Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Real-Time Inverse Kinematics Using Dual Particle Swarm Optimization DPSO of 6-DOF Robot for Nuclear Plant Dismantling

Hamza Khan[1], Hyun Hee Kim[2], Saad Jamshed Abbasi[1] and Min Cheol Lee*

[1]School of Mechanical Engineering, Pusan National University
Busan, Korea (Tel: +8210-9535-0496; e-mail: hamzakhan.0496@gmail.com).
(e-mail: saadjamshed93@gmail.com)
[2]Division of Robotics Convergence, Pusan National University
Busan, Korea (email: sleepingjongmo@nate.com)
* School of Mechanical Engineering, Pusan National University,
Busan, Korea (e-mail: mclee@pusan.ac.kr), corresponding author

Abstract: Robotic manipulator inverse kinematics (IK) solution has a significant role in robotics. In this paper, a new paradigm of particle swarm optimization (PSO) called dual particle swarm optimization (DPSO) for the inverse kinematics problem of a robotic manipulator used in the nuclear plant decommissioning process is proposed. The introduced approach of particle swarm optimization relies on the approach of dividing one particle swarm optimization algorithm into two such that each algorithm optimizes separately the problem for position and orientation, in this way the system will achieve faster convergence toward desired results with fewer iterations. The proposed algorithm is designed and implemented in MATLAB/Simulink to solve the inverse kinematics problem for a 6 degree of freedom selective compliance articulated robot arm (SCARA) type robot with joint space constraints. For a real-time experiment, a pair of the joysticks was integrated with MATLAB/Simulink. Using a dual joystick, the position and orientation for the robot were set. Furthermore, the experimental results demonstrate the effectiveness of DPSO on real-time within given constraints.

***Keywords**:* Particle Swarm Optimization-PSO, Inverse Kinematics, Robotic Manipulator, Joystick.

1. INTRODUCTION

Robots capability for the autonomous operation and their competence to perform a large number of set of tasks have captured the attention of researchers and industries (Graetz and Michaels, 2018). The scope of industrial robot applications was established from the conventional handling, assembly and welding tasks leading to a wide range of production (Abele et al., 2007). Robots and their precise operation are getting popular. Usage of a robot has been increased in a large number of applications such as handling tasks that might be tough or risky for human beings (Hao et al., 2011). In industrial applications of a robot, the main objective of replacing the human with robots is to provide efficiency and accuracy. For nuclear applications, the thermal and radioactive environment of nuclear plant requires human-less automation (Moore, 1985). Attention on nuclear power plants dismantlement after the incident of the Fukushima nuclear power plant in Japan has been gained all over the world (Abbasi et al., 2019).

Humans provide guidance to the robot to achieve the desired task which includes the robot end-effector position and orientation (Mehmood et al., 2014). Inverse kinematics (IK) of the robotic manipulator help for the robot to find the set of joint angles that aligns the robotic manipulator's posture (position and orientation) with the desired posture. The guidance starts with planning a trajectory for a robotic manipulator then followed by the joint motion controller and thereafter the robot control to generate enough joint torque to accurately track the trajectory (Siciliano et al., 2010).

For each given task, it is not possible to directly control the manipulator as there is no guidance or path to be tracked for the desired task, therefore, a trajectory is planned for a manipulator's end-effector. The planned trajectory is then divided into several points so that for each point the inverse kinematics is performed to get the desired joint trajectory for each link which becomes the desired set point for each link joint.

IK solution has a significant role in robotics. Usually, the end-effector position is considered for a solution as introducing orientation will create a lot of problems. A task is assigned to the robot, which is in Cartesian space, where the control is in joint space. Different techniques have been used for IK of a robotic manipulator. (Siciliano, 1990) proposed Closed Loop Inverse Kinematics (CLIK) algorithm, which performs IK in an iterative way for the desired pose (position and orientation). The proposed algorithm is based pseudo-inverse of a Jacobian matrix *J*. This approach was analysed on a 6-DOF SCARA robot, but the results were not satisfactory because of high computational complexity (Siciliano, 1990).

To overcome computational complexity, researchers have proposed many different approaches as an alternative. One of them is a heuristic optimization technique emerging widely in the field. Heuristic techniques include Genetic Algorithm (GA) (Zhang and Nelson, 2011), Ant Bee Colony (ABC)(Çavdar et al., 2013) or Particle Swarm Optimization (PSO) (Rokbani and Alimi, 2013). In these techniques, the solution is iteratively improved by adopting a set of operations which mimics a

natural process like birds flocking. For non-linear constrained problem the most promising solution is particle swarm optimization (PSO). Despite the different proposed techniques for IK based on simplicity and accuracy, constraints are barely considered.

This research proposes a new paradigm of PSO algorithm called Dual Particle Swarm Optimization (DPSO) algorithm to solve IK of 6-DOF SCARA robot in the presence of joint and workspace constraints using a joystick. When using the DPSO, the cumulative error of end-effector position and orientation is very small as compared to classical GA, PSO and inverse Jacobian method.

The paper follows the following pattern: Section 2 presents the classical PSO algorithm, inverse kinematics approach using PSO and related work. Section 3 presents the problem statement of the situation and the proposed algorithm for IK. Section 4 presents the system model and kinematics including system space limitations. Section 5 discusses the experimental setup and simulation results used to validate the proposed algorithm and Section 6 entails the concluding remarks of the paper.

2. BACKGROUND

2.1 Particle Swarm optimization

PSO is an optimizing technique which tries to improve its population solution iteratively. PSO is developed by (Eberhart and Kennedy, 1995). In the original formulation, the authors presented an optimization technique that contains several particles called swarm trying to optimize the problem by moving in problems search space. The measure of quality is a *fitness function (objective function) f(.)* which depends upon different parameters. The algorithm consist of several particles and each particle position is considered as a possible solution to that problem. Iteratively the particle changes its position with a velocity. After *k*th iteration, the position and velocity of the independent particle of the generated population $\mathcal{P}$ at *k*th iteration is defined by M-dimensional vectors as follows

$$x^i(k) = \left[x_1^i(k) \ldots x_M^i(k)\right]^T \quad (1)$$

$$v^i(k) = \left[v_1^i(k) \ldots v_M^i(k)\right]^T \quad (2)$$

where $i = 1 \ldots \|\mathcal{P}\|$, x and v represent the position and velocity of the *i*-th member of M-dimensional vector respectively. In PSO, each particle also knows the information about its neighbour, by exchanging their information so that they know about the globally best particle in that swarm and the swarm must follow the best particle. The communication of each particle with their neighbour is defined as

$$N_i(k) = \{j \in \mathcal{P} : i \leftrightarrow j\} \quad (3)$$

where the symbol ↔ represents the exchange of information between particles. The movement and the direction of particles within the search space in PSO depends upon the personal best solution $X_{pB}^i(k)$ and the global best $X_{gB}(k)$ (Poli et al., 2007). When the neighbours communicate, they share the information of best-known particle which is globally best in the swarm and according to this information, the new globally best particle is updated by satisfying the following condition.

$$X_{gB}(k) = \left\{X_{pB}^i(k) : f\left(X_{pB}^i(k)\right) < X_{gB}(k)\right\} \quad (4)$$

where $X_{gB}(k)$, $X_{pB}^i(k)$ and $f\left(X_{pB}^i(k)\right)$ are the global best at *k*-th iteration, personal best of *i*-th particle at *k*-th iteration and objective function to compute personal best of *i*-th particle at *k*-th iteration respectively. Each particle at every single iteration updates their position and compare their personal best with the global best. If the personal best of a particle at *k*th iteration in minimization problem is less than the global best as described in (4), the global best is updated with that particle personal best. Where the personal best of each particle is updated according to (5) i.e. if the personal best of *i*-th particle at *k*-th iteration is less than the last known personal best of the same particle

$$X_{pB}^i(k) = \left\{X_{pB}^i(k) : f(obj) < X_{pB}^i\right\} \quad (5)$$

As PSO algorithm starts, the initial population with n-number particles are generated within the predefined range with zero velocity. At each iteration *i*-th, particle velocity and position are updated by (6) and (7).

$$v^i(k+1) = \omega . v^i(k) + c_1 . rand . \left(X_{pB}^i(k) - x^i(k)\right) + c_2 . rand . \left(X_{gB}(k) - x^i(k)\right) \quad (6)$$

$$x^i(k+1) = x^i(k) + v^i(k+1) \quad (7)$$

where ω represents the inertia coefficient, c_1 and c_2 are the learning factor or acceleration factor and $rand$ represents the generation of random number within the specified range $[-v_{max}, v_{max}]$. As the particles get closer to the desired result, the velocity is optimized in such way that the process slows down so that the solution does not deviate from the desired result or have a large or increasing error.

2.2 Related Work for Inverse Kinematics

Many researches have performed IK using PSO and evaluated their results for different types of robotic manipulators. (Rokbani and Alimi, 2013) Analysed the PSO statistically for the robotics IK and concluded that PSO can easily take care of IK of robotic manipulator without computing the inverse model. (Junior et al., 2018) used PSO to solve the IK problem for a 4 DOF robotic manipulator based on full resampling of the particles. (Collinsm and Shen, 2017) used PSO for a high degree of freedom IK which yields the desired position and orientation with a very low error. (Adly and Abd-El-Hafiz, 2016) proposed single and multiple objective PSO for 5-DOF and 7-DOF robotic manipulator end-effector position. While (Falconi et al., 2014) presented the idea of IK using PSO for the cluttered environment with obstacles. This approach divides the particles in a subgroup with specific tasks with faster convergence for a 3-DOF planner robot. (Huang et al., 2012) presented the IK solution based on PSO algorithm applicable to 7-DOF robot manipulator and validated the results in simulation. An improved PSO (IPSO) is presented by

(Du and Wu, 2011) for the solution of IK and simulated the algorithm and (Collinsm and Shen, 2017) added a self-collision avoidance to the IK using PSO. When compared with DPSO, DPSO possesses small error with less calculation time even on the real-time experiment.

3. DPSO INVERSE KINEMATICS

3.1 Problem Statement

Initially, to solve the real-time IK of 6-DOF robotic manipulator used for nuclear power plant dismantling, the inverse Jacobian iterative method was used. The iterative form of the Jacobian matrix can be written as (8)

$$\theta_{i+1} = \theta_i + d\theta = \theta_i + J^{-1}dP \tag{8}$$

where i is the number of iterations to be performed, $P = \begin{bmatrix} P_x & P_y & P_z & \emptyset & \theta & \varphi \end{bmatrix}^T$ is the posture matrix contains three position and three orientation coordinates and J^{-1} is the inverse of the Jacobian matrix. Where the inverse kinematics is represented as

$$[D_\theta] = [J^{-1}][D] \tag{9}$$

$$d\theta_{5\times1} = J^{-1}_{5\times6} dP_{6\times1} \tag{10}$$

Where the matrix of Jacobian (Park and Lee, 2018) is given as

$$J = \begin{bmatrix} z_0 \times (o_5 - o_0) & z_1 \times (o_5 - o_1) & \cdots & z_4 \times (o_5 - o_4) \\ z_0 & z_1 & \cdots & z_4 \end{bmatrix} \tag{11}$$

$$z_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, z_1 = R_0^1 z_0, \cdots, z_4 = R_3^4 z_3 = R_0^1 \cdots R_3^4 z_0 \tag{12}$$

where o_i and R_i^j are the distance between origin and *i*th-axis coordinate and orientation matrix from *i*th-axis to *j*th-axis respectively. The total 2000 iterations were performed for better results of IK when performing only for end-effector position. But when the orientation of end-effector was introduced in the solution, the results have a very large error ($0^\circ - 120^\circ$) in orientation because of the kinematics structure of the robotic manipulator in the current study. Therefore, the heuristic approach was used to determine accurate IK. DPSO analysis includes the calculation of end-effector position (in Cartesian space) and orientation (roll-pitch-yaw) simultaneously but with very low calculation time of $t_{allow} = .25sec$ with a maximum allowable tolerance of position ($E_p = \pm 5mm$) and orientation ($E_o = \pm .008$ rad) of the end-effector. In real-time the sampling time of a system is very small which makes the calculation time much faster, while moving the robot desired trajectory is divided into several points and for each point, the calculation should be very fast. No work has been identified which has proven the effectiveness of PSO to perform IK on real-time with given requirements. Therefore, the proposed new technique of PSO named DPSO has been proposed to solve the IK problem by using two PSO algorithms and then integrating them to get better results.

3.2 Proposed Algorithm

PSO iteratively improves the solution, the PSO algorithm updates each particle constantly based on the best solution so far. Each particle move with some velocity towards the desired point and the velocity reduces as the particle gets closer to the best-known particle in the population.

The proposed scheme utilizes two different PSO algorithms working separately, each for analyzing position and orientation. The PSO-1 tries to reach to the end effector desired position and then PSO-2 will slightly adjust the joint angles to obtain the desired orientation. Following are the steps for the proposed algorithm.

Step 1: Initial population of N particles is generated randomly within the predefined range. Where N is the number of particles generated depend upon the robot degree of freedoms and is defined as

$$\mathrm{N} = \frac{n_{dof}\times 10}{t_{allow}\times E_p} \tag{13}$$

where n_{dof} represents the robot's number of degrees of freedom.

Step 2: Global best is set to infinity. Forward kinematics is computed for each particle. The function of the forward kinematics of a robot whose joint configuration is defined by $q_i(k)$ is given by

$$f_{kine}\big(q_i(k)\big) = {}^0T_M\big(q_i(k)\big) = \begin{bmatrix} {}^0R_M^i(k) & {}^0p_M^i(k) \\ [0 \quad 0 \quad 0] & 1 \end{bmatrix} \tag{14}$$

where ${}^0R_M^i(k) \in \mathbb{R}^{3\times3}$ and ${}^0p_M^i(k) \in \mathbb{R}^{3\times1}$ represents the end-effector orientation and position respectively.

Step 3: Compute the objective function for each particle.

$$f_{obj1} = \|P_d - P_c\| = \sqrt{(x_d - x_c)^2 + (y_d - y_c)^2 + (z_d - z_c)^2} \tag{15}$$

where $P_d\ (x_d, y_d, z_d)$ and $P_c(x_c, y_c, z_c)$ are the desired and current position in Cartesian space respectively.

Step 4: Particle personal best is computed according to (5), and then global best is selected according to (4) i.e. if the objective function of an *i*-th particle is less than the personal best of the same particle at *k*-th iteration the personal best is updated of that particle and if the personal best of that particle is less than the global best then the global best is updated with the personal best of that particle. Update the position for next particle using (6) and (7).

Step 5: Compute the personal best and the global best at each iteration and check for the stopping criteria. If the algorithm has reached the desired joint variable for the required position.

Step 6: If the required position is obtained within $E_p = \pm 5mm$, stop the further iterations for the position.

Step 7: Generate M particles around the global best which is obtained in step 6. Where M is defined as

$$\mathrm{M} = \frac{it_{pso1}\times t_{pso1}}{E_o \times d} \tag{16}$$

where it_{pso1}, t_{pso1} and d are the total iteration performed in PSO-1 to obtain the desired position, time taken by PSO-1 and any constant that makes the M < N respectively. In this case d = 10.

Step 8: Newly generated particle will start adjusting the joint variables in such a way that the end effector position remains the same nut the desired orientation is obtained by repeating the step 2, step 3, step 4 and step 5.

The velocity of PSO-2 is reduced, as the newly generated particles will be moving in the search space near the desired position. If the acceleration factor is high, the output error will occur because the particle will search in the space out of range. To solve this problem, therefore, the new velocity used for PSO-2 is defined as

$$v^i(k+1) = \omega.v^i(k) + c_2.rand.\left(X_{gB}(k) - x^i(k)\right) \quad (17)$$

The objective function of PSO-2 is

$$f_{obj2} = \rho\|P_d - P_c\| + \sigma\|O_d - O_c\| = \rho f_{obj1} + \sigma\sqrt{(R_d - R_c)^2 + (P_d - P_c)^2 + (Y_d - Y_c)^2} \quad (18)$$

where O_d (R_d, P_d, Y_d) and O_c (R_c, P_c, Y_c) are the desired orientation and current orientation respectively. R, P *and* Y represent the roll, pitch and yaw respectively. Also, $0 \leq \rho \leq 1$ and $0 \leq \sigma \leq 1$ are the weighing importance of position and orientation. As PSO-2 will search for desired orientation without affecting the position, therefore, in proposed algorithm ρ and σ are set as $\rho = .25$ and $\sigma = .75$ respectively.

Step 9: Stop the iterations if the desired orientation is obtained within $E_o = \pm.008\text{rad}$.

Step 10: The algorithm will check whether the results are within the space limitation of the robot. If not, then the algorithm will adjust them within the specified range.

4. SYSTEM DESIGN AND KINEMATIC MODEL

The robotic manipulator is a 6-DOF SCARA robot. The first link is composed of telescopic structure which in actual is the prismatic joint. Rest is a 5-DOF robotic arm being used in the nuclear-decommissioning operation. The CAD model isometric view of SCARA is presented in Fig 1.

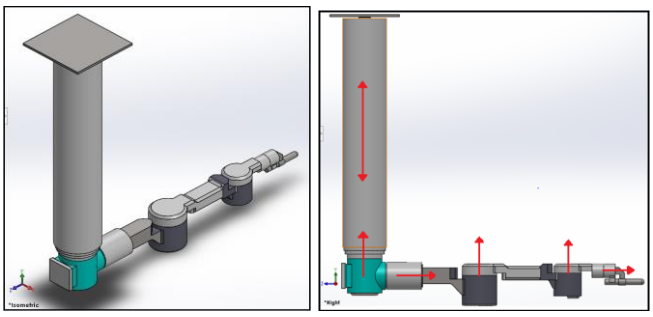

Fig. 1. 6-DOF SCARA type robot CAD model and axis configuration

SolidWorks is used to get the virtual prototype model of the actual system as shown with the axis configuration in Fig .1. The red lines indicate the axis of rotation and the prismatic joint of the 6-DOF SCARA robot. For kinematics, the Denevit-Hartenberg (DH) parameters of the robot are presented in Table 1. Robot joint space and prismatic joint limitations are shown in Table 2.

Table 1. DH Parameters of Scara Robot

Joint	θ (rad)	d (m)	a (m)	α (rad)
1	0	d_1*	0	0
2	$\theta_2{}^*$	0	0	$-\frac{\pi}{2}$
3	$\theta_3{}^*$	0.750	0	$\frac{\pi}{2}$
4	$\theta_4{}^*$	0	0.820	0
5	$\theta_5{}^*$	0	0	$\frac{\pi}{2}$
6	$\theta_6{}^*$	0.730	0	0

Table 2. Joint space limitation

$d_1{}^*$	$\theta_2{}^*$	$\theta_3{}^*$	$\theta_4{}^*$	$\theta_5{}^*$	$\theta_6{}^*$
2.5~10	-2π~2π	-2π~2π	$\frac{-3\pi}{4}$~$\frac{3\pi}{4}$	$\frac{-3\pi}{4}$~$\frac{3\pi}{4}$	-2π~2π

5. EXPERIMENTAL SETUP AND RESULTS

MATLAB/Simulink environment is used to program and then simulate the proposed algorithm. As the robot is used for the nuclear decommissioning process, it is remotely controlled using a joystick and virtual reality (VR). Because of the dangerous environment, the operator works in off-site in an operating room while the robot works in on-site to perform the real task. Through joystick, the required position and orientation is set and controlled.

The Robot end effector should follow the joystick command in such a way that joystick gives the required position and orientation and then PSO solves the IK for required data. The joint angles are generated which are then used for the forward kinematics of robot and movement. Initially, the proposed algorithm is executed on MATLAB for the validation of the performance and results, then for real-time simulation joystick is used.

5.1 Test Run Simulations

Initially, the simulations were performed without Joystick to evaluate the difference between classical PSO and the proposed DPSO algorithm, and thereafter the results of the proposed algorithm will be ensured. Randomly generated few desired pose (position and orientation) are shown in Table 3. These poses were tested for the desired results. The PSO parameters for the test run are shown in Table 4. As expected, the algorithm proved its efficiency over the test run whose results are presented in Table 5.

Where ω_d (in Table 4.), it_{PSO1}, it_{PSO2} and T_R in Table 5 are the damping ratio of inertia coefficient, iterations performed by

PSO-1, iterations performed by PSO-2 and run time respectively. Initially, the algorithm was not converging toward the desired result, so the inertia of the algorithm was adjusted at each iteration which is given as

$$\omega = \omega_d \times \omega \tag{19}$$

Table 3. Desired position and orientation

No	$P_d\ (x_d, y_d, z_d)$	$O_d(R_d, P_d, Y_d)$
1	$-1.99, .642, -3.14$	$-2.35, -.82, -1.74$
2	$-.6, .425, -2.84$	$-1.43, .87, -3.03$
3	$-.002, .82, -10$	$1.57, 1.57, 0$
4	$1.62, 1.62, -5$	$-1.57, .78, 0.78$

Table 4. PSO Parameters

	PSO-1	PSO-2
ω	0.5	0.5
ω_d	0.99	0.9
c_1	1.5	0
c_2	1.5	1

Table 5. DPSO Test run results

No	E_p	E_o	it_{PSO1}	it_{PSO2}	T_R
1	0.000234	0.0033	29	8	0.21
2	0.000202	0.0056	21	12	0.23
3	0.000381	0.0049	23	14	0.2
4	0.000493	0.0009	29	9	0.24

The convergence graphs of classical PSO and the proposed algorithm are shown in Figs. 2 and 3, respectively. Fig. 2 shows the results that the classical PSO performed the task with an average of 75 iterations and slow convergence which is time taking whereas Fig. 3 shows the proposed algorithm can converge toward the desired point with an average of 32 iterations in total.

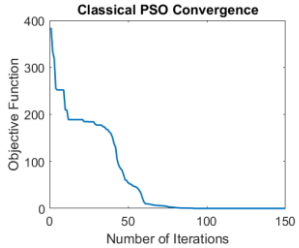

Fig. 2. Classical PSO Objection function evaluation

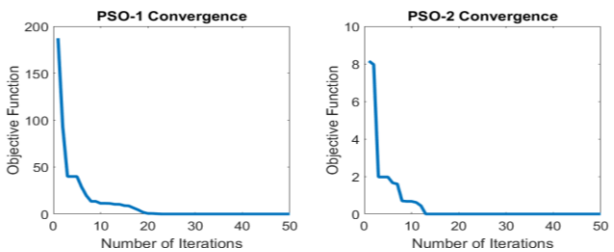

Fig. 3. DPSO objective function evaluation

In simulations, the average time of simulation when using the inverse Jacobian method was 0.15 - 0.2sec with 1000 iterations, whereas, for DPSO, the average time of simulation was 0.2sec because the number of populations according to (13) are 48, therefore, the total iterations performed was $48 * 20 + 34 * 12 = 1368$. Although the inverse Jacobian method took less time than DPSO, the accuracy if inverse Jacobian was not so good (55%), while the accuracy of DPSO was up to 99%.

5.2 Real Experiment with Joystick

As the proposed algorithm is for the real-time calculation, therefore, the test run results cannot validate the proposed algorithm. The experimental setup includes joystick which is connected to MATLAB/Simulink and will provide the end-effector desired pose. The experimental setup is shown in Fig. 4. Two joysticks are being used in this experiment. The left and right joystick set the desired point and orientation respectively. The values from joystick are then scaled within the robot workspace limits. IK using DPSO is performed and thereafter generated joint angles are then ported to the virtual physical system model in the Simulink. And the error of orientation and position is plotted.

Fig. 5. shows the end-effector position error during real-time when controlling using a joystick.

Fig. 4. Experimental setup

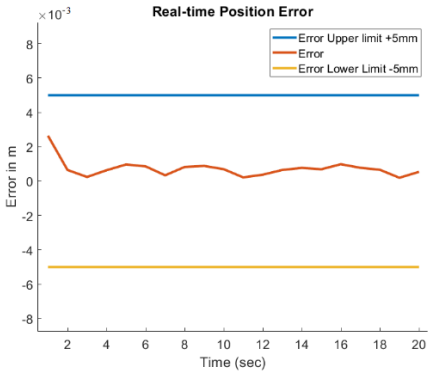

Fig. 5. Position error in real-time

Similarly, the real-time orientation error is shown in Fig. 6. While experimenting on real-time, a random shape was generated as shown in Fig. 7 which shows top view to explain the DPSO effectiveness over real-time with the help of red and blue plotted points presenting the desired and current position of the robot respectively. It was confirmed that the result of robot trajectory follows well the joystick command.

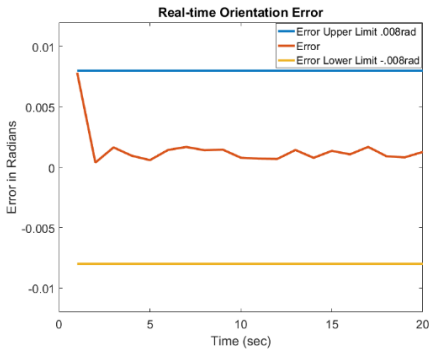

Fig. 6. Orientation error in real-time

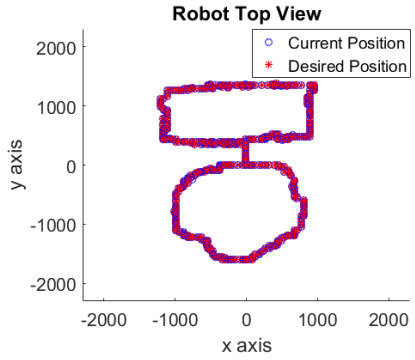

Fig. 7. Generated shape using the joystick

6. CONCLUSION

Inverse kinematics have a great role in robotics before controlling the robotic manipulator. Usually, end-effector position is considered but the solution gets messy when the orientation is considered simultaneously because the orientation is also an important part for the end-effector required pose. Therefore, in this paper, we proposed a novel IK approach for inverse kinematics of 6-DOF robot using dual particle swarm optimization (DPSO) algorithms each for position and orientation respectively. Initially, the classical PSO algorithm was used for position and orientation but the results were not satisfactory with greater position and orientation error and slow convergence. However, after analyzing the problem under the given conditions, the defined objective functions are capable enough to consider not only the end-effector positioning problem but also the final orientation. The algorithm first evaluates the desired position using PSO-1 and then using PSO-2, the algorithm generates the desired orientation. The efficiency of DPSO for IK problem has been proven through simulations and real-time tele-operation implementation. The results demonstrated the capability of the proposed approach that it has a big impact on the number of iterations needed to complete the objective.

ACKNOWLEDGEMENT

This research was supported by the nuclear research and development program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (MSIT, Korea). [NRF-2019M2C9A1057807]
This research was funded by the Technology Innovation Program (10073147, Development of Robot Manipulation Technology by Using Artificial Intelligence) funded By the Ministry of Trade, Industry & Energy(MOTIE, Korea)

REFERENCES

ABBASI, S. J., KALLU, K. D. & LEE, M. C. 2019. Efficient Control of a Non-Linear System Using a Modified Sliding Mode Control. *Applied Sciences,* 9**,** 1284.

ABELE, E., WEIGOLD, M. & ROTHENBÜCHER, S. 2007. Modeling and identification of an industrial robot for machining applications. *CIRP annals,* 56**,** 387-390.

ADLY, M. & ABD-EL-HAFIZ, S. Inverse kinematics using single-and multi-objective particle swarm optimization. 2016 28th International Conference on Microelectronics (ICM), 2016. IEEE, 269-272.

ÇAVDAR, T., MOHAMMAD, M. & MILANI, R. A. 2013. A new heuristic approach for inverse kinematics of robot arms. *Advanced Science Letters,* 19**,** 329-333.

COLLINSM, T. J. & SHEN, W.-M. Particle swarm optimization for high-dof inverse kinematics. 2017 3rd International Conference on Control, Automation and Robotics (ICCAR), 2017. IEEE, 1-6.

DU, Y. & WU, Y. Application of IPSO algorithm to inverse kinematics solution of reconfigurable modular robots. 2011 International Conference on Mechatronic Science, Electric Engineering and Computer (MEC), 2011. IEEE, 1313-1316.

EBERHART, R. & KENNEDY, J. A new optimizer using particle swarm theory. MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science, 1995. Ieee, 39-43.

FALCONI, R., GRANDI, R. & MELCHIORRI, C. 2014. Inverse kinematics of serial manipulators in cluttered environments using a new paradigm of particle swarm optimization. *IFAC Proceedings Volumes,* 47**,** 8475-8480.

GRAETZ, G. & MICHAELS, G. 2018. Robots at work. *Review of Economics and Statistics,* 100**,** 753-768.

HAO, W. G., LECK, Y. Y. & HUN, L. C. 6-DOF PC-Based Robotic Arm (PC-ROBOARM) with efficient trajectory planning and speed control. 2011 4th International Conference on Mechatronics (ICOM), 2011. IEEE, 1-7.

HUANG, H.-C., CHEN, C.-P. & WANG, P.-R. Particle swarm optimization for solving the inverse kinematics of 7-DOF robotic manipulators. 2012 IEEE international conference on systems, man, and cybernetics (SMC), 2012. IEEE, 3105-3110.

JUNIOR, J. D. L. S., DE OLIVEIRA JESUS, R. C., MOLINA, L., CARVALHO, E. A. N. & FREIRE, E. O. FRPSO: Inverse Kinematics Using Fully Resampled Particle Swarm Optimization. 2018 Latin American Robotic Symposium, 2018 Brazilian Symposium on Robotics (SBR) and 2018 Workshop on Robotics in Education (WRE), 2018. IEEE, 402-407.

MEHMOOD, N., IJAZ, F., MURTAZA, Z. & SHAH, S. I. A. Analysis of end-effector position and orientation for 2P-3R planer pneumatic robotic arm. 2014 (iCREATE, 2014). IEEE, 47-50.

MOORE, T. 1985. Robots for nuclear power plants. *IAEA Bulletin,* 27**,** 31-38.

PARK, S. & LEE, M. C. 7DOFs Robot Numerical Approach Method with Jacobian. 2018 International Conference on Information and Communication Technology Robotics (ICT-ROBOT), 2018. IEEE, 1-4.

POLI, R., KENNEDY, J. & BLACKWELL, T. 2007. Particle swarm optimization. *Swarm intelligence,* 1**,** 33-57.

ROKBANI, N. & ALIMI, A. M. 2013. Inverse kinematics using particle swarm optimization, a statistical analysis. *Procedia Engineering,* 64**,** 1602-1611.

SICILIANO, B. 1990. Kinematic control of redundant robot manipulators: A tutorial. *Journal of intelligent and robotic systems,* 3**,** 201-212.

SICILIANO, B., SCIAVICCO, L., VILLANI, L. & ORIOLO, G. 2010. *Robotics: modelling, planning and control*, Springer Science & Business Media.

ZHANG, X. & NELSON, C. A. 2011. Multiple-criteria kinematic optimization for the design of spherical serial mechanisms using genetic algorithms. *Journal of Mechanical Design,* 133**,** 011005.