```

Auto OKAN-BILGISAYAR — Motors On, Stopped (Speed 100%)

Jogging

Tap a property to change it	
Mechanical unit:	ROB_1...
Absolute accuracy:	Off
Motion mode:	Linear...
Coordinate system:	Base...
Tool:	tool0...
Work object:	wobj0...
Payload:	load0...
Joystick lock:	None...
Increment:	None...

Position	Positions in coord: WorkObject	
X:	294.04	mm
Y:	270.22	mm
Z:	269.67	mm
EZ:	-126.44	°
EY:	16.41	°
EX:	169.03	°

(a) (b)

Figure 7. Application of the inverse kinematic solution of the Boomerang algorithm on ABB IRB120 robot in operation with (a) RobotStudio Flexpendant runtime screen, (b) Physical robot and Flexpendant screen.

4.1. Physical application of the algorithm

In this application, the algorithm runs on a client computer and the robot controller works as a server. The desired position and orientation were solved by Boomerang algorithm to obtain the joint angles. These calculated joint angles were then transferred to IRC5 robot controller via TCP/IP socket communication. The IRC5 that operates as a server waits for the joint angle data by continually listening TCP/IP port. When the joint angles are transferred to the server, the client is informed that the transfer has been successfully performed and the robot arm starts moving to the calculated joint angles simultaneously. Final position and orientation by the robot arm were observed through the Flexpendant input-output device in Figure 7 and compared with the desired position and orientation. Performed tests verified that the proposed algorithm successfully solves the inverse kinematics of industrial robots in various cases as in a 6 DOF industrial robot. Consequently, the physical application of Boomerang algorithm on a physical robotic system proved that the proposed algorithm could be an alternative method for inverse kinematics of industrial robots.

4.2. Assembly application

After computational accuracy of the developed algorithm was demonstrated experimentally, we also used the algorithm in precision assembly operations on a case. Boomerang algorithm was used for the assembly operation of the prototype, which consists of a cylindrical metal part used as a base, a spring used as a damping element and a cover. The inner diameter of the spring to be assembled is 13.25 mm and the outer diameter of the socket of the base piece is 12.85 mm. The position and orientation information shown on the Flexpendant screen in Figure 8a shows the position required to assemble the two parts concentrically. However, due to the difference in diameter between the base and the spring, a deviation of 0.225 mm can be accepted in this operation.

With another saying, considering the diameter difference of the two parts, the assembly operation has an error tolerance of 0.225 mm, and the assembly operation fails in case of positioning errors exceeding the tolerance. The position and orientation data required for concentric centering were entered into the Boomerang algorithm and the inverse kinematic solution was performed and the robot arm moved to the position shown in Figure 8b with the angles obtained. As can be seen from the Flexpendant screen in Figure 8b, the desired position and orientation errors are much better than the precision required for assembly. As shown in Figure 8c, the spring assembly handled correctly with positioning errors 0.1 mm, 0 mm and 0.1 mm for x, y and z coordinates, respectively.

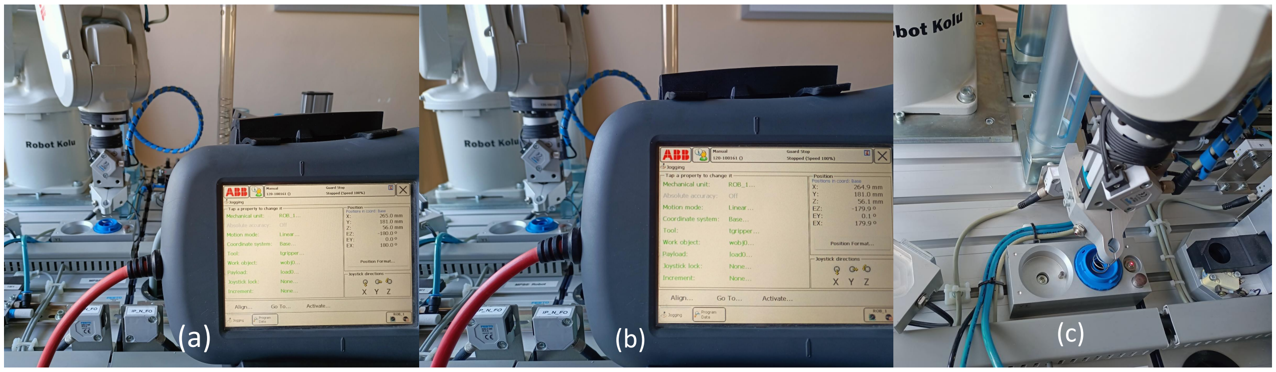


Figure 8. Assembly application with Boomerang Algorithm and positioning errors (a) desired position-orientation values, (b) Boomerang algorithm solution, (c) assembled spring position with calculated joint values.

As a result of the tests carried out, it has been seen that the joint angles calculated with the Boomerang algorithm provide the required positioning accuracy for the properly performed precise assembly applications. Also, with the demonstrated assembly application, it has been shown that the Boomerang algorithm is suitable for use in assembly applications where positioning accuracy is important.

5. Discussion

In this study, we have proposed a PSO-variant algorithm called Boomerang for solving inverse kinematics of industrial robots. Boomerang algorithm has recursive structure and uses an alternative method to DH parameters, named OPW, to define robot kinematic structure for reducing computational load. It has been proven that the OPW method used in the developed algorithm reduces the computational load compared to the DH parameters. One hundred random samples taken from the workspace were tested using the Boomerang algorithm with the same parameters by changing the DH and OPW based objective functions. It has been shown that a significant advantage is provided in the computation times when the objective function defined by OPW is used. We have compared Boomerang algorithm with formerly proposed PSO algorithms over two different industrial robots with same dataset and hardware in simulation studies. The success rate of PSO1 algorithm is 100%, but its computation time, and position and orientation errors are relatively high. PSO2 has good results in position and orientation error but has moderate computation time and the least success rate. PSO3 has the least position errors with 85% success rate, but computation time (13.7625 s) is the worst amongst. Boomerang algorithm obtained the best orientation with a MAE of 0.0761° and fairly good position with a MAE of 0.0408 mm. The success rate of the proposed algorithm is 95%. Moreover, proposed algorithm reaches to a precise inverse kinematics solution in 0.7825 s. From the simulation and nonparametric statistical tests, we found that the proposed Boomerang algorithm is more effective compared to the other PSO variants in terms of computational time, position and orientation errors, and its convergence to a solution. For the feasibility assessment of the PSO variant algorithms a "feasibility" function is defined which correlates computational time, position and orientation errors and success rate. PSO1 algorithm has a feasibility of 60% where PSO2 and PSO3 algorithms' feasibility measures are 73% and 76%, respectively. The feasibility of Boomerang algorithm is found to be more superior by 8% compared to the other PSO variants. The accuracy of the simulation results has been tested by using the Boomerang algorithm in a physical application on ABB IRB120 robot arm. As a result of the tests performed, we have observed that the Boomerang algorithm can reach an inverse kinematic solution with a computational time of less than a second with a mean average position error of 0.04 mm and orientation error of $0.07°$. Experimental results were also supported by the successful completion of the assembly application, which was performed using the Boomerang algorithm without exceeding needed error tolerance of 0.225 mm for the selected application. The angles calculated by the Boomerang algorithm resulted in a positioning error of 0.1 mm, 0 mm, and 0.1 mm for respectively x, y, and z coordinates and an orientation error of 0.1, 0, and 0.1 degrees for roll, pitch, and yaw angles. The biggest disadvantage of the algorithm is that although it produces a solution with a high probability (95%), it does not guarantee to find a solution as in other PSO variants. Moreover, since the algorithm generates a solution for a pose, possible poses can be dangerous in dynamic environments and for obstacle avoidance.

6. Conclusion

From the simulation results, statistical nonparametric tests, and physical applications, we observed that the proposed method succeeded in solving the inverse kinematics of industrial robots. Boomerang algorithm can bring the end effector to the desired position and orientation. With the proposed algorithm, the inverse kinematic analysis of randomly selected positions in the workspace can be performed as statistically shown probability of 95%, but as with all other optimization-based methods, the solution is not guaranteed to be found and it may not appropriate to achieve the desired pose when the dynamic obstacles is case, because the proposed algorithm seek for a solution to a single pose with desired position-orientation pair not path. It is also shown that the OPW method, which is the DH alternative we propose for use in optimization-based inverse kinematics algorithms, and the forward kinematics equations used to obtain the solution reduce the computation time and can be an alternative and useful to implement in applications. Furthermore, according to the nonparametric tests and feasibility comparison results in terms of computational time, position and orientation errors and success rate of the PSO variant algorithms, the proposed algorithm obtained the best performance. Wilcoxon nonparametric statistical analysis based on MAEs of position and orientation show that Boomerang algorithm demonstrates a significant improvement against other PSO variant algorithms. The mean absolute errors of the inverse kinematic solutions obtained by the proposed method are 0.0408 mm for positioning and $0.0761°$ for orientation. Also, the physical tests and assembly application on an industrial robot arm verified that the proposed algorithm successfully solves the inverse kinematics in less than a second. Consequently, the simulation and experimental results showed that the proposed algorithm is an alternative way to define kinematic structure and has ability to solve inverse kinematic of 6 DOF serial manipulators.

References

[1] Siciliano B, Sciavicco L, Villani, Oriolo G. Robotics: Modelling, Planning and Control. London: Springer London, 2009.

[2] Kucuk S, Bingul Z. Robot Kinematics: Forward and Inverse Kinematics in Industrial Robotics: Theory, Modelling and Control. Rijeka, Croatia: Intech, 2006.

[3] Uzuner S, Akkuş N, Toz M. 5 Eksenli Manipülatörün Eklem Uzayında Yörünge Planlaması. J Polytech. 2017; 20 (1):151-157.

[4] Uzuner S, Akkus N, Toz M. 5-DOF serial robot manipulator design, application and inverse kinematic solution through analytical method and simple search technique. Pamukkale Univ. J. Eng. Sci. 2020;26 (2):392-401. doi: 10.5505/pajes.2019.95881

[5] Aristidou A, Lasenby J. Inverse Kinematics: a review of existing techniques and introduction of a new fast iterative solver. University of Cambridge, Technical Report 2009. doi: 10.1016/S0140-6736(01)72055-7

[6] Olsen AL, Petersen HG. Inverse kinematics by numerical and analytical cyclic coordinate descent. Robotica 2011; 29 (4): 619-626. doi: 10.1017/S026357471000038X

[7] Kennedy J, Eberhart R. Particle swarm optimization. In: Proceedings of ICNN'95 International Conference on Neural Networks; Perth, WA, Australia; 1995; 4: 1942-1948; doi: 10.1109/ICNN.1995.488968

[8] Dereli S, Köker R. A meta-heuristic proposal for inverse kinematics solution of 7-DOF serial robotic manipulator: quantum behaved particle swarm algorithm. Artif. Intell. Rev. 2020; 53 (2): 949–964. doi: 10.1007/s10462-019-09683-x

[9] Çavdar T, Mohammad M, Milani RA. A new heuristic approach for inverse kinematics of robot arms. Adv. Sci. Lett. 2013; 19 (1): 329-333. doi: 10.1166/asl.2013.4700

[10] Zhao Y, Huang T, Yang Z. A new numerical algorithm for the inverse position analysis of all serial manipulators. Robotica 2006; 24 (3): 373–376. doi: 10.1017/S0263574705002298

[11] Durmuş B, Temurtaş H, Gün A. An inverse kinematics solution using particle swarm optimization. In: International Advanced Technologies Symposium 2011; Elazığ, Turkey; 4: pp. 193–197.

[12] Lopez-Franco C, Hernandez-Barragan J, Alanis AY, Arana-Daniel N. A soft computing approach for inverse kinematics of robot manipulators. Eng. Appl. Artif. Intell. 2018; 74: 104–120. doi: 10.1016/j.engappai.2018.06.001

[13] An J, Li X, Zhang Z, Man W, Zhang G et al. Application of An Improved Particle Swarm Optimization Algorithm in Inverse Kinematics Solutions of Manipulators. In: 2020 IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC); Chongqing, China; 2021. 9: pp. 1680–1684.

[14] Du Y, Wu Y. Application of IPSO algorithm to inverse kinematics solution of reconfigurable modular robots. In: Proceedings 2011 International Conference on Mechatronic Science, Electric Engineering and Computer (MEC 2011); Jilin, China; IEEE, 2011. pp. 1313-1316.

[15] Toz M. Chaos-based Vortex Search algorithm for solving inverse kinematics problem of serial robot manipulators with offset wrist. Appl. Soft Comput. 2020; 89, 106074. doi: 10.1016/j.asoc.2020.106074

[16] de LS Junior J, de O. Jesus RC, Molina L, Carvalho EAN, Freire EO. FRPSO: Inverse Kinematics Using Fully Resampled Particle Swarm Optimization. In: 2018 Latin American Robotic Symposium, 2018 Brazilian Symposium on Robotics (SBR) and 2018 Workshop on Robotics in Education (WRE); Joao Pessoa, Brazil; 2018. pp. 402–407.

[17] Toz M. Inverse Kinematic Solution of a 6-DOF Serial Robot Manipulator with Offset Wrist by using ALO Algorithm. Sigma J. Eng. & Nat. Sci. 2017; 8 (2): 81–90.

[18] Falconi R, Grandi R, Melchiorri C. Inverse kinematics of serial manipulators in cluttered environments using a new paradigm of particle swarm optimization. In: IFAC Proceedings Volumes (IFAC-PapersOnline); Cape Town, South Africa 2014; 47 (3): 8475-8480.

[19] Rokbani N, Alimi AM. Inverse kinematics using particle swarm optimization, a statistical analysis. In: Procedia Eng. 2013; 64: 1602-1611. doi: 10.1016/j.proeng.2013.09.242

[20] Ram RV, Pathak PM, Junco SJ. Inverse kinematics of mobile manipulator using bidirectional particle swarm optimization by manipulator decoupling. Mech. Mach. Theory 2019; 131: 385–405. doi: 10.1016/j.mechmachtheory.2018.09.022

[21] Dereli S, Köker, Öylek Rİ, Ay M. A Comprehensive Research on The Use Of Swarm Algorithms in The Inverse Kinematics Solution. J. Polytech. 2018; 22 (1): 75-79. doi: 10.2339/politeknik.374830

[22] Nguyen MT, Yuan C, Huang JH. Kinematic Analysis of A 6-DOF Robotic Arm. In: IFToMM World Congress on Mechanism and Machine Science; Krakow, Poland; 2019. pp. 2965–2974.

[23] Huang HC, Chen CP, Wang PR. Particle swarm optimization for solving the inverse kinematics of 7-DOF robotic manipulators. In: Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics (SMC); Seoul, Korea (South); IEEE, 2012. pp. 3105-3110. doi: 10.1109/ICSMC.2012.6378268

[24] Alkayyali M, Tutunji TA. PSO-based algorithm for inverse kinematics solution of robotic arm manipulators. In: Proceedings of the 2019 20th International Conference on Research and Education in Mechatronics (REM); Wels, Austria; IEEE, 2019. pp. 1-6. doi: 10.1109/REM.2019.8744103

[25] Dereli S, Köker R. IW-PSO approach to the inverse kinematics problem solution of a 7-DOF serial robot manipulator. Sigma J. Eng. Nat. Sci. 2018; 36 (1): 77–85.

[26] Jha P, Biswal BB. Optimization Approach for Inverse Kinematic Solution. In: Hurtado, EG (editor). Kinematics, London, UNITED KINGDOM: IntechOpen, 2017. doi: 10.5772/intechopen.71409

[27] El-Sherbiny A, Elhosseini MA, Haikal AY. A new ABC variant for solving inverse kinematics problem in 5 DOF robot arm. Appl. Soft Comput. 2018; 73: 24-38. doi: 10.1016/j.asoc.2018.08.028

[28] Brandstötter M, Angerer A, Hofbaur M. An Analytical Solution of the Inverse Kinematics Problem of Industrial Serial Manipulators with an Ortho-parallel Basis and a Spherical Wrist. In: Proceedings of the Austrian Robotics Workshop; Linz, Austria 2014; 22 (8): 7-11.

[29] Küçük S, Bingül Z. The inverse kinematics solutions of industrial robot manipulators. In: Proceedings of the IEEE International Conference on Mechatronics (ICM'04); Istanbul, Turkey; IEEE, 2004. pp. 274-279. doi: 10.1109/icmech.2004.1364451

[30] García-Nieto J, Alba E. Restart particle swarm optimization with velocity modulation: A scalability test. Soft Comput. 2011; 15 (11): 2221-2232. doi: 10.1007/s00500-010-0648-1

[31] Rokbani N, Casals A, Alimi AM. IK-FA, a new heuristic inverse kinematics solver using firefly algorithm in Computational intelligence applications in modeling and control. Springer, Cham, 2015; pp. 369-395.

[32] Rokbani N, Slim M, Alimi AM. The Beta distributed PSO, β-PSO, with application to Inverse Kinematics. In: IEEE 2021 National Computing Colleges Conference (NCCC); Taif, Saudi Arabia; 2021. pp. 1-6, doi: 10.1109/NCCC49330.2021.9428811

[33] Rokbani N, Neji B, Slim M, Mirjalili S, Ghandour R. A Multi-Objective Modified PSO for Inverse Kinematics of a 5-DOF Robotic Arm. Appl. Sci.-Basel 2022; 12 (14):7091.