УДК 621.865.8(045)
DOI: 10.22213/2413-1172-2019-3-33-41

SOLVING THE KINEMATICS OF AN AERIAL HUMAN-LIKE MANIPULATOR USING THE DIFFERENTIAL EVOLUTION ALGORITHM *

I.N. Ibrahim, Post-graduate, Kalashnikov ISTU, Izhevsk, Russia
M.A. Al Akkad, PhD in Engineering, Associate Professor, Kalashnikov ISTU, Izhevsk, Russia

This paper concentrates on deriving the real-time kinematics solution of a manipulator attached to an aerial vehicle, while the vehicle's movement itself is not analyzed. The manipulator kinematics solution using the Denavit - Hartenberg model was introduced, too.

The fundamental scope of this paper is to get a global online solution of the design configurations with a weighted objective function subject to some constraints. Adopting the resulted forward kinematics equations of the manipulator, the trajectory planning problem turns into an optimization task. Several and well-known computing methods are documented in the literature for solving constrained complicated nonlinear functions, where in this study the differential evolution algorithm is adopted, which is a combination of a mathematical search method and an evolution algorithm.

It is a constrained metaheuristic and population-based approach. Moreover, it is able to solve the inverse kinematics problem considering the mobile platform, in addition to avoiding singularities, since it does not demand the inversion of a Jacobian matrix.

Simulation experiments were carried out for trajectory planning of the sixth degree of freedom aerial manipulator and the obtained results for three different target points confirmed the feasibility and effectiveness of the suggested method.

Keywords: manipulator, inverse kinematics, metaheuristics methods, evolution algorithms, optimization methods, differential evolution algorithm.

Introduction

The inverse kinematics (IK) solver is a primary problem in robotic manipulation, particularly when real-time and precision in calculations are demanded. Mathematically, the numerical solution of kinematics is intricate because of the high degree of nonlinearity. Furthermore, Linear and dynamic programming techniques usually fail or reach local optimum in solving NP-hard problems with a large number of variables and non-linear objective functions. Moreover, traditionally Jacobian-based solutions are identified to scale inadequately with the high number of degrees of freedom (DOF) in addition to singularities existence [2]. In contrast, in [3] a comparative study of several methods based on the Jacobian matrix was presented, clarifying that the modified Levenberg - Marquardt method is much better for a quite large set of random configurations than others but may lose convergence compared to Jacobian transpose and pseudocode inverse methods. Recently many researchers proposed a new method for solving real-time IK without using the Jacobian matrix based on the position of end-effector (EE), using numerical and analytical mathematical tools but did not mention exactly the performance as the time consuming to get the solution [4]. In [5] also similar method for $(2n + 1)$ DOF hyper-redundant manipulator arm was applied. Authors in [6] combined two methods as a real-time IK solver for a human-like arm manipulator based on closed-form analytical equations for a given position. While otherspresented an on-line adaptive strategy based on the Lyapunov stability theory, in addition to Radial Basis Function Network (RBFN) and quadratic programming, which requires complex hardware resources [7].The simulation was done for the position of EE in addition to avoid obstacles and was conducted on PA-10 a 7-DOF manipulator. In [8] a kinematic and time-optimal trajectory planning was considered for redundant robots, two approaches were presented, joint space decomposition and a numerical null-space method for a given pose. They were tested on 7-DOF industrial robots and demanded high consuming time for resolving IK. Nowmetaheuristic optimization algorithms are an encouraging alternative approach to traditional IK techniques due to their strong performance on

© Ibrahim I.N., Al Akkad M.A., 2019
* This research is funded by Kalashnikov Izhevsk State Technical University grant 15.06.01/18AAИ.
Работа выполнена при поддержке гранта Ижевского государственного технического университета имени М. Т. Калашникова 15.06.01/18ААИ.

challenging and high-DOF problems in many various domains, the solution can be solved by minimizing an objective function, allowing the EE to follow the desired path avoiding obstacles and dynamics singularities. In [9] it was explained and proved that differential evolution (DE) algorithm has emerged as one of the most powerful and versatile global numerical optimizers for non-differential and multimodal problems, they showed challenges of the variants of DE which may provide less time and more robustness in solving IK. In [10] a quadratic programming with branching idea was presented with a weighted multi-objective function, which gave a short-time response while [11] showed a comparative research of four different heuristic optimization algorithms GA, PSO, QPSO and GSA for a 4-DOF manipulator in order to reach the target position. In [12] a comparative study of IK solver for a mobile manipulator using DE algorithm was presented. It was concluded that hybrid DE with biogeography-based optimization called HBBO provides good results but a higher computational cost for weighted fitness function and pose target, in contrast, DE proved to be superior to PSO, CS, and TLBO, additionally the PSO algorithm verified that it does not solve the inverse kinematic problems correctly. In [13] a developed methodology was applied to a synthesized six-bar mechanism, it used DE with geometric centroid of precision positions technique (GCCP). In [14] DE was used to improve the design of a fuzzy controller for a wall-following hexapod robot. In [15] a modified self-adaptive DE was proposed in order to improve the static force of humanoid robots, showing robust, safe, reliable performance compared with other metaheuristics. While [16] presented an approximation tool for the inverse model of the industrial robot based on an adaptive neural model optimized by advance DE.

The work in this paper is an extension of the work in [17–19]. The proposed algorithm isthe DE algorithm, which is characterized as accurate and fast converging in discovering the solution as mentioned in [18]. Initially, we define an objective function to minimize the error between the desired and the actual end-effector pose. The objective function considers the minimal movement between the previous and the actual joint configurations. To overcome the constrained problems, we use a penalty function to penalize all those manipulator configurations that violate the allowed joint boundary. Hence, the proposed approach estimates the feasible manipulator configuration needed to reach the desired end-effector pose.

Manipulator Kinematics

In order to determine the relationship between the coordinate frames, which are assigned to robots' links and joints, homogeneous transformations are required. Three parameters are employed to describe the rotation while another three parameters are used to define the translation. Accordingly, the Denavit - Hartenberg (DH) convention was used to describe kinematically the rigid motion by assigning the values of four quantities for each link, two describe the link itself, and two describe the link's connection to a neighboring link. Where Q, a, d and α are the joint angle, link length, link offset and link twist between joints. While T_i is the homogeneous transformation matrix between the frames that is a function of θ while the other three parameters are constant. The position of all links of an arm-part manipulator can be specified with a set of 6 joint variables from the shoulder's joints Figure 1. This set of variables is often referred to as a 6×1 joint vector [17, 18].

The data in Table 1 represent link parameters of the arm-part based on DH strategy in two formulas: standard and modified DH. Whereas the standard simulation form of LabVIEW Robotics module was used, in order to validate the design.

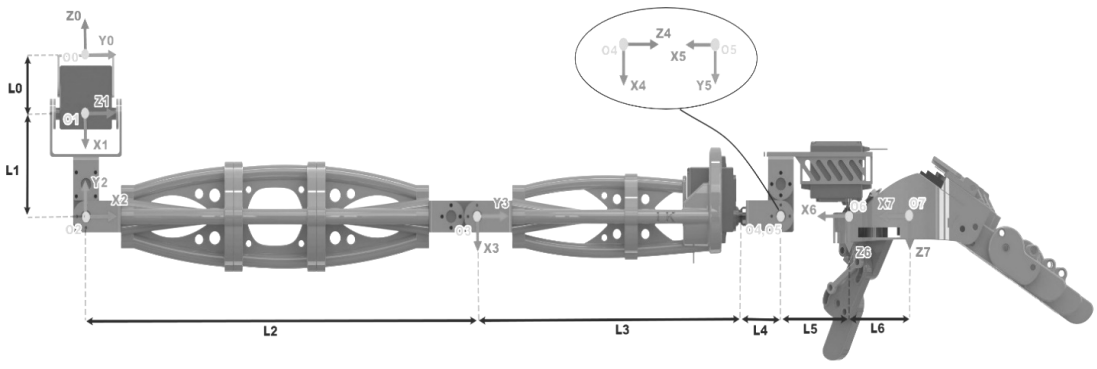

Fig. 1. The manipulator with its joints and links. It has seven links and six revolute joints in the arm-part while the hand-part contains 5 fingers. Each joint represents a single DOF

Table 1. **Link parameters of the manipulator's arm-part**

Modified Denavit Hartenberg					Standard Denavit Hartenberg					
α_{i-1}	a_{i-1}[cm]	d_i[cm]	θ_i	θ_i (Initial value)	α_i	a_{i-1}[cm]	d_i[cm]	θ_i	θ_i (Initial value)	Joint offset
$-\pi/2$	l_0	0	θ_1	$\pi/2$	$-\pi/2$	6.4	0	θ_1	0	0
$\pi/2$	l_1	0	θ_2	$-\pi/2$	0	30.2	0	θ_2	$-\pi/2$	$-\pi/2$
0	l_2	0	θ_3	$-\pi/2$	$\pi/2$	0	0	θ_3	$\pi/2$	$\pi/2$
$-\pi/2$	0	l_3+l_4	θ_4	0	$\pi/2$	0	23.5	θ_4	0	0
$\pi/2$	0	0	θ_5	$-\pi/2$	$-\pi/2$	5.3	0	θ_5	$\pi/2$	$\pi/2$
$-\pi/2$	l_5	0	θ_6	0	0	5.6	-2	θ_6	0	0

The space of all joint variables is referred to as the joint-space $\Theta=[\theta_1,\theta_2,\ldots,\theta_6]^T$. Here we have been concerned with computing the Cartesian space representation from the knowledge of the joint-space information. Hence, the homogeneous transformations of the links were used ${}_i^{i-1}T$. If the robot's joint-position sensors are estimated by servomechanisms, the Cartesian position and orientation of the hand-part can be computed by ${}_7^0T$ [17, 18].

$$ {}_7^0T=\begin{bmatrix} b_{11} & b_{12} & m_{12} & b_{11}l_6 \\ b_{21} & b_{22} & m_{22} & b_{21}l_6 \\ b_{31} & b_{32} & m_{32} & b_{31}l_6 \\ 0 & 0 & 0 & 1 \end{bmatrix}, $$

where

$$ b_{11}=\left(\left(c\theta_1c\theta_{23}c\theta_4-s\theta_1s\theta_4\right)c\theta_5-c\theta_1s\theta_{23}s\theta_5\right)c\theta_6+ $$
$$ +(-s\theta_5r_{11}-s\theta_5c\theta_1s\theta_{23})s\theta_6; $$
$$ b_{12}=-\left(\left(c\theta_1c\theta_{23}c\theta_4-s\theta_1s\theta_4\right)c\theta_5-c\theta_1s\theta_{23}s\theta_5\right)s\theta_6+ $$
$$ +(-c\theta_1c\theta_{23}s\theta_4-s\theta_1c\theta_4)c\theta_6; $$
$$ m_{12}=-s\theta_5\left(c\theta_1c\theta_{23}c\theta_4-s\theta_1s\theta_4\right)-c\theta_1s\theta_{23}s\theta_5; $$
$$ b_{21}=\left(c\theta_5s\theta_{23}c\theta_4+s\theta_5c\theta_{23}\right)c\theta_6-s\theta_6s\theta_4s\theta_{23}; $$
$$ b_{22}=-\left(c\theta_5s\theta_{23}c\theta_4+s\theta_5c\theta_{23}\right)s\theta_6-c\theta_6s\theta_4s\theta_{23}; $$
$$ m_{22}=-s\theta_5s\theta_{23}c\theta_4+c\theta_5c\theta_{23}; $$
$$ b_{31}=\left(\left(s\theta_1c\theta_{23}-c\theta_1s\theta_4\right)c\theta_5+s\theta_1s\theta_{23}s\theta_5\right)c\theta_6+ $$
$$ +(s\theta_1c\theta_{23}s\theta_4+c\theta_1c\theta_4)s\theta_6; $$
$$ b_{32}=-\left(\left(s\theta_1c\theta_{23}-c\theta_1s\theta_4\right)c\theta_5+s\theta_1s\theta_{23}s\theta_5\right)s\theta_6- $$
$$ -(s\theta_4c\theta_{23}s\theta_1+c\theta_1c\theta_4)c\theta_6; $$
$$ m_{32}=-s\theta_5\left(s\theta_1c\theta_{23}-c\theta_1s\theta_4\right)+s\theta_1s\theta_{23}c\theta_5. $$

The movement of the arm part is used to control the motion of the hand-part located at frame 7 in the workspace related to the base frame. Moreover, the motion ranges of the joints are shown in Table 2 which were readjusted to be more fitting for accomplishing more tasks compared to the joints of the human arm.

Table 2. **Motion range of the manipulator's arm-part**

	angle	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
Arm-part	range	–90 → +90	47 → +115	–90 → +15	–90 → +90	–90 → +25	–90 → +90

Proposed Optimization Techniques for solving kinematics

The evolutionary optimization algorithms can solve the complicated nonlinear equations completely and efficiently. The solution of the inverse kinematics for the manipulator is a very difficult problem to obtain by traditional approaches. Besides, the suggested strategies do not require the inversion of any Jacobian matrix, and then it avoids singularities configurations. In this paper, to optimize this problem, the differential evolution algorithm was used. In general, this optimization technique is based on the forward kinematics equations, which always produces a solution in cooperation with an objective function. Hence, the general aspect of the problem can be expressed as minimizing $J(\Theta)$, constrained by $\Theta_{\min}\le\Theta\le\Theta_{\max}$. Furthermore, the objective function could be defined as the weighted sum of the errors as follows

$$ J(\Theta)=\sigma P_{error}(\Theta)+\varepsilon O_{error}(\Theta)= $$
$$ =\sigma\left\|P_G-P_E(\Theta)\right\|+\varepsilon\left\|O_G-O_E(\Theta)\right\|, \qquad (1) $$

where $P_{error}(\Theta)$ and $O_{error}(\Theta)$ represent the position and orientation errors respectively and could be computed as a difference in distance between the target and current position, in this work we used an Euclidean formula as a representation of distance. While the parameters σ and ε are the weights of the position and the orientation, respectively. Let $G=(P_G,O_G)$ be a given target end-effector pose

while $E(\Theta)=\left(P_E(\Theta),O_E(\Theta)\right)$ is the current end-effect or pose in the workspace corresponding to configuration $\Theta=\left[\theta_1,\theta_2,\ldots,\theta_6\right]^T$ which can be calculated using forward kinematics, where *P* refers to the 3D position vector of pose while *O* refers to the vector of Roll-Pitch-Yaw Euler angles of pose (in radians), respectively. Which the optimization algorithms are exploring directly in the configuration space of the manipulator. Hence, each individual $\Theta_i=\left[\theta_{i,1},\theta_{i,2},\ldots,\theta_{i,j},\ldots,\theta_{i,6}\right]^T$ represents the i^{th} candidate set of joint angles. Henceforward, at each iteration, we evaluate each candidate configuration Θ_i by passing it through the forward kinematics module and measuring the position and orientation error between where the end-effect or would be at configuration Θ_i and the target end-effect or pose. In order to enforce joint limits, each dimension *j* of element Θ_i should be limited to searching in the range of valid joint angles $\Theta_i\in[\Theta_{\min},\Theta_{\max}]$. This can be realized by clamping each dimension *j* within these bounds at each iteration immediately after it is updated.

Differential Evolution Algorithm

The DE algorithm was introduced by Storn and Price [1] and studied in [9, 15, 18]. It is one of the most powerful stochastic population-based optimization algorithms. It was invented to optimize functions in an n-dimensional continuous domain. moreover, it occupies several benefits such as simple implementation, good performance, global optimization, robust, low space complexity, converges fast, and has a good balance between exploration and exploitation. The DE algorithm can be considered both as an evolution algorithm and as a mathematical technique, because it uses the concepts of population and evolution, in addition to using a mathematical searching method.

The initialized to a uniform sampling of the instance space, are continuously enhanced by periodically adding a scaled variant of the difference vector to a third individual to generate a new candidate solution and then producing the succeeding generation. DE consists of four stages: initialization, mutation, crossover, and selection. The last three of these are iterated until a termination condition such as the maximum number of generations is reached. Nevertheless, unlike other evolutionary algorithms before-mentioned as evolution strategies, mutation is performed by applying the scaled difference between members of the population. This has the impact of adjusting the step size to the fitness aspect over time. The implementation of this method is illustrated in Algorithm 1.

Algorithm 1. **The pseudo-code of the differential evolution algorithm**

Initialization:
$Population^{(1)}\leftarrow\left\{\Theta_1^{(1)},\Theta_2^{(1)},\ldots,\Theta_i^{(1)},\ldots,\Theta_{NP}^{(1)}\right\}$,
$g\leftarrow 1, g_{\max}$
Evolution Process:
While *Termination criteria not met* do
for $i\leftarrow 1, NP$ *do*
Mutation Process: $v_i^{(g)}\leftarrow mutate\left(\Theta_i^{(g)}\right)$
Crossover Process: $u_i^{(g)}\leftarrow crossover\left(\Theta_i^{(g)},v_i^{(g)}\right)$
Selection Process:
if $f\left(u_i^{(g)}\right)\le f\left(\Theta_i^{(g)}\right)$ *then*
insert $u_i^{(g)}$ int *o* $population^{(g+1)}$
else
insert $\Theta_i^{(g)}$ int *o* $population^{(g+1)}$
end if
end for
$g\leftarrow g+1$
end while

The trajectory planning strategy can be transformed into an optimization issue with multiple constraints. Firstly, it demands to determine the dimension of the population *NP*, the generation number *g* with maximum $g_{\max}$, the dimension real-valued of the individual is equal to the configuration space of the manipulator, the scale factor *F*, and the crossover factor C_r. Then individuals in the population are expressed by: $\Theta_i^{(g)}=\left(\theta_{i,1}^{(g)},\theta_{i,2}^{(g)},\ldots,\theta_{i,6}^{(g)}\right);\ i=1,2,\ldots,NP,$ represents the design variable of the *i*-th individual in generation *g*. DE begins by initializing a population of *NP* to cover as much as possible of the exploration space constrained by the minimum and maximum bounds $\Theta_{\min}=\left[\theta_{\min,1},\theta_{\min,2},\ldots,\theta_{\min,i},\ldots,\theta_{\min,6}\right]^T$ and $\Theta_{\max}=\left[\theta_{\max,1},\theta_{\max,2},\ldots,\theta_{\max,i},\ldots,\theta_{\max,6}\right]^T$. Hence, the *i*-th individual may then be initialized as: $\theta_{i,j}^{(1)}=\theta_{\min,j}+\text{rand}(0,1)\left[\theta_{\max,j}-\theta_{\min,j}\right]$, with $\text{rand}(0,1)$ being a uniformly random value between 0 and 1. Henceforward, The mutant strategy is adopted after initialization to generate a donor vector $\nu_i^{(g)}=\left(\upsilon_{i,1}^{(g)},\upsilon_{i,2}^{(g)},\ldots,\upsilon_{i,6}^{(g)}\right)$ by its corresponding target vector $\Theta_i^{(g)}$.

The following have been proposed in [9, 18]:
DE/rand/1:

$$\nu_i^{(g)} = \Theta_{r_1}^{(g)} + F_i\left(\Theta_{r_2}^{(g)} - \Theta_{r_3}^{(g)}\right)$$

DE/best/2:

$$\nu_i^{(g)} = \Theta_{best}^{(g)} + F_i\left(\Theta_{r_1}^{(g)} - \Theta_{r_2}^{(g)}\right) + F_i\left(\Theta_{r_3}^{(g)} - \Theta_{r_4}^{(g)}\right)$$

DE/current-to-best/1:

$$\nu_i^{(g)} = \Theta_i^{(g)} + F_i\left(\Theta_{best}^{(g)} - \Theta_i^{(g)}\right) + F_i\left(\Theta_{r_1}^{(g)} - \Theta_{r_2}^{(g)}\right)$$

either or: this strategy merges two methods to generate the donor vector [26].

$p_f \leftarrow$ *mutation probability* $\in [0,1]$,
$a \leftarrow$ *random number* $\in [0,1]$
if $a < p_f$ *then*
use DE / rand / 1:

$$\nu_i^{(g)} = \Theta_{r_1}^{(g)} + F_i\left(\Theta_{r_2}^{(g)} - \Theta_{r_3}^{(g)}\right)$$

else
use DE / rand / 2:

$$\nu_i^{(g)} = \Theta_{r_1}^{(g)} + \mathrm{K}\left(\Theta_{r_2}^{(g)} - \Theta_{r_1}^{(g)} + \Theta_{r_3}^{(g)} - \Theta_{r_1}^{(g)}\right)$$

end if

Where, F_i is the scaling factor within 0 and 1, indices r_1, r_2, r_3, and r_4 are randomly selected integers in the range $[1, NP]$, such that $r_1 \neq r_2 \neq r_3 \neq r_3 \neq i$. $\Theta_{best}^{(g)}$ is the best individual in the current population, also p_f and a are the mutation probability and random number, respectively. At that point, a crossover between $\nu_i^{(g)}$ and $\Theta_i^{(g)}$ is performed to generate a trial vector $u_i^{(g)} = \left(\mu_{i,1}^{(g)}, \mu_{i,2}^{(g)}, \ldots, \mu_{i,6}^{(g)}\right)$. Two methods were used in this paper, a binomial and an exponential crossover procedure [8]. The binomial crossover provides a trial vector by selecting an element from the donor vector whenever a randomly produced value formed from a uniform distribution is below the crossover rate C_r. Additionally, an element h is randomly taken per iteration to always come from a donor vector as follows:

$$\mu_{i,j}^{(g)} = \begin{cases} \upsilon_{i,j}^{(g)} & \textit{if } i = h \textit{ or } \mathrm{rand}(0,1) \le C_r \\ \theta_{i,j}^{(g)} & \textit{otherwise.} \end{cases}$$

Exponential crossover tries to exploit relationships between adjacent elements. It works by choosing a random starting element and selecting the next L consecutive elements in a circular manner from the donor vector. The number of elements L is calculated as follows:

Algorithm 2. **Exponential crossover**

$L \leftarrow 0$
repeat
$L \leftarrow 0$
until $\mathrm{rand}(0,1) > C_r$ *or* $L > D$

After crossover, the objective function as explained in Eq. 1 is evaluated for the trial vector $u_i^{(g)}$. According to the greedy selection only, as shown in algorithm 1. Afterward, the better of $u_i^{(g)}$ and $\Theta_i^{(g)}$ will be picked to remain into the next generation.

Simulation Results

In this work, we solve inverse kinematics of the redundant manipulator with six joints to follow a destination pose. The manipulator's joints are $\theta_j, J = 1, 2, \ldots, 6.$ The DH parameters are presented in Table 1. In the inverse kinematics experiments, the desired end-effector pose for the arm-part of the manipulator was determined as a variable $G = \left(P_G, O_G\right) = \left(x, y, z, roll, pitch, yaw\right) = \left(-20, 3, 40, 0, 10, 15\right)$. Moreover, the parameters of the objective function were adjusted as follows $\varepsilon = 1 - \beta = 0.7$ so there is a balance between position and orientation to be optimized. In case of DE algorithm, Table 3 shows DE settings while Table 4 presents the results of utilizing DE for some scenarios.

Table 3. **Setting of the DE Algorithm**

Mutation Method	Random
Scale Factor	0.9
Crossover Method	Uniform
Crossover Probability	0.95

As presented in Table 4, the purpose of these experiments is to find the iteration and population-which achieves a minimum error and execution time. The total error was obtained using thefollowing formula:

$$E_{Total} = \sum_{i}^{iterations} \left[\left(P_{R_i} - P_G\right) + \left(O_{R_i} - O_G\right)\right].$$

This formula computes the error in position and orientation for the end-effector in each iteration. The algorithm gives multiple solutions after each

iteration because of the redundant nature of our manipulator. The total error is computed for all iterations when applying the algorithm with custom parameters. While the execution time was obtained by calculating the time consumption for the algorithm to reach the iterations. It is clear that the 9th test is the proper solution with a convergence time equal to 184 ms, and a total error equal to 0.00379635. Taking into consideration that the adaptation of the DE parameters nearby the setting of this result may improve the solution to be more fitting but with longer convergence time. As presented in Table 4, it is obvious that the execution time depends on the size of the population and the iterations, respectively. Further, the population size achieves the diversity feature which let the algorithm explores more solutions in the workspace while a high iteration gives a solution much closer to the target.

Here, we have applied the algorithm on multiple targets as shown in Table 5.

These target points were taken from real experiments of the manipulator for the task of reaching and grasping an object. Here we are going to validate the response for the total error and execution time. Figure 2 shows the values of the objective function for those new targets and the solutions for them were presented in Table 6.

Table 4. **Inverse Kinematics Results of the Differential Evolution Algorithm**

Test No.	Population	Iterations	$J(\Theta)$	Total error	Execution time [ms]	Reaching target $(x, y, z, roll, pitch, yaw)$
1	6	250	4.60464	–8.22166	247	$(-20.0619, 3.00659, 40.0325, 3.74, -4.49369, 17.5544)$
2	6	500	1.53162	1.78975	594	$(-19.56, 2.64638, 39.8767, 0.465805, 13.2258, 13.135)$
3	8	600	1.19735e-5	8.26589e-6	861	$(-20, 3, 40, -8.25797e-6, 10, 15)$
4	8	800	3.60207e-7	6.13326e-7	1125	$(-20, 3, 40, 2.59552e-8, 10, 15)$
5	10	500	0.000230295	0.000677389	986	$(-19.99, 3.0001, 40.0001, 4.21087e-5, 10.0004, 15)$
6	10	750	1.55363e-7	1.13202e-7	1301	$(-20, 3, 40, 1.13202e-7, 10, 15)$
7	10	1000	4.53651e-9	–3.13365e-9	1917	$(-20, 3, 40, -1.40416e-9, 10, 15)$
8	12	250	0.22408	–0.001729	567	$(-19.809, 3.0609, 40.1352, 0.05281, 10.1526, 15.087)$
9	12	100	0.00231037	0.00193043	184	$(-20.0001, 3.00003, 40, -0.0006738, 9.99932, 14.9998)$
10	12	500	3.42549e-5	0.000107144	1146	$(-20, 3, 40, 3.25415e-5, 10, 15)$
11	20	500	0.000828201	0.00156291	1736	$(-19.9994, 3.00, 40.0002, 0.000503176, 10.00, 14.99)$
12	20	750	1.28825e-6	–4.33271e-6	2628	$(-20, 3, 40, -3.29831e-6, 10, 15)$
13	30	500	0.00107932	–0.00085163	2614	$(-20.0004, 2.999, 39.999, 0.0005774, 9.999, 15.000)$
14	30	1000	9.20132e-7	3.09732e-8	5255	$(-20, 3, 40, 3.978432e-8, 10, 15)$

Table 5. **Applying the algorithm on three target points**

Test no.	$(P_G, O_G) = (x, y, z, roll, pitch, yaw)_G$	$\theta = (\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)$
1	(22.66,18.75,35,107.78,21,65.94)	(19.3134, 108.021, –79.9889, 82.0306, –19.5808, 36.2275)
2	(32.2,8.47,40,90,0.4,50)	(0.189, 77.562, –52.425, 70.65, –34.05, 28.523)
3	(32.2,8.5,40,90,0.4,40)	(–0.956, 72.63, –47.256, 64.066, –42.752, 33.713)

Table 6. **The algorithm response for multiple target points**

Test no.	Population	Iterations	$J(\Theta)$	Total error	Execution time [ms]	Reaching target $(x, y, z, roll, pitch, yaw)$
1	12	100	0.00231037	0.00193043	184	(22.6612, 18.751, 35.0009, 107.779, 21.002, 65.938)
2			0.00127429	–0.00362273	181	(32.199, 8.46, 39.999, 89.997, 0.3998, 50.0005)
3			0.00796475	0.0221528	177	(32.2145, 8.4989, 40.007, 90.013, 0.407, 39.982)

Figure 2 presents the values of the objective function for the first target, while Figure 3 illustrates the position and orientation of the end-effector for the first target after applying the solutions to validate the IK solver. Also Figure 4 and Figure 6 present the values of the objective function for the 2nd and 3rd targets. Furthermore, Figure 5 and Figure 7 show the position and orientation of the end-effector for the 2nd and 3rd targets.

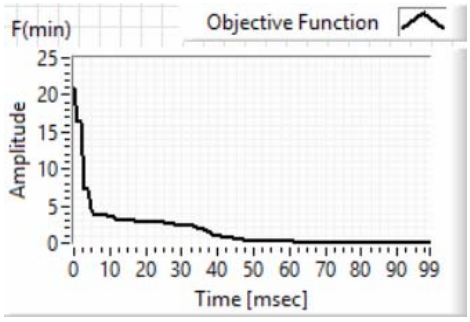

Fig. 2. Displays the values of the objective function for the first target after applying the IK-DE solver

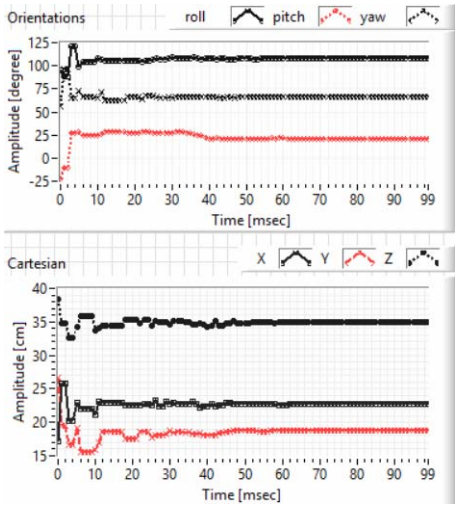

Fig. 3. Illustrates the position and orientation of end-effector for the first target after applying the solutions to validate the IK-DE solver

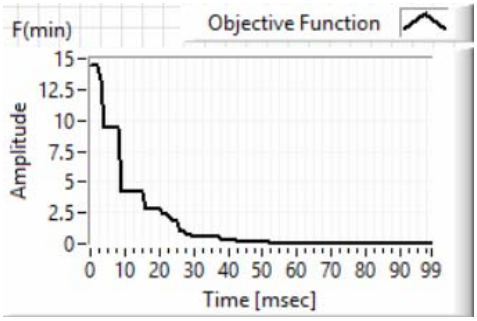

Fig. 4. Displays the values of the objective function for the second target after applying the IK-DE solver

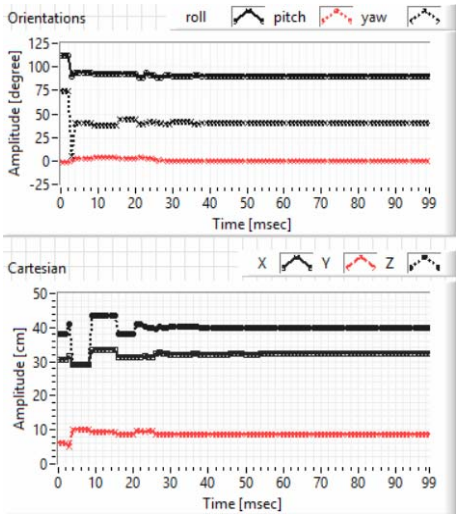

Fig. 5. Illustrates the position and orientation of end-effector for the second target after applying the solutions to validate the IK-DE solver

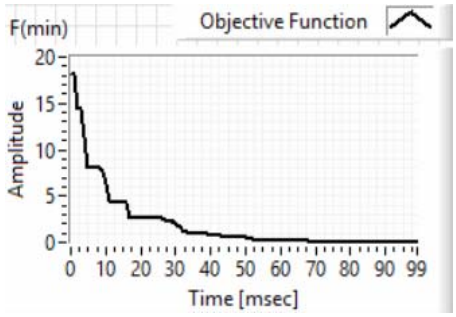

Fig. 6. Displays the values of the objective function for the third target after applying the IK-DE solver

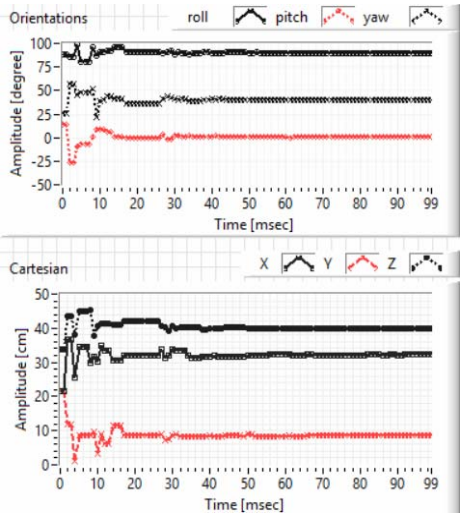

Fig. 7. Illustrates the position and orientation of end-effector for the third target after applying the solutions to validate the IK-DE solver

Conclusion

In comparison with other researchers work, the inverse kinematics of a human-like six joints manipulator to follow a certain pose was solved. The DE algorithm was used and the parameters of the objective function to be optimized were adjusted to have balance between position and orientation. It was obvious that the execution time depends on both the population size and the iterations. The population size achieves the diversity feature, which allows the algorithm to explore more solutions in the workspace while the high iteration gives a solution much closer to the target. The IK solver was validated. Each new solution is considered as a global solution within its iteration, and it grants the algorithm the ability to explore new global solution. Therefore, it is important to alter the settings of the DE algorithm to geta solution based on the objective function in shorter time. The adaptation of the algorithm parameters nearby the setting point may improve the solution to be more fitting but with longer convergence time. The obtained results for three different target points confirmed the feasibility and effectiveness of the suggested method.

References

1. Storn R., Price K. [Differential Evolution - A Simple and Efficient Heuristic for global Optimization over Continuous Spaces]. *Journal of Global Optimization*, 1997, vol. 11, no. 4, pp. 341-359.

2. Buss S.R. [Introduction to inverse kinematics with Jacobian transpose, pseudoinverse and damped least squares methods]. *IEEE Journal of Robotics and Automation*, 2004, 17, pp. 1-19.

3. Dulęba I., & Opałka M. [A comparison of Jacobian-based methods of inverse kinematics for serial robot manipulators]. *International Journal of Applied Mathematics and Computer Science*, 2013, 23, pp. 373-382.

4. Wang X., Zhang D., Zhao C. [The inverse kinematics of a 7R 6-degree-of-freedom robot with non-spherical wrist]. *Advances in Mechanical Engineering*, 2017, 9, 1687814017714985.

5. Ananthanarayanan H., & Ordóñez R. [Real-time Inverse Kinematics of (2n+ 1) DOF hyper-redundant manipulator arm via a combined numerical and analytical approach]. *Mechanism and Machine Theory*, 2015, *91*, pp. 209-226.

6. Tolani D., & Badler N. I. [Real-time inverse kinematics of the human arm]. *Presence: Teleoperators & Virtual Environments*, 1996, 5, pp. 393-401.

7. Toshani H., & Farrokhi M. [Real-time inverse kinematics of redundant manipulators using neural networks and quadratic programming: a Lyapunov-based approach]. *Robotics and Autonomous Systems*, 2014, *62*, pp. 766-781.

8. Reiter A., Müller A., & Gattringer H. [Inverse kinematics in minimum-time trajectory planning for kinematically redundant manipulators]. *Industrial Electronics Society: 42nd Annual Conference of the IEEE*, 2016, pp. 6873-6878. IEEE.

9. Geitle M. [Improving differential evolution using inductive programming]: Master's thesis, 2017.

10. Bodily D.M., Allen T.F., Killpack M.D. [Motion planning for mobile robots using inverse kinematics branching]. *Robotics and Automation (ICRA): International Conference IEEE*, 2017, pp. 5043-5050. IEEE.

11. Ayyıldız M., Çetinkaya K. [Comparison of four different heuristic optimization algorithms for the inverse kinematics solution of a real 4-DOF serial robot manipulator]. *Neural Computing and Applications*, 2016, 27, pp. 825-836.

12. López-Franco C., Hernández-Barragán J., Alanis A.Y., Arana-Daniel N., López-Franco M. [Inverse kinematics of mobile manipulators based on differential evolution]. *International Journal of Advanced Robotic Systems*, 2018, 15, 1729881417752738.

13. Shiakolas P.S., Koladiya D., Kebrle J. [On the optimum synthesis of six-bar linkages using differential evolution and the geometric centroid of precision positions technique]. *Mechanism and Machine Theory*, 2005, 40, pp. 319-335.

14. Juang C.F., Chen Y.H., Jhan Y.H. [Wall-following control of a hexapod robot using a data-driven fuzzy controller learned through differential evolution]. *IEEE Transactions on Industrial electronics*, 2015, 62, pp. 611-619.

15. Pierezan J., Freire R.Z., Weihmann L., Reynoso-Meza G., dos Santos Coelho L. [Static force capability optimization of humanoids robots based on modified self-adaptive differential evolution]. *Computers & Operations Research*, 2017, 84, pp. 205-215.

16. Ngoc Son N., Anh H.P.H., Thanh Nam N. [Robot manipulator identification based on adaptive multiple-input and multiple-output neural model optimized by advanced differential evolution algorithm]. *International Journal of Advanced Robotic Systems*, 2016, 14, 1729881416677695.

17. Ibrahim I.N. [Ultra-Light Weight Robotic Manipulato]. *Vestnik IzhGTU imeni M.T. Kalashnikova*, 2018, vol. 21, no. 1, pp. 12-18 (in Rus.). DOI: 10.22213/2413-1172-2018-1-12-18.

18. Ibrahim I.N. [A Comparative Study for an Inverse Kinematics Solution of an Aerial Manipulator Based on the Differential Evolution Method and the Modified Shuffled Frog-Leaping Algorithm]. *Mekhatronika, avtomatizatsiya, upravlenie*, 2018, vol. 19, no. 11, pp. 714-724 (in Russ.).

19. Ibrahim I.N., Al Akkad M.A. [Studying the Disturbances of Robotic Arm Movement in Space Using the Compound-Pendulum Method]. *Vestnik IzhGTU imeni M.T. Kalashnikova*, 2017, vol. 20, no. 2, pp. 156-159 (in Russ.).

Исследование кинематики для манипулятора беспилотного летательного аппарата на основе дифференциального алгоритма эволюции

И. Н. Ибрахим, аспирант, ИжГТУ имени М. Т. Калашникова, Ижевск, Россия
М. А. Аль Аккад, кандидат технических наук, доцент, ИжГТУ имени М. Т. Калашникова, Ижевск, Россия

Рассмотрено кинематическое решение в реальном времени для манипулятора, прикрепленного к беспилотному летательному аппарату; движение самого транспортного средства в данном исследовании не анализируется. Представленное кинематическое решение для манипулятора основано на модели Денавита – Хартенберга.

Основной целью исследования является получение глобального решения в реальном времени для конфигурации проектирования с взвешенной целевой функцией с наложением некоторых ограничений. Применение уравнений прямой кинематики манипулятора, полученных в результате исследования, позволяет превратить задачу планирования траектории в задачу оптимизации.

Хорошо известны несколько типов вычислительных методов для решения ограниченных сложных нелинейных функций. При этом предлагается дифференциальный алгоритм эволюции, который является комбинацией математического метода поиска и алгоритма эволюции. С его помощью представляется возможным решение обратной кинематической задачи с учетом мобильности платформы. Кроме того, данный метод предотвращает появление сингулярных точек, поскольку он не требует инверсии матрицы Якоби.

Результаты экспериментального моделирования для планирования траектории манипулятора с шестью степенями свободы подтвердили целесообразность и эффективность предлагаемого метода.

Ключевые слова: манипулятор, обратная кинематика, метаэвристические методы, эволюционный алгоритм, методы оптимизации, дифференциальный алгоритм эволюции.

Получено 04.03.2019

For Citation

Ibrahim I.N., Al Akkad M.A. [Solving the Kinematics of an Aerial Human-like Manipulator Using the Differential Evolution Algorithm]. *Vestnik izhGTU imeni M.T. Kalashnikova*, 2019, vol. 22, no. 3, pp. 33-41. DOI: 10.22213/2413-1172-2019-3-33-41.