DOI: 10.22213/2658-3658-2019-40-50

The Investigation of Trajectory Control for an Anthropomorphic Manipulator Attached to a Vehicle

I. N. Ibrahim [1], M. A. Al Akkad [2]

[1] Mechatronics Department, Kalashnikov Izhevsk State Technical University, Izhevsk, Russian Federation
E-mail: ibrncfe@gmail.com
[2] Computer Science Department, Kalashnikov Izhevsk State Technical University, Izhevsk, Russian Federation
E-mail: aimanakkad@yandex.ru

Received: July 25, 2019

Abstract. This paper concentrates on deriving an inverse kinematics solution of a manipulator attached to an aerial vehicle for real-time applications. Analyzing the vehicle's movement itself is not considered. The kinematics solution using Denavit-Hartenberg model was introduced. Adopting the resulted forward kinematics equations of the manipulator, the trajectory planning problem turns into an optimization task. For solving constrained complicated nonlinear functions, shuffled frog leaping search algorithm is suggested to get a global online solution of the design configurations with a weighted objective function subject to some constraints. 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 have been carried out for the trajectory planning of a six degree of freedom aerial manipulator, and the obtained results confirmed the feasibility and effectiveness of the suggested method.

Keywords: Inverse Kinematics, Metaheuristics Methods, Evolutionary Algorithms, Optimization Techniques, Shuffled Frog Leaping Algorithm

1. INTRODUCTION

Metaheuristic optimization algorithms are an encouraging alternative to traditional numerical solution methods of inverse kinematics (IK), when working in real-time and precision is required. Furthermore, the linear and dynamic programming techniques usually fail to reach local optimum in solving NP-hard problems with a large number of variables and non-linear objective functions. This paper focuses on population-based heuristic search methods for optimization problems and on memetic algorithms (MAs), where memes propagate themselves in the meme pool by leaping from brain to brain via a process that can be called imitation. This idea can be applied to a robotic agent to use MAs in resolving its movement in the workspace. The solution can be achieved by minimizing an objective function allowing the end-effector to follow the optimal path to avoid singularities and obstacles. A special type of optimization algorithm was developed and deployed for the solving the IK of a humanlike manipulator.

Jacobian-based solutions are identified to scale inadequately with the high number of degrees of freedom [1], in addition to singularities existence. A comparative study of several methods based on the Jacobian matrix [2], clarified that the modified Levenberg-Marquardt method is much better for a large set of random configurations, but may lose convergence compared to Jacobian transpose and pseudocode inverse methods. For solving real-time IK without using the Jacobian matrix, numerical and analytical mathematical tools based on the end-effector position were proposed, but without mentioning the solution time consumption [3]. A similar method for (2n+1) DoF hyper-redundant manipulator arm was also applied [4]. Two methods were combined as a real-time IK solver for a human-like arm manipulator based on closed-form analytical equations for a given position [5]. On-line adaptive strategy based on Lyapunov stability theory, radial basis function network (RBFN) and quadratic programming was proposed, but it requires complex hardware resources [6], the simulation was done for the end-effector position in addition to avoid obstacles and was conducted on the 7-DoF PA-10 robot manipulator. A kinematic and time-optimal trajectory planning was considered for redundant robots, two approaches were presented, joint space decomposition and numerical null-space method for a given pose [7], and were tested by 7-DoF industrial robots, but demanded high time consumption for solving the IK problem. Differential evolution (DE) was explained and proved as one of the most powerful and versatile global numerical optimizers for non-differential and multimodal problems [8], and requires less time and has more robustness in solving the IK problem. Quadratic programming, branching, and a weighted multi-objective function that gave a short-time response of seconds were used [9], while comparative research of four different heuristic optimization algorithms GA, PSO, QPSO and GSA for 4-DoF manipulator in order to reach the target as a position was presented [10]. It was proved that Quantum PSO is the best with average execution time of 1.65 seconds. The performance of many PSO variants was investigated to resolve two DoF IK problems for a given position [11], proving that PSO-VG is the fastest which took less average convergence iterations of 740 for 15 particles. A fitness function was derived and minimized to resolve the pose IK problem based on PSO for multiple DoF up to 180 [12], concluding that the runtime and iterations are 4.22 seconds and 118 respectively for a 9-DoF. A hybrid method called DEMPSO based on DE and Modified PSO algorithms was developed in order to minimize the solution time for the pose, moreover a comparative study for several swarm intelligent optimization algorithms as ABC and ACO algorithms were presented [13]. DEMPSO results showed great advantage concerning execution time for reaching the position similar to the performance of DE for the orientation aim. The simulation was conducted with population size 30 for 10-DoF serial-parallel robot. Comparison of three evolutionary algorithms as GA, PSO, and DE was made [14]. A comparative study of IK solvers for a mobile manipulator using DE algorithm was presented [15], concluding that hybrid DE and 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 verified that the PSO algorithm does not solve the IK problems correctly. A developed methodology was applied on a six-bar mechanism [16], using DE with the geometric centroid of precision positions technique (GCCP). DE was used to improve the design of a fuzzy controller for a wall-following hexapod robot [17]. A modified self-adaptive DE was proposed [18], in order to improve the static force of humanoids robot, showing robust, safe, reliable performance compared with other metaheuristics. An approximation tool for an industrial robot inverse model based on an adaptive neural model optimized by advanced DE was presented [19]. An optimal joint trajectory planning method was proposed using forward kinematics of 7-DoF free-floating space robot based on DE method [20], depicting the general aspect of equality and inequality constraints which govern each joint in the manipulator. Shuf-

fled frog-leaping algorithm SFLA was introduced which is a population-based collaborative search metaphor inspired by natural memes [21]. The effectiveness, suitability, and global optimal resolving have been demonstrated in addition to the short processing time. MSFLA for a high dimensional continuous function optimization was proposed [22]. This method yields strong robustness and best convergence. A comparative study for PSO, SFLA, MSFLA, and MSFLA-EO, designated that MSFLA is better than others, and was assumed for obtaining the optimum preventive maintenance scheduling of generating units in power systems [23]. A comparative study among five evolutionary-based optimization algorithms GA, MA, PSW, ACO, and SFLA was presented [24], showing that SFLA is the best concerning the processing time for solving the F8 function.

In this work, in order to solve the IK of a mobile manipulator the MSFLA algorithm is proposed, which is accurate and has fast convergence in discovering the solution [25], and as an extension of our work in [26][27][28][29]. Initially, we define an objective function to minimize the error between the desired and the actual end-effector pose. The objective function takes into account 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.

2. MANIPULATOR KINEMATICS

The data in Table 1 represent link parameters of the manipulator's arm-part based on Denavit-Hartenberg (DH) convention in two forms: standard and modified. Whereas the standard simulation form of LabView Robotics module was used, in order to validate the design. Where θ, 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 initial values of θ_i form an input to the IK-solver and are important to define the positions of joints of the manipulator in its initial state. Joints must have a constant offset distance and a variable rotation angle.

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

Standard Denavit Hartenberg				
α_i	a_i [cm]	d_i [cm]	θ_i	**Initial Value of** θ_i
$-\pi/2$	6.4	0	θ_1	0
0	30.2	0	θ_2	$-\pi/2$
$\pi/2$	0	0	θ_3	$\pi/2$
$\pi/2$	0	23.5	θ_4	0
$-\pi/2$	5.3	0	θ_5	$\pi/2$
0	5.6	-2	θ_6	0

The position of all links of the manipulator's arm-part can be specified with a set of 6 joint variables from the shoulder's joints till wrist's joints as shown in Figure 1. This set of variables is often referred to as a 6×1 joint vector [25]. The space of all joint variables is re-

ferred to as the joint-space $\Theta = \left[\theta_1, \theta_2, \ldots, \theta_6\right]^T$. 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 ${}^{i-1}_{i}T$. If the robot's joint-position sensors are estimated by servo-mechanisms, the Cartesian position and orientation of the hand-part can be computed by ${}^{0}_{7}T$ [25].

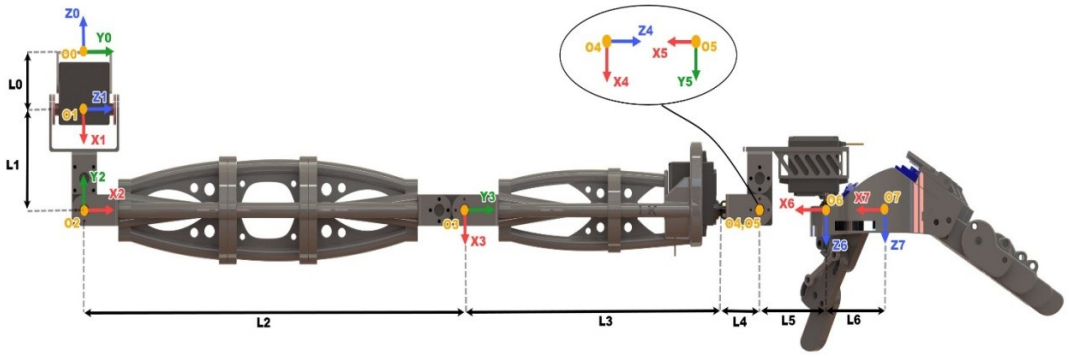

Figure 1. Sketch of the outer shape of the manipulator including its joints and links. It has seven links and six revolute joints in arm-part while the last part is considered as an end-effector of the manipulator and contains 5 fingers. Each joint represents a single DOF

3. Proposed Optimization Techniques for Solving Kinematics

The evolutionary optimization algorithms can solve the complicated nonlinear equations completely and efficiently. The solution of the IK 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 MSFL algorithms is used. 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:

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

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 $E(\Theta) = (P_E(\Theta), O_E(\Theta))$ is the current end-effector 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. Each individual $\Theta_i = \left[\theta_{i,1}, \theta_{i,2} \ldots \theta_{i,j} \ldots \theta_{i,6}\right]^T$ represents an i^{th} candidate set of joint angles. 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-effector would be at configuration Θ_i and the target end-effector 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.

4. MODIFIED SHUFFLED FROG-LEAPING ALGORITHM

The SFLA algorithm is inspired by the memetic evolution of frogs exploring food in a lake, which consolidates the benefits of the genetic-based MAs and by the social behavior-based particle swarm optimization [21]. SFLA incorporates: firstly, the evolution process on each memeplex that embraces different cultures of frogs, where the culture stimulates a fitness value, and serves as a local search within memeplex analogous to PSO algorithm, which imitates the social behavior of the leaping action of frogs searching for food. Secondly the influence by the ideas of other frogs from other memeplexes throughout the shuffling rule. This animates the cooperation process which it implies an adaptation idea and improves the success rate of discovering the solution in the optimization puzzle. In this process, a modification was applied to the frog-leaping action that enhances the exploration manner in the space [22][23]. The randomization strategy in the evolution process offers the algorithm the ability to discover the local best solution within search space stochastically in addition to the communication process that possibly finds a global optimum solution in shorter time. The local search and the shuffling processes continue until the defined convergence criteria are satisfied. The pseudocode of the algorithm is presented in Algorithm 1.

Algorithm 1. The pseudo-code of the Shuffled Frog-Leaping Algorithm

Initialization:
$Population \leftarrow \{\Theta_1, \Theta_2, \ldots, \Theta_i, \ldots, \Theta_{NP}\}$;
$m \leftarrow number\ \ of\ \ memeplexes$;
$n \leftarrow quantity\ \ of\ \ frogs\ \ in\ \ each\ \ memeplex$;
$l \leftarrow 1, iN$
while (*convergence criteria is satisfied Or until met iN*) *do*
  Rank Step: Evaluate each frog Θ_i using a fitness function;
  Partition Step:
  Construct an array *U* of frogs and their fitness's values;
  Sort the array *U* in descending order based on the fitness column;
  Construct $(Y^k; k = 1, \ldots, m)$ memeplexes each including *n* frogs;
  Evaluation Step:
  for ℓ *do*
    for $k \leftarrow 1, m$ *do*
      Determine the worst and best frogs position based on their fitness's values;
      Improve the worst frog position using a leaping distance;
    end for
  end for
  Shuffle Memeplexes Step: combine the evolved memeplexes;
  Check Convergence: Update the population best frog's position Θ_g;
  ℓ ℓ
end while

The MSFLA meta-heuristic strategy is summarized in the following steps:

a. Initialization: construct the population *NP* of frogs randomly similar to the first step in DE algorithm, then Select *m*, and *n*, where *m* is the number of memeplexes and *n* is the number of frogs in each memeplex. The total amount of frogs *NP* can be calculated as $NP = m.n$, additionally, the i^{th} frog is expressed as a vector with a dimension equal to the configuration space as follows: $\Theta_i = \left(\theta_{i,1}, \theta_{i,2}, \ldots, \theta_{i,6}\right);\ i = 1, 2, \ldots, NP.$

b. Rank: compute the performance value f_i for each frog Θ_i. Sort the *NP* frogs in a descending order according to their fitness. Save them in an array: $U = \left\{f_i, \Theta_i; i = 1, 2, \ldots, NP\right\}$, so that $i = 1$ denotes the frog with the best performance value and could save it as a Θ_g in each iteration while the algorithm is running.

c. Partition: partition array *U* into m memeplexes $Y_1, Y_2, \ldots, Y_m$, each including *n* frogs, such that: $Y^k = [\Theta_i^{\,k}, f_i^k \mid \Theta_i^{\,k} = \Theta_{(k+m(i-1))}, f_i^k = f_{(k+m(i-1))}, i = 1, \ldots, n]; k = 1, \ldots, m.$ In this process, the first frog goes to the first memeplex, the second frog goes to the second memeplex, frog *m* goes to the m^{th} memeplex, and frog $m+1$ goes back to the first memeplex, etc.

d. Memetic Evaluation: evolve each memeplex $Y^k; k = 1, \ldots, m$ according to the frog-leaping algorithm as follow. Within each memeplex, the frogs with the best and the worst fitness values are defined as Θ_b and Θ_w. The frog with the global best fitness is defined as Θ_g, then, an improvement process is applied only to the frog with worst fitness in each cycle. Hence, the position of the frog with the worst fitness is modified which emulates the leaping process as follows: leaping distance $D = C_L \,\text{rand}(0,1)\left[\Theta_b - \Theta_w\right]$, then new position $\Theta_w = \Theta_w + D;\quad D \in]-D_{\max}, D_{\max}[$. Where, $\text{rand}(0,1)$ is a random number between 0 and 1, $D_{\max}$ is the maximum allowed change in a frog's position and C_L is the modification of the algorithm which it is a constant indicates the amount of frog-leaping in each memeplex. The evaluation process, for all memeplexes, is repeated by an adaptable number of iterations iM, until no improvement becomes possible.

e. Shuffle Memeplexes: shuffle frogs and replace all memeplexes $Y^k; k = 1, \ldots, m$ into *U*, such that $U = \left\{f_i, \Theta_i; i = 1, 2, \ldots, NP\right\}$ similar to the initialization phase, afterwards sort *U* in order to decrease the performance value, update the population best frog's position Θ_g.

f. Check convergence: if the convergence criteria is satisfied then stop otherwise return to the partition step and continue for a specific quantity of iterations *iN*.

After each iteration the first frog in the sorted list represents a global solution. The number of iterations *iM* specifies the search depth within memeplexes while *iN* governs the solution producing process.

5. Simulation Results

The IK of a redundant manipulator with six joints to follow a destination pose was solved. The manipulator's joints correspond to the variable $\theta_j : j = 1, 2, \ldots, 6$ are constrained. In the IK experiments, the desired end-effector pose for the manipulator's arm-part was determined by this vector $G = \left(P_G, O_G\right) = \left(x, y, z, roll, pitch, yaw\right) = \left(-20, 3, 40, 0, 10, 15\right)$. The parameters of the objective function were adjusted as follows $\varepsilon = 1 - \sigma = 0.7,$ so there is a balance between position and orientation to be optimized. In case of MSFLA, the parameters of

the algorithm were introduced in Table 2, and a summary of the results of utilizing the algorithm for multiple scenarios was introduced in Table 3. These experiments are carried out to show all possibilities of the MSFL algorithm with variations in the number of iterations and in the population size. The purpose is to show the relationship between error and execution time with relation to iterations and population then to adapt the algorithm to take more optimum solution. An execution time of 729 ms for a human-like robotic arm with 6 joints is good for this type of Metaheuristic algorithm to make a comparison with those methods and algorithms used by other researchers that were mentioned above in the introduction.

Figure 2 shows the values of the objective function.

Table 2. Setting of the MSFL Algorithm.

m	Number of memeplexes	3
n	Number of frogs within memeplexes	NP/m
C_L	Amount of Leaping	1.3

Table 3. Inverse Kinematics Results of MSFL Algorithm

Tests	Population	Iterations		$J(\Theta)$	Total Error	Execution Time [ms]	Reaching Target $(x, y, z, roll, pitch, yaw)$
		iN	iM				
1	20	30	10	11.618	29.71	729	(–15.7365, 5.43, 52.57, 6.63, 12.66, 16.164)
2	30	30	10	7.6614	12.08	1045	(–21.183, 2.915, 50.77, –0.201, 10.01, 14.85)
3	40	30	15	10.5382	19.21	1685	(–25.08, 8.56, 46.818, –6.2, 9.6, 5.4251)
4	40	40	30	18.4625	18.46	4526	(–25.23, 8.34, 47.59, –2.53, 8.62, 14.13)
5	60	40	30	8.2925	8.292	6645	(–24.46, 0.0421, 44.59, 1.65, 11.116, 14.05)
6	80	50	40	11.024	11.02	13540	(–26.998, 3.594, 42.87, –0.068, 9.81, 15.67)
7	100	60	60	29.774	29.77	24191	(–20.03, 30.039, 39.971, –7.71, 3.679, –0.72)
8	130	70	60	0.1511	0.649	46282	(–20.09, 2.99, 40.004, 0.208, 9.64, 14.89)
9	170	60	50	0.6168	2.168	40459	(–20.151, 2.84, 40.09, 0.89, 10.36, 16.15)
10	200	90	40	0.1139	0.298	57362	(–19.927, 3.0072, 39.98, –0.134, 10.105, 14.89)
11	200	100	60	0.0729	0.378	92779	(–20.002, 2.998, 39.99, –0.137, 10.151, 14.92)
12	200	120	80	2.7672	5.8164	150246	(–20.48, 4.153, 40.49, 0.807, 4.087, 13.787)
13	200	200	100	2.6713	1.9339	318481	(–19.27, 2.235, 41.94, –0.979, 10.88, 11.48)
14	250	90	40	0.003	0.016	69818	(–19.99, 3.00023, 39.99, 0.0049, 10.006, 14.99)
15	250	140	80	1.266	6.553	215027	$(-20, 3, 40, -7.01976e-10, 10, 15)$
16	250	140	100	4.647e-9	1.05e-8	260325	(–20.09, 2.97, 40.0008, –1.687, 6.689, 13.57)
17	300	140	80	1.01e-9	3.33e-9	255989	$(-20, 3, 40, 1.16487e-9, 10, 15)$
18	500	90	40	5.49e-10	9.9e-10	136888	$(-20, 3, 40, -1.66261e-11, 10, 15)$
19	500	200	100	3.02e-15	1.56e-14	681646	$(-20, 3, 40, 3.22962e-15, 10, 15)$
20	1000	30	45	0.0968	0.025	95197	(–20.031, 3.04, 40.04, –0.0037, 10.96, 14.876)

The position and orientation of the manipulator's end-effector after applying the solutions to validate the IK solver are presented in Figure 3 and Figure 4, while the joints' positions of the manipulator in Figure 5.

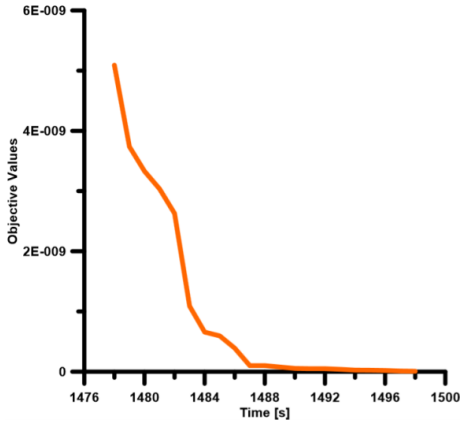

Figure 2. The values of the objective function after applying IK-MSFLA solver

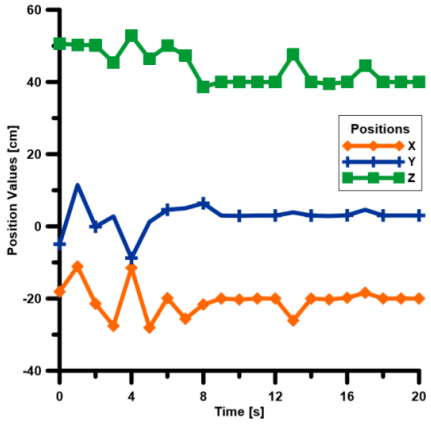

Figure 3. The position of end-effector for the manipulator after applying the solutions to validate IK-MSFLA solver

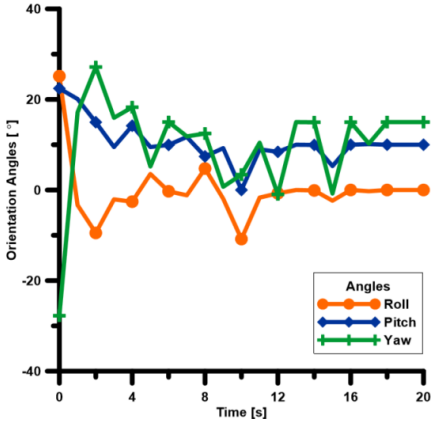

Figure 4. The orientation of the manipulator's end-effector after applying the solutions to validate IK-MSFLA solver

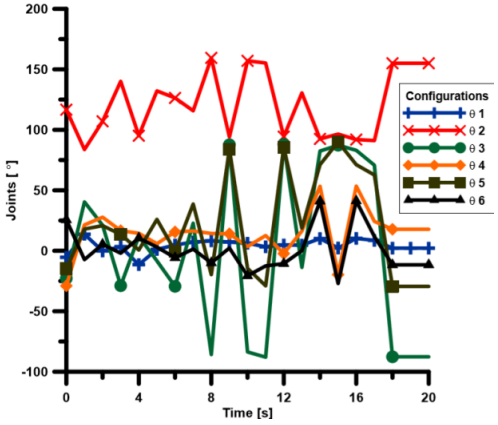

Figure 5. The joints' positions of the manipulator after applying the solutions to validate IK-MSFLA solver

Preliminary tests have been carried out in the laboratory for investigating the performance of the algorithms in addition to analyzing the response, stability, robustness, and smooth motion of the manipulator. The experiments consisted of the execution of various trajectories with the manipulator as shown in Figure 6.

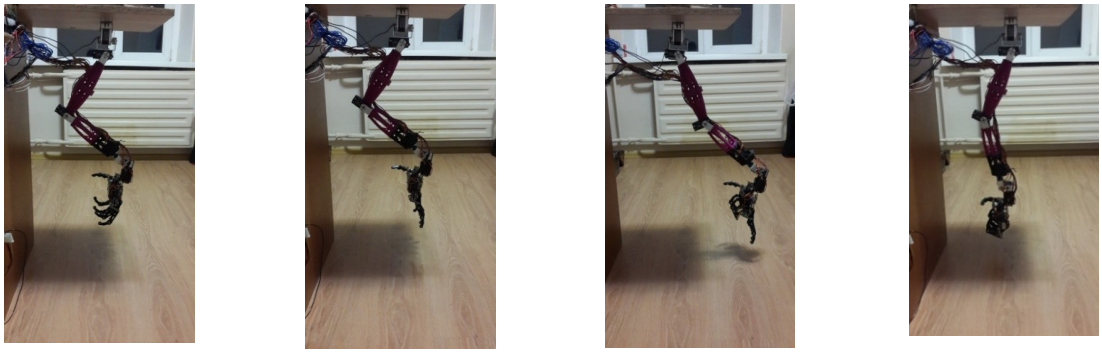

Figure 6. Inverse kinematics tests of a human-like manipulator prototype. It is a lightweight manipulator of 1.1 kg, it has a range of 85 cm in the workspace and a maximal payload of 0.2 kg. The length of the robot in the stretching state is 1.04 m

Figure 7 presents the real-time data from the manipulator plus the platform.

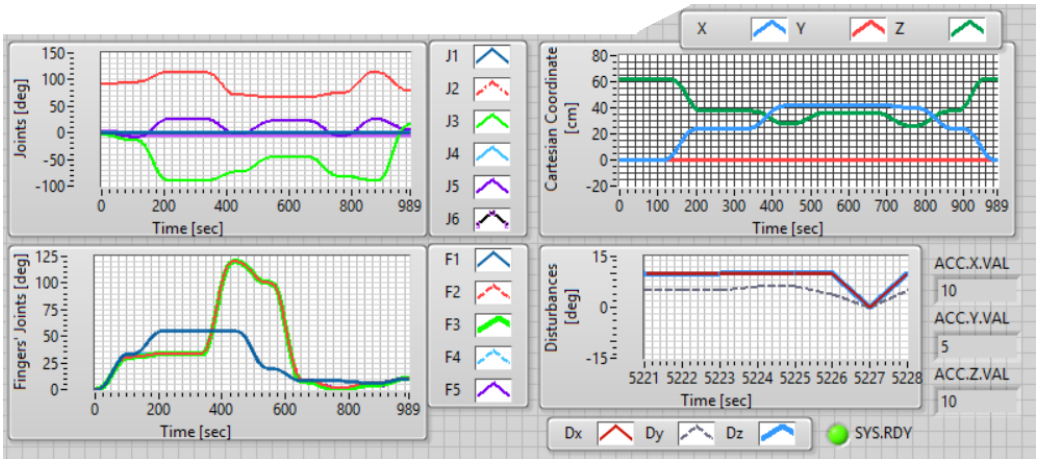

Figure 7. Real-time measurements from the manipulator and the platform, there are configurations of the joints' manipulator, coordinates of the hand, configurations of the fingers and finally the disturbances

6. 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 modified shuffled frog-leaping 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 get a 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 confirmed the feasibility and effectiveness of the suggested method.

This research is funded by Kalashnikov Izhevsk State Technical University grant 15.06.01/18AAH.

References

1. Buss, S. R. (2004). Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. IEEE Journal of Robotics and Automation, 17(1-19), 16.
2. Dulęba, I., & Opałka, M. (2013). A comparison of Jacobian-based methods of inverse kinematics for serial robot manipulators. International Journal of Applied Mathematics and Computer Science, 23(2), 373-382.
3. Wang, X., Zhang, D., & Zhao, C. (2017). The inverse kinematics of a 7R 6-degree-of-freedom robot with non-spherical wrist. Advances in Mechanical Engineering, 9(8), 1687814017714985.
4. Ananthanarayanan, H., & Ordóñez, R. (2015). Real-time Inverse Kinematics of (2n+ 1) DOF hyper-redundant manipulator arm via a combined numerical and analytical approach. Mechanism and Machine Theory, 91, 209-226.
5. Tolani, D., & Badler, N. I. (1996). Real-time inverse kinematics of the human arm. Presence: Teleoperators & Virtual Environments, 5(4), 393-401.

6. Toshani, H., & Farrokhi, M. (2014). Real-time inverse kinematics of redundant manipulators using neural networks and quadratic programming: a Lyapunov-based approach. Robotics and Autonomous Systems, 62(6), 766-781.
7. Reiter, A., Müller, A., & Gattringer, H. (2016, October). Inverse kinematics in minimum-time trajectory planning for kinematically redundant manipulators. In Industrial Electronics Society, IECON 2016-42nd Annual Conference of the IEEE(pp. 6873-6878). IEEE.
8. Geitle, M. (2017). Improving differential evolution using inductive programming (Master's thesis).
9. Bodily, D. M., Allen, T. F., & Killpack, M. D. (2017, May). Motion planning for mobile robots using inverse kinematics branching. In Robotics and Automation (ICRA), 2017 IEEE International Conference on (pp. 5043-5050). IEEE.
10. Ayyıldız, M., & Çetinkaya, K. (2016). Comparison of four different heuristic optimization algorithms for the inverse kinematics solution of a real 4-DOF serial robot manipulator. Neural Computing and Applications, 27(4), 825-836.
11. Rokbani, N., & Alimi, A. M. (2013). Inverse kinematics using particle swarm optimization, a statistical analysis. Procedia Engineering, 64, 1602-1611.
12. Collinsm, T. J., & Shen, W. M. (2017, April). Particle swarm optimization for high-DOF inverse kinematics. In Control, Automation and Robotics (ICCAR), 2017 3rd International Conference on (pp. 1-6). IEEE.
13. Mao, B., Xie, Z., Wang, Y., Handroos, H., Wu, H., & Shi, S. (2017). A hybrid differential evolution and particle swarm optimization algorithm for numerical kinematics solution of remote maintenance manipulators. Fusion Engineering and Design, 124, 587-590.
14. Kachitvichyanukul, V. (2012). Comparison of three evolutionary algorithms: GA, PSO, and DE. Industrial Engineering and Management Systems, 11(3), 215-223.
15. López-Franco, C., Hernández-Barragán, J., Alanis, A. Y., Arana-Daniel, N., & López-Franco, M. (2018). Inverse kinematics of mobile manipulators based on differential evolution. International Journal of Advanced Robotic Systems, 15(1), 1729881417752738.
16. Shiakolas, P. S., Koladiya, D., & Kebrle, J. (2005). On the optimum synthesis of six-bar linkages using differential evolution and the geometric centroid of precision positions technique. Mechanism and Machine Theory, 40(3), 319-335.
17. Juang, C. F., Chen, Y. H., & Jhan, Y. H. (2015). Wall-following control of a hexapod robot using a data-driven fuzzy controller learned through differential evolution. IEEE Transactions on Industrial electronics, 62(1), 611-619.
18. Pierezan, J., Freire, R. Z., Weihmann, L., Reynoso-Meza, G., & dos Santos Coelho, L. (2017). Static force capability optimization of humanoids robots based on modified self-adaptive differential evolution. Computers & Operations Research, 84, 205-215.
19. Ngoc Son, N., Anh, H. P. H., & Thanh Nam, N. (2016). 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, 14(1), 1729881416677695.
20. Wang, M., Luo, J., Fang, J., & Yuan, J. (2018). Optimal Trajectory Planning of Free-Floating Space Manipulator Using Differential Evolution Algorithm. Advances in Space Research.
21. Eusuff, M., Lansey, K., & Pasha, F. (2006). Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization. Engineering optimization, 38(2), 129-154.
22. Li, X., Luo, J., Chen, M. R., & Wang, N. (2012). An improved shuffled frog-leaping algorithm with extremal optimisation for continuous optimisation. Information Sciences, 192, 143-151.
23. Samuel, G. G., & Rajan, C. C. A. (2014). A modified shuffled frog-leaping algorithm for long-term generation maintenance scheduling. In Proceedings of the Third International Conference on Soft Computing for Problem Solving (pp. 11-24). Springer, New Delhi.
24. Afzalan, E., Taghikhani, M. A., & Sedighizadeh, M. (2012). Optimal placement and sizing of DG in radial distribution networks using SFLA. International Journal of Energy Engineering, 2(3), 73-77.
25. Ibrahim I.N. (2018). Ultra Light-Weight Robotic Manipulator. Bulletin of Kalashnikov ISTU. - 2018. - Vol. 21. - N. 1 - P. 12-18. doi: 10.22213/2413-1172-2018-1-12-18 (in Russian).

26. Ibrahim, I. N., & Al Akkad, M. A. (2017). Studying the Disturbances of Robotic Arm Movement in Space Using the Compound-Pendulum Method. Bulletin of Kalashnikov ISTU, 20(2), 156-159.
27. Ibrahim, I. N., Al Akkad, M. A., & Abramov, I. V. (2017, June). Attitude and altitude stabilization of a microcopter equipped with a robotic arm. In Control and Communications (SIBCON), 2017 International Siberian Conference on (pp. 1-8). IEEE.
28. Ibrahim, I. N., Al Akkad, M. A., & Abramov, I. V. (2018, May). Exploring Ackermann and LQR stability control of stochastic state-space model of hexacopter equipped with robotic arm. In Journal of Physics: Conference Series (Vol. 1015, No. 3, p. 032160). IOP Publishing.
29. Ibrahim, I. N., Akkad, M. A. A., & Abramov, I. V. (2018, March). UAV efficient PID and LQR controllers design based on its stochastic state space dynamics model including disturbances. In Electronic and Networking Technologies (MWENT), 2018 Moscow Workshop on (pp. 1-9). IEEE.