An Inverse Kinematics Demonstration of a Custom Robot using C# and CoppeliaSim

Sudip Chakraborty[1] & P. S. Aithal[2]
[1]Post-Doctoral Researcher, College of Computer science and Information science, Srinivas University, Mangalore-575 001, India
OrcidID: 0000-0002-1088-663X; E-mail: sudip.pdf@srinivasuniversity.edu.in
[2]Vice-Chancellor, Srinivas University, Mangalore, India
OrcidID: 0000-0002-4691-8736; E-Mail: psaithal@gmail.com

Area of the Paper: Computer Science.
Type of the Paper: Simulation-based Research.
Type of Review: Peer Reviewed as per |C|O|P|E| guidance.
Indexed In: OpenAIRE.
DOI: http://doi.org/10.5281/zenodo.4755778
Google Scholar Citation: IJCSBE

How to Cite this Paper:
Chakraborty, Sudip, & Aithal, P. S., (2021). An Inverse Kinematics Demonstration of a Custom Robot using C# and CoppeliaSim. *International Journal of Case Studies in Business, IT, and Education (IJCSBE)*, *5*(1), 78-87. DOI: http://doi.org/10.5281/zenodo.4755778.

International Journal of Case Studies in Business, IT and Education (IJCSBE)
A Refereed International Journal of Srinivas University, India.

Crossref DOI : https://doi.org/10.47992/IJCSBE.2581.6942.0102

An Inverse Kinematics Demonstration of a Custom Robot using C# and CoppeliaSim

Sudip Chakraborty[1] & P. S. Aithal[2]
[1]Post-Doctoral Researcher, College of Computer science and Information science, Srinivas University, Mangalore-575 001, India
OrcidID: 0000-0002-1088-663X; E-mail: sudip.pdf@srinivasuniversity.edu.in
[2]Vice-Chancellor, Srinivas University, Mangalore, India
OrcidID: 0000-0002-4691-8736; E-Mail: psaithal@gmail.com

ABSTRACT

Purpose: Inverse Kinematics (I.K.) is not as easy as Forward kinematics (F.K.), where we get a definite result. I.K. algorithm provides several possible solutions. From those finding the best solution is such a critical task. For standard robots which are commercially available in the market, the user is not concerned about I.K.'s complexity. They provide the control board and programming IDE to make it easy. However, when we develop a robotic arm from our D.H. parameter and driver board, complexity arises due to lots of difficulties for executing and successful completion. To make life easy, keeping CoppeliaSim background can eliminate the calculation overhead and get good results. The custom robot is running with less computation power. It may be a good approach. We are using C# for User Interaction. Following step by step, anyone can create a robust I.K. engine with little effort. The complete code is available in GitHub to test and experiment further.

Design/Methodology/Approach: The data are propagated through Interprocess communication. For the user interaction, we use visual studio IDE using the most accessible language, C#. The user interaction data are sent to another application, CoppeliaSim, which calculates inverse kinematics, and effective results are displayed through robotic arm movement.

Findings/Result: Implementing this procedure can get the excellent result of the robotics arm. Furthermore, by imposing the Value on the real robot, we can get effective results. It minimizes the research overhead on I.K. calculation.

Originality/Value: Without knowing I.K. calculation complexity, receiving the Value, we can apply it to the real robot. Two issues we can solve here. One is the calculation, and another one is experiment overhead.

Paper Type: Simulation-based Research.

Keywords: Inverse Kinematics, I.K., 7DF Robot, I.K. Calculation, CoppeliaSim with C#, C# socket communication, CoppeliaSim Socket Programming.

1. INTRODUCTION :

Inverse Kinematics is a challenging task for every robot researcher engaged with a serial manipulator or robotics arm. There are no methods available where we can get better or perfect results. It takes lots of computation overhead, which requires a high computation capability system, which is too much costly on project budgets. Unlike forward kinematics, I.K. (Inverse kinematics) has no definite solution. From the multiple results, the best path selection sometimes tricky. For robotics arm research, simulation helps us to see the outcome. To get the best result and observing the development, we will use the CoppeliaSim Robot simulator. It is free and Open-source. So, anyone can start an experiment just by downloading them from their website. We create a custom 7df robotic arm [1], then we configure for I.K. execution by setting all joints as an inverse kinematics model and joints rotation limitation.
Then starts the simulation. From our c# application, we move the target location and get the joints value. Which can be feed to the real robot. We observe in the simulator, the robot reaches the target or not. If the target is different from the tip position, get the message. We can use a robot driver with a low footprint CPU because primary processing of I.K. calculation by the robot simulator.

2. RELATED WORKS :

Sudip Chakraborty et al. in their paper demonstrate how to create a Custom Robotic A.R.M. in CoppeliaSim" [2]. Deepak Tolani et al. developed a set of inverse kinematics algorithms that is basically for an anthropomorphic arm [3]. Dinesh Manocha and John F. Canny present an algorithm and implementation for efficient inverse kinematics for a general *6R* manipulator [4]. Li-Chun et al. proposed a method based on a combination of two nonlinear programming techniques [5]. ANDREW A. et al., in their paper, shown a method on a new rapidly convergent constrained nonlinear optimization algorithm that uses a modified Newton-Raphson technique [6]. Gaurav et al., in their paper, focus on two established procedures, the pseudo inverse with explicit optimization and the extended Jacobian method [7]. Samuel R. et al., in their paper, introduced two ways for the inverse kinematics of multi bodies with multiple end effectors. The second method used damped least-squares called selectively damped [8]. Paolo Baerlocher et al. present an efficient recursive algorithm [9]. Daniel R. et al. have been proposed that high joint speeds could be avoided by introducing redundant joints and using an appropriate kinematic inversion algorithm [10]. Sreenivas Tejomurtula presents a solution based on structured neural networks that can be trained quickly. The proposed method yields multiple and precise answers, and it is suitable for real-time applications [11]. Aristidou et al. in their paper, suggest that I.K.'s family of solvers is best suited for particular problems [12].

Andreas Aristidou et al. implement FABRIK, a fast, iterative solver for the Inverse Kinematics problem. It has a low computational cost and produces visually realistic poses. Constraints can easily be incorporated within multiple chains with multiple end effectors are also supported [13]. Serdar Kiiqiik et al. approach closed-form solution of robot inverse kinematics problem [14]. R. M¨uller-Cajar1 et al. introduce a new method utilizing the law of cosines of a kinematic chain when given a target. their method incurs computational and rotational costs than Cyclic-Coordinate Descent (C.C.D) [15]. Manfred L. Husty et al. introduces a very efficient algorithm to compute the inverse kinematics of a general 6R serial kinematic chain. The main idea is to use classical multidimensional geometry to structure the problem [16]. C.S.G. Lee et al. Introduces I.K. of PUMA robot calculated in two stages. The first three joints are calculated by looking at the projection of the position vector. The last three joints are solved using the calculated joint solution from the first three joints [17]. Joey K. Parker proposed a genetic algorithm that allows additional constraints to be specified easily. The GA is applied to a test problem in which the maximum joint displacement in a point-to-point positioning task is minimized [18]. Shaher Momani et al., in their paper, the solution of inverse kinematics problem of robot manipulators using genetic algorithms (G.A.) [19]. Tarık C et al. experiment with robotic manipulator by using cubic trajectory planning. [20]. Patrick Beeson et al. provides quantitative comparisons, using multiple humanoid platforms, between an improved implementation of the K.D.L. inverse Jacobian algorithm [21]. PYUNG H. CHANG et al. proposed a method that same equilibrium state for the resolved-motion method. The method is demonstrated to give more accurate trajectories than the resolved-motion method [22].

Stefan Chiaverini et al., in their paper, present experimental results on the implementation of the damped least-squares method for the six-joint ABB IRb2000 industrial robot manipulator. Several inverse kinematics schemes are reviewed, which allow robot control through kinematic singularities [23]. Dominik et al. have an excellent approach to robot experiments using Rapidly-exploring Random Trees (RRTs) [24]. Bekir Karlik et al., in their work, implement the kinematics of a six-degrees-of-freedom robot manipulator using ANN [25]. De Xu et al., in their report, propose an analytical solution for a 5-DOF manipulator [26]. Ahmed El-Sherbiny et al., in their article, presents a comparative study between different soft computing-based methods applied to the problem of inverse kinematics. [27]. John Q. Gan et al. paper presents the first complete analytical solution to the inverse kinematics of the P2Arm [28]. Jingguo Wang1et. Al, in their paper, A seven-DOF redundant manipulator is designed to do the computer simulations, and the natural experiments are carried out on a Power cube modular manipulator. Their results demonstrated the effectiveness of the proposed algorithm [29]. V.N. Iliukhin et al. carried out research aimed at creating a robotic manipulator controlled employing Brain-Computer Interface for improving household self-reliance of persons with disabilities and expanding the scope of their activity [30]. Michal Kelemen et al. a new algorithm for the control of kinematically-redundant

manipulator considering three secondary tasks, namely a joint limit avoidance task, a kinematic singularities avoidance task, and an obstacle avoidance task [31].

Studying the above-related novel research, we realize that all need heavy computation power CPU to operate with the real-time operation. For lower footprint, the CPU control board is a bottleneck to execute the task. So, we distribute the calculation overhead to the simulator and our robot can run smoothly.

3. OBJECTIVES :

(1) We solve the I.K. calculation in a unique approach using the CoppeliaSim simulator.
(2) Introduce CoppliaSim simulator as an I.K. solver.
(3) From this Experiment, we can drive the joints motor of the actual robot.
(4) We can easily find digital clone solutions for robot movement.
(5) From this experiment, we can learn how TCP/IP socket communication is established between two applications.

4. APPROACH AND METHODOLOGY :

The complete block diagram of our Experiment is below.

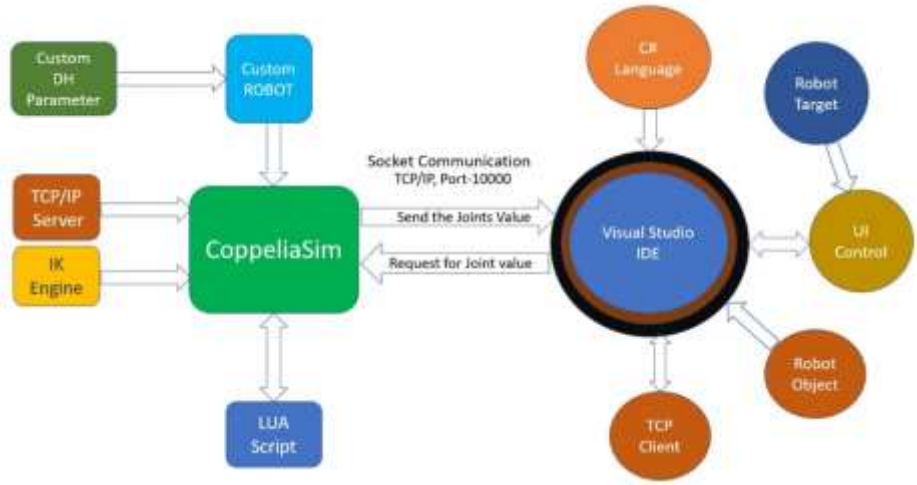

Fig. 1: The block diagram of our Experiment

Here the two applications play a significant role. CoppeliaSim is shown on the left side, and the visual studio is on the right side. The CoppeliaSim constitutes several modules. When open the simulator, the first appear Workspace where we build a robot using our custom D.H. parameter [2]. Then we add the LUA script into the robot base object. We set some configurations to operate the inverse kinematics through the program. Then run the system. It creates a server socket and waits for client requests. It responds upon client request and processes I.K. through the simulator's I.K. Engine. On the other side, for user interaction, we develop a user Interface using C# language. When the C# application runs, it opens a window with some element for user interaction. Changing the target position, we can observe that the robot is following the target. If it happens, our Experiment is successful. We can change the D.H. parameter for further Experiments, and taking the result can drive another robot's joints in real robots. Create the visual studio application, and we take the help of internet tutorials. In recommend section, we can get the link for this project's source code.

5. EXPERIMENT :

Now we can start our Experiment. We need to figure out the D.H. parameter of our custom robot and create a robotic arm [2] in the CoppeliaSim environment. We can go with the below D.H. parameter or create our own.

Table 1: List of D.H. Parameter of Custom Robot

Link	θ	d(meter)	a(meter)	α(Deg.)
1	0.0000	0.0400	0.0000	0.0000
2	0.0000	0.0000	0.0000	90.0000
3	0.0000	0.0000	0.0550	0.0000
4	0.0000	0.0000	0.0400	0.0000
5	0.0000	0.0000	0.0000	0.0000
6	0.0000	0.0270	0.0300	0.0000
7	0.0000	0.0270	-0.0300	-90.0000

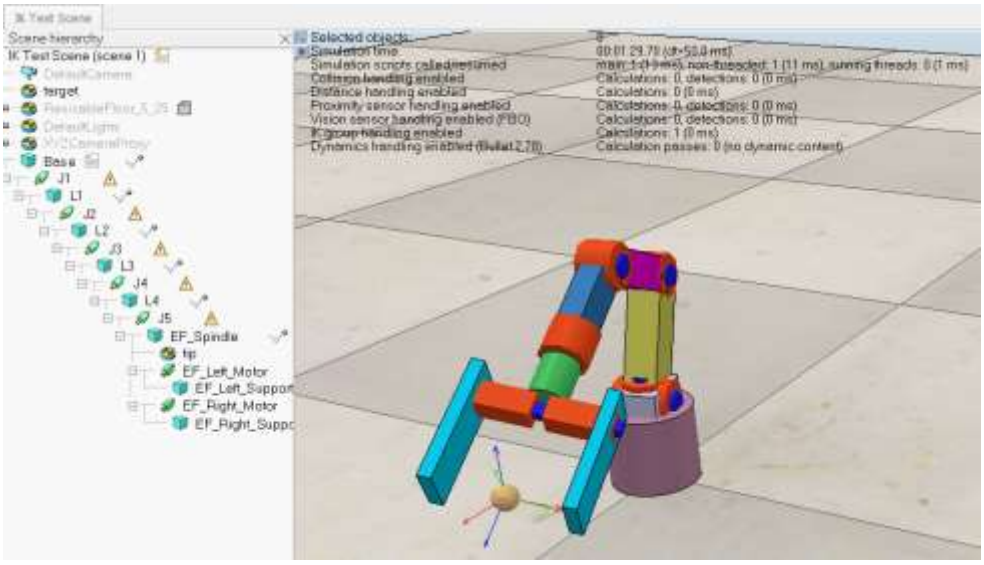

Fig. 2: After designing the robot, it looks like the above figure

Fig. 3: IK group selection

After completing the design, from the side Toolbar, select the "Calculation Module properties" Button (f(x)). The calculation module window will open. Then from the Kinematics Tab, press "Add new I.K. group." it will add one I.K. group under the button. Double click and rename it as "ud." It can be anything. For better understanding, we select the name to understand. "ud" stands for undamped. We have then checked on "**Explicit handling**." which is the most crucial parameter. If uncheck, IDE will

automatically process I.K., and it is not able to control from scripts. Moreover, we cannot get the status of I.K. is failed or not.

Furthermore, under the "Calc. method," select "Pseudo Inverse" for the damped method. Next, we have to press "Edit I.K. elements." From the combo box, select "tip." and press the left side button "Add new I.K. element with the tip." It will create an I.K. element "tip." It is shown in the box, which is under the button. The complete windows look like figure 4. then close the windows and another I.K. group we need to create. It the same way as we see before.

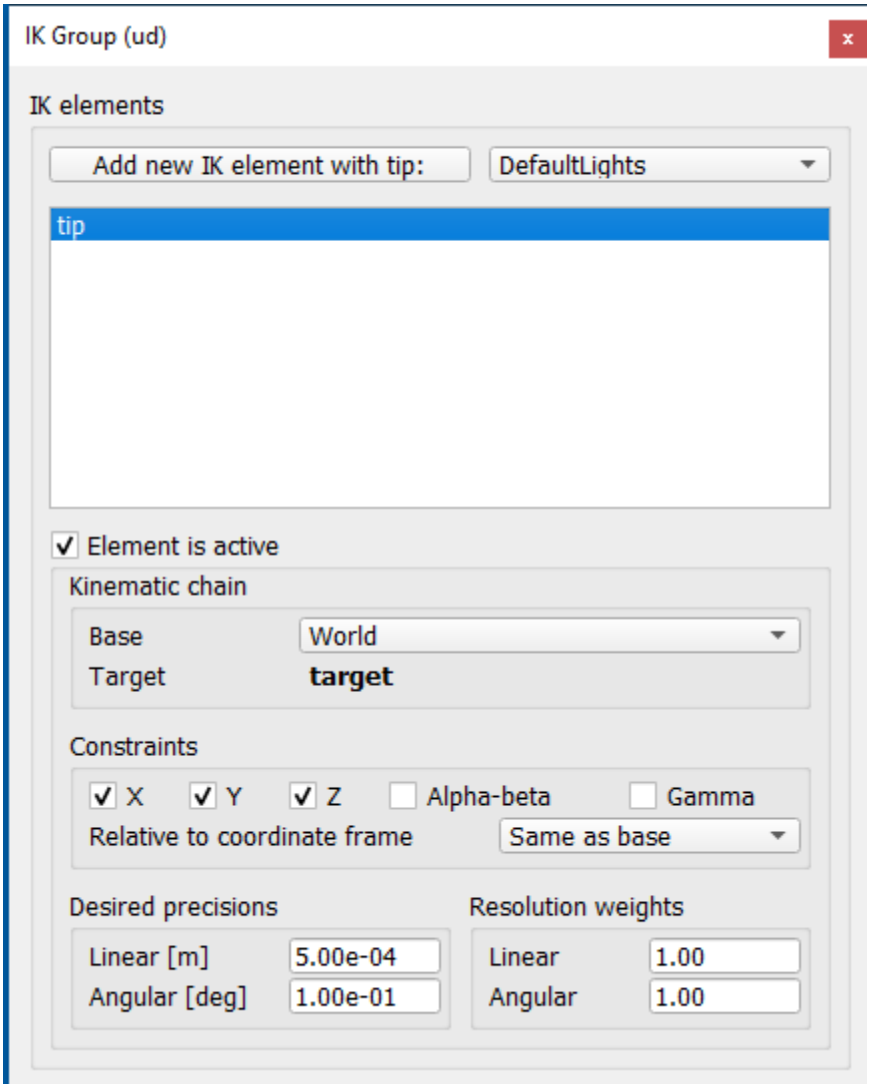

Fig. 4: IK selection window

Sometimes only one single procedure cannot solve the problem and does not reach the given target. So alternate method should try, and if again fails, we can assume that the end effector is unreachable to the given target.

Now we add another I.K. group using D.L.S. methods, i.e., Damp least square method. We press the button" Add new OK group." Remane "dmp" like that. Must check "**Explicit handling**" to operate I.K. from LUA script. Select "D.L.S." from Calc. method. Next, press the Edit I.K. elements button. From the top combo box, select tip and press the left side button "Add new I.K. element with the tip." Close the window. The final selection looks the figure 5.

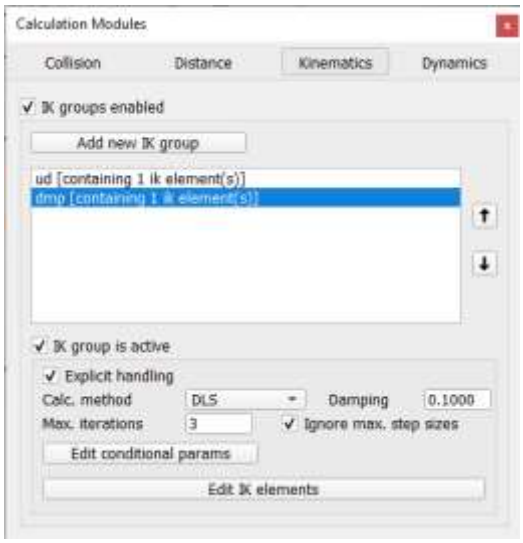

Fig. 5: DLS method selection window

Now we must add some code inside the scripts. See recommendation section for link. After adding the code, if we run the CoppeliaSim pressing the run button, it will not respond. The cause is that after running the simulation, it starts the TCP server using socket 6000. It is a blocking function. It waits for the client's request. The arrival of the client request will back in the main loop and execute the following code. So, we suggest that after adding the code, do not run. We need to do another side as well, i.e., the visual studio part. Open visual studio. Create a new project. We can create this type of user interface or as our requirement. For U.I. creation, we can take help from the internet. After U.I. creation, we can add the code to each U.I. element. Build the code. It should build successfully. The prebuild executable code is available inside the downloaded folder. Now we run the P.C. application, then CoppeliaSim by the run button. If everything is OK, changing the slider, the robot will move. The Open and Close Button is to operate the gripper (figure 6).

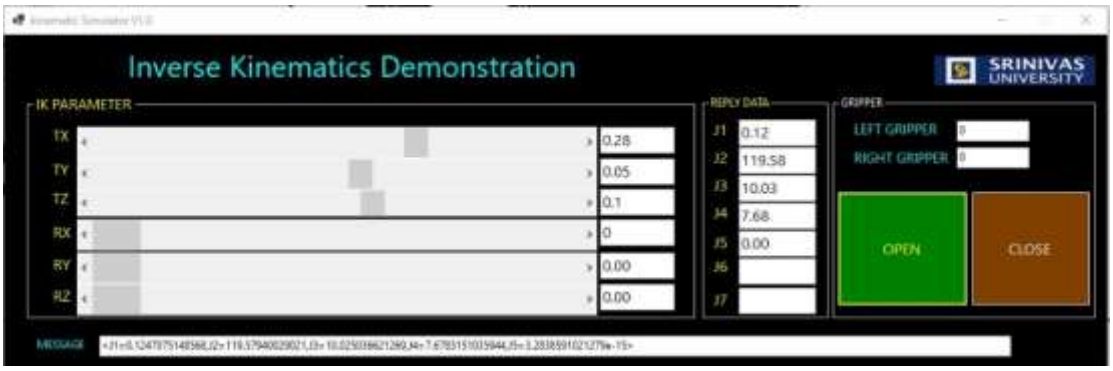

Fig. 6 : Inverse Kinematics Demonstration

The end effector has three orientations. For Translation TX, T.Y. and T.Z. For x-axis movement, change the slider position TX labelled. For Y-axis movement, change T.Y. tagged slider and for Z axis movement, change the T.Z. Slider. Another three sliders for rotation orientation. R.X. Slider for Rotation around X-axis, R.Y. for Y-axis movement, and R.Z. for rotation around the Z-axis. These rotation buttons for the further scope of improvement. Here is not implemented. When we change the target position, the tip of the robot follows the target. Furthermore, for that particular target, we can fetch the joints angle value. Later we can feed the data to the custom robot for inverse kinematics.

6. RECOMMENDATIONS :

- Before experimenting with inverse kinematics, we should experiment with forwarding kinetics. It builds good knowledge on robotics arm movement.

- The link for the complete project can be found https://github.com/sudipchakraborty/An-Inverse-Kinematics-Demonstration-using-C-and-CoppeliaSim
- More research needs to implement into practical implementation.
- Before creating our own, we recommend observing and understand the overall architecture of the Experiment.
- From the joints result fetching by P.C. application from the robot simulator, we can feed to the real robot. Moreover, the robot driver drives the joint angle to calculate Inverse kinematics, which consumes a lot of processing power. Using this procedure, we can be built the driver with a lower footprint microcontroller like Arduino NANO, Arduino U.N.O.
- If we did not get the desired result, we should check the D.H. parameter, given the length and rotation value inside the textbox in different places.
- We can get Target position data from a 3D depth camera or an ultrasonic sensor to drive the robot.

7. CONCLUSION :

I.K. is the essential part of the articulated robotics arm or serial manipulator. Lots of solutions are available in the field. Nevertheless, most of the procedure is proprietary, cannot be accessed by the robot researcher. Using the CoppeliaSim robot simulator, we can solve our inverse kinematics issue. Following the described procedure, we can easily make our custom robot and move on to the given target. The target can drive by a 3D Depth camera or sonar sensor. With some more research, we can implement it into practical industrial applications. It will distribute the computation overhead to the High resourced object. Furthermore, we can use a low power board with a low-footprint CPU for the motor driver, which drives the robot.

REFERENCES :

[1] Chakraborty, Sudip, & Aithal, P. S., (2021). Forward Kinematics Demonstration of 6DF Robot using CoppeliaSim and C#. *International Journal of Applied Engineering and Management Letters (IJAEML)*, *5*(1), 29-37. DOI: http://doi.org/10.5281/zenodo.4680570.

[2] Chakraborty, Sudip, & Aithal, P. S., (2021). A Custom Robotic A.R.M. in CoppeliaSim. *International Journal of Applied Engineering and Management Letters (IJAEML)*, *5*(1), 38-50. DOI: http://doi.org/10.5281/zenodo.4700297.

[3] Deepak Tolani, Ambarish Goswami, Norman I. Badler, (2000). Real-Time Inverse Kinematics Techniques for Anthropomorphic Limbs. *Graphical Models*, *62*(5), 353-388. DOI: https://doi.org/10.1006/gmod.2000.0528.

[4] Manocha, D. and Canny, J. F. (1994). Efficient inverse kinematics for general 6R manipulators. *IEEE Transactions on Robotics and Automation*, *10*(5), 648-657, DOI: 10.1109/70.326569.

[5] Wang, L. T. and Chen, C. C. (1991). A combined optimization method for solving the inverse kinematics problems of mechanical manipulators. *IEEE Transactions on Robotics and Automation*, 7(4), 489-499, DOI: 10.1109/70.86079.

[6] Goldenberg, A. Benhabib, B. and Fenton, R. (1985). A complete generalized solution to the inverse kinematics of robots, *IEEE Journal on Robotics and Automation*, *1*(1), 14-20, DOI: 10.1109/JRA.1985.1086995.

[7] Tevatia, G. and Schaal, S. (2000). Inverse kinematics for humanoid robots. Proceedings 2000 ICRA. Millennium Conference. *IEEE International Conference on Robotics and Automation*. Symposia Proceedings (Cat. No.00CH37065), pp. 294-299, Vol.1, DOI: 10.1109/ROBOT.2000.844073.

[8] Samuel R. Buss and Jin-su Kim (2011). Selectively Damped Least Squares for Inverse Kinematics, *University of California, San Diego*, Pages 37-49, DOI: 10.1080/2151237X.2005.10129202

[9] Baerlocher, P., Boulic, R. (2004). An inverse kinematics architecture enforcing an arbitrary number of strict priority levels. *Vis Comput.*, *20*(1), 402–417. https://doi.org/10.1007/s00371-004-0244-4.

[10] Baker, D. R., Wampler, C. W. (1988). On the Inverse Kinematics of Redundant Manipulators. *The International Journal of Robotics Research*, *7*(2), 3-21. DOI:10.1177/027836498800700201.

[11] Sreenivas Tejomurtula, Subhash Kak (1999). Inverse kinematics in robotics using neural networks, *Information Sciences*, 116(2–4), 147-164, DOI: https://doi.org/10.1016/S0020-0255(98)10098-1.

[12] Aristidou, A. Lasenby, J. Chrysanthou, Y. Shamir, A. (2017). Inverse Kinematics Techniques in Computer Graphics: A Survey. *COMPUTER GRAPHICS forum*, pp. 1–24. DOI: https://doi.org/10.1111/cgf.13310

[13] Andreas Aristidou, Joan Lasenby, (2011). FABRIK: A fast, iterative solver for the Inverse Kinematics problem. *Graphical Models*, *73*(5), 243-260, DOI: https://doi.org/10.1016/j.gmod.2011.05.003.

[14] Kucuk, S. and Bingul, Z. (2004). The inverse kinematics solutions of industrial robot manipulators. *Proceedings of the IEEE International Conference on Mechatronics*, I.C.M. '04., 2004, pp. 274-279, DOI: 10.1109/ICMECH.2004.1364451.

[15] Muller-Cajar, R., Mukundan, R. (2007). Triangulation - A New Algorithm for Inverse Kinematics. Hamilton, New Zealand: Image and Vision Computing New Zealand (IVCNZ) 2007 Conference, 5-7 Dec 2007. *Proceedings of Image and Vision Computing New Zealand*, 181-186.

[16] Manfred L. Husty, Martin Pfurner, Hans-Peter Schröcker, (2007). A new and efficient algorithm for the inverse kinematics of a general serial 6R manipulator. *Mechanism and Machine Theory*, *42*(1), 66-81, DOI: https://doi.org/10.1016/j.mechmachtheory.2006.02.001.

[17] C. s. g. Lee and M. Ziegler, (1984). Geometric Approach in Solving Inverse Kinematics of PUMA Robots. *IEEE Transactions on Aerospace and Electronic Systems*, *20*(6), 695-706, DOI: 10.1109/TAES.1984.310452.

[18] Parker, J. Khoogar, A., and D. Goldberg, (1989). Inverse kinematics of redundant robots using genetic algorithms. *IEEE International Conference on Robotics and Automation*, Scottsdale, AZ, pp. 271-276. DOI: 10.1109/ROBOT.1989.100000.

[19] Shaher Momani, Zaer S. Abo-Hammour, and Othman M. K. Alsmadi (2016). Solution of Inverse Kinematics Problem using Genetic Algorithms. *Applied Mathematics & Information Sciences*, *10*(1), 1-9. DOI: 10.12785/amis/Solution of inverse kinematics problem.

[20] Raşit Köker, Cemil Öz, Tarık Çakar, Hüseyin Ekiz (2004). A study of neural network-based inverse kinematics solution for a three-joint robot. *Robotics and Autonomous Systems*, *49*(3–4), 227-234, DOI: https://doi.org/10.1016/j.robot.2004.09.010.

[21] Beeson, P. and Ames, B. (2015). TRAC-IK: An open-source library for improved solving of generic inverse kinematics. IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids), 2015, pp. 928-935, DOI: 10.1109/HUMANOIDS.2015.7363472.

[22] Pyung Chang, (1987). A closed-form solution for inverse kinematics of robot manipulators with redundancy. *IEEE Journal on Robotics and Automation*, *3*(5), 393-403, DOI: 10.1109/JRA.1987.1087114.

[23] Chiaverini, S. Siciliano, B. and Egeland, O. (1994). Review of the damped least-squares inverse kinematics with experiments on an industrial robot manipulator. *IEEE Transactions on Control Systems Technology*, *2*(2), 123-134, DOI: 10.1109/87.294335.

[24] Bertram, D., Kuffner, J., Dillmann, R., and Asfour, T. (2006). An integrated approach to inverse kinematics and path planning for redundant manipulators. *Proceedings of IEEE International Conference on Robotics and Automation*, ICRA 2006, pp. 1874-1879, DOI: 10.1109/ROBOT.2006.1641979.

[25] Bekir Karlik, Serkan Aydin (2000). An improved approach to the solution of inverse kinematics problems for robot manipulators. *Engineering Applications of Artificial Intelligence*, *13*(2), 159-164, DOI: https://doi.org/10.1016/S0952-1976(99)00050-0.

[26] Xu, D., Acosta Calderon, C.A., Gan, J.Q. et al. (2005). An analysis of the inverse kinematics for a 5-DOF manipulator. *Int J Automat Comput.*, *2*(1), 114–124, DOI: https://doi.org/10.1007/s11633-005-0114-1.

[27] Ahmed El-Sherbiny, Mostafa A. Elhosseini, Amira Y. Haikal, (2018). A comparative study of soft computing methods to solve inverse kinematics problem. *Ain Shams Engineering Journal*, *9*(4), 2535-2548, DOI: https://doi.org/10.1016/j.asej.2017.08.001.

[28] John Q. Gan, Eimei Oyama, Eric M. Rosales and Huosheng Hu, (2004). A complete analytical solution to the inverse kinematics of the Pioneer 2 robotic arm. *Robotica, 23*(1), 123–129. Cambridge University Press. DOI: 10.1017/S0263574704000529

[29] Wang J, Li Y, Zhao X. (2010). Inverse Kinematics and Control of a 7-DOF Redundant Manipulator Based on the Closed-Loop Algorithm. *International Journal of Advanced Robotic Systems*, *7*(4), 1-10. DOI:10.5772/10495.

[30] Iliukhin, V. N. Mitkovskii, K.B. Bizyanova, D. A. Akopyan, A. A. (2017). The Modeling of Inverse Kinematics for 5 D.O.F. Manipulator. *Procedia Engineering*, *176*(1), 498-505, DOI: https://doi.org/10.1016/j.proeng.2017.02.349.

[31] Kelemen, M., Virgala, I., Lipták, T., Miková, Ľ., Filakovský, F., Bulej, V. A. (2018). Novel Approach for an Inverse Kinematics Solution of a Redundant Manipulator. *Appl. Sci., 8(1)*, 2229. https://doi.org/10.3390/app8112229.

****