Determining Inverse Kinematics Of A Serial Robotic Manipulator Through The Use Of Genetic Programming

Conference Paper · June 2021

CITATION
1

READS
695

7 authors, including:

Zlatan Car
University of Rijeka
251 PUBLICATIONS **2,112** CITATIONS
SEE PROFILE

Sandi Baressi Šegota
Juraj Dobrila University of Pula
127 PUBLICATIONS **1,313** CITATIONS
SEE PROFILE

Nikola Anđelić
University of Rijeka
139 PUBLICATIONS **1,767** CITATIONS
SEE PROFILE

Ivan Lorencin
Juraj Dobrila University of Pula
114 PUBLICATIONS **1,649** CITATIONS
SEE PROFILE

DETERMINING INVERSE KINEMATICS OF A SERIAL ROBOTIC MANIPULATOR THROUGH THE USE OF GENETIC PROGRAMMING

Zlatan Car[1], Sandi Baressi Šegota[1], Nikola Anđelić[1], Ivan Lorencin[1], Jelena Musulin[1], Daniel Štifanić[1], Vedran Mrzljak[1]

[1] Faculty of Engineering
University of Rijeka, Vukovarska 58, 51000 Rijeka, Croatia
e-mail: car@riteh.hr, sbaressisegota@riteh.hr, nandelic@riteh.hr, ilorencin@riteh.hr, jmusulin@riteh.hr, dstifanic@riteh.hr, vmrzljak@riteh.hr

Abstract:

Inverse kinematics is one of the key parts of any industrial robotic manipulator modeling. Solving the inverse kinematics of a robotic manipulator in the classical analytical manner is fairly complex and error-prone. While previous research has shown the possibility of AI application for inverse kinematics solutions, such models have certain pitfalls which makes the probability of their use low. For this reason, the authors propose the utilization of Genetic Programming (GP), which is an AI method that generates models in the shapes of equations. The application of the algorithm in question shows promise, with errors being lower than 0.5 degrees for all regressed joints when evaluated using Mean Absolute Error (MAE). This points towards the fact that GP could be utilized in such an approach – but some details of the algorithm need to be addressed, such as the tendency to generate large and simplifiable equations, or lower precision when compared to previous AI-based solutions to the same problem.

Key words: genetic programming, industrial robotics, inverse kinematics

1. Introduction

One of the basic tasks in the modeling of an industrial robotic manipulator is the determination of inverse kinematic transformations. These transformations allow for transformation between the end-effector coordinates in the global workspace coordinate system (x, y, and z coordinates) into the local manipulator coordinate system defined through the joint angle values (q_i). The standard approach to determining these equations is to solve a set of equations, defined through the direct kinematics, which consists of n equations, where n is the number of manipulator's joints. For more complex robotic manipulators this is a complex task, with a high probability of an error occurring [1].

Artificial intelligence (AI) algorithms have shown capabilities to achieve simple solutions to complex issues, by creating models through the process of Machine Learning (ML). ML allows for models to be generated using existing data. This is accomplished through the iterative adjustment of internal parameters of the model based on the current model error for a given data point. AI and ML applications have been used to a great effect in the field of robotics, with many applications. Solowjow et al. [2] display the application of deep learning implementation on a

programmable logic controller with the task or robot grasping improvement. Toquica et al. [3] demonstrate an application of AI modeling for the solution to an inverse kinematics problem for a parallel robotic manipulator. Baressi Šegota et al. [4] demonstrate the application of evolutionary algorithms in path planning of single and collaborative serial industrial manipulators. Wei et al. [5] demonstrate the use of deep learning algorithms in the development of end-to-end policies for robotic swarm achieving highly precise and adjusted policies, as well as showing the application of EC algorithms for such complex systems [6] .

Most existing solutions to inverse kinematics using AI-based methods face a model-shape limitation. Namely, the generated models are not language-agnostic. This means that to use the model in question researchers need to be familiar with programming languages and function libraries used for model creation, greatly limiting the applicability of research. The authors attempt to address this issue through the use of the GP algorithm. GP is an algorithm that combines ML and evolutionary computation (EC) approaches. It is based on the generation of mathematical equations, and their evolution through the application of EC operators such as crossover, mutation, and reproduction [7]. Each of the generated solutions is checked against the dataset and then further evolved, with focus given to better solutions. The authors propose that such an approach could generate inverse kinematics solutions which will then be easily implemented in any programming language and most tools that support basic arithmetic and trigonometric functions.

2. Methodology

The methodology will be presented in this section. First, the overview of dataset creation will be given, followed by the description of the used GP methodology (hyperparameter selection and result evaluation).

2.1 Direct kinematics

The used dataset is generated by the authors through the determination of direct kinematics of the robotic manipulator. The selected robotic manipulator is ABB IRB 120 [8], with the diametric view of the manipulator, with rotational axes marked, being given in Figure 1.

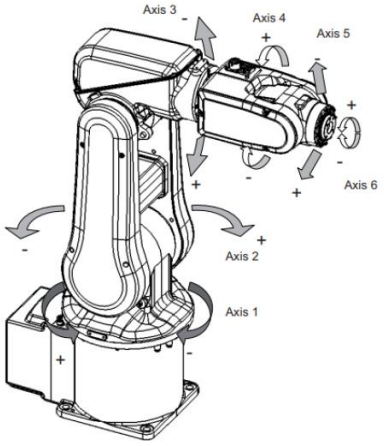


Fig. 1. Diametric view of the used robotic manipulator [8].

The first step is the determination of the direct kinematic equations for the manipulator in question. This is accomplished using the Denavit-Hartenberg (DH) method. DH is an iterative method that places coordinate systems on the bases of individual robotic manipulator joints.

Then, based on the interactions between the coordinate systems, the parameters of connections are determined. These parameters are then placed into transformation matrices for the individual joint, with the transformation matrix of the entire manipulator being obtained as a product of the individual transformation matrices as

$$T_0^6 = \prod_{i=1}^{6} T_{i-1}^i. \tag{1}$$

To determine the individual matrix parameters, the D-H schematic, depicting the systems needs to be drawn. This schematic is shown in Figure 2.

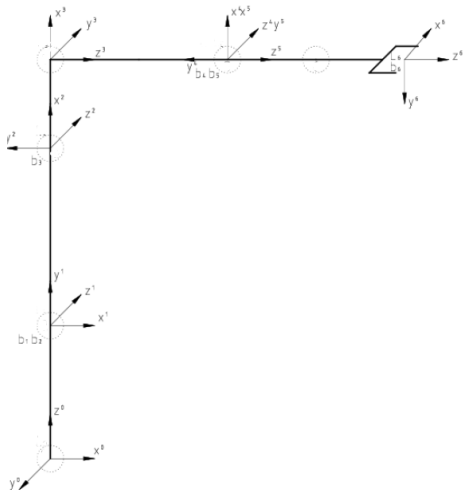


Fig. 2. Simplified schematic of a robot manipulator with DH method determined coordinate systems

After the schematic is drawn and the transformation matrix has appropriate values inserted into it, elements of the matrix depicting the end-effector positioning and orientation can be extracted from the matrix in question. These equations allow us to determine the *x*, *y*, and *z* position in the tool space, as well as the orientation angles φ, θ, and ψ. These allow us to determine the exact position and tool orientation of the end-effector for a given set of joint angles

$$Q = [q_1, q_2, ..., q_n], \tag{2}$$

where *n* equals 6 in the presented case. Such an approach can allow us for dataset generation. The values of kinematic parameters determined in this manner are given in Table 1.

Joint (k)	1	2	3	4	5	6
θ_k	q_1	q_2	q_3	q_4	q_5	q_6
a_k	0	l_2	l_3	0	0	0
d_k	l_1	0	0	l_4	0	l_6
α_k	-π/2	0	-π/2	π/2	-π/2	0

Table 1. The determined DH kinematic parameters, with $q_{1\text{-}6}$ being variables, and $l_{1\text{-}6}$ being manipulator dimension available in [8].

2.2. Dataset generation

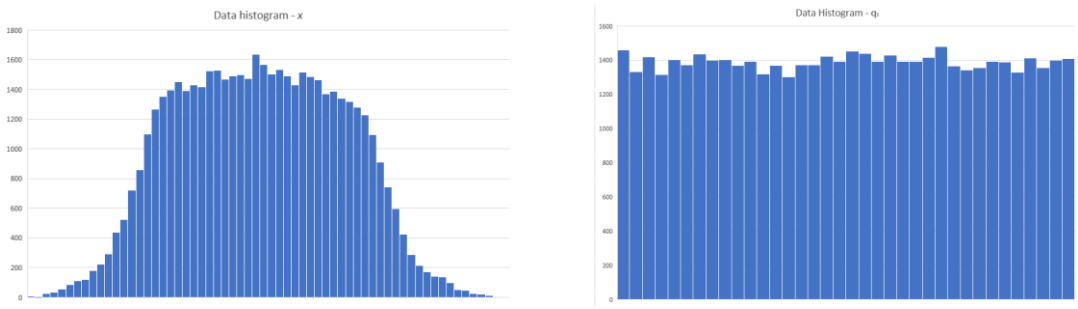
The dataset is generated through the use of the aforementioned equations. The upper and

lower possible values are set according to manufacturer-defined values [8] and given in Table 2.

Joint	Lower Limit [°]	Upper Limit [°]	Range [°]
1	-165	165	330
2	-110	110	220
3	-70	110	180
4	-160	160	320
5	-120	120	240
6	0	360	360

Table 2. The joint limits used in the generation of the used dataset

Then, values for each joint are generated uniformly randomly within those limits. Once the vector of joint angles Q according to Equation 2 is generated, the obtained DH equations are used to generate the tool space coordinates. This approach generates uniformly distributed values which will be used as outputs in the GP modeling (angles), and normally distributed values which will be used as inputs – which are ideal distributions for the ML algorithm application [9]. Examples of the distributions are given as histograms in Figure 3. For brevity, only the tool space coordinate x, and angle joint q_1 are given.



(a) (b)

Fig. 3. An example of achieved distributions for: (a) coordinate x and (b) coordinate q_1

2.3. Genetic Programming

To apply GP on the generated dataset the hyperparameter selection needs to be performed. Hyperparameters are the values that influence the behavior of the algorithm. In the presented work the Grid Search (GS) hyperparameter space search has been performed. To use the GS the hyperparameter space needs to be defined. Hyperparameter space is an n-dimensional space containing all possible values of hyperparameters (one for each dimension). To apply GS, discretization is performed of this space to limit the possible hyperparameter values. The searched values in the paper are population size which determines the number of candidate equations tested and used in evolution, generations which determine the number of GP population iterations, crossover rate which defines the probability of crossover operation being applied, and maximum selection sample size which determines the number of candidate solutions used within selections for EC operations. The possible values of these hyperparameters are given in Table 3.

Hyperparameter	Tested Values
Population size	20, 50, 100, 200, 500, 1000, 2000
Generations	50, 100, 500, 1000, 2000
Crossover rate	0.7, 0.8, 0.9, 0.95
Selection Sample Size [%]	10, 25, 50, 75, 100

Table 3. Tested hyperparameter values.

As all possible combinations are tested, it can be concluded (from the above table) that the number of tested hyperparameter combinations is the product of all individual hyperparameter counts – in this case, 700. This means that 700 different models are evolved and evaluated – for each joint. As GP can only regress a single value at the time 6 separate models, one for each joint need to be trained. It must be noted that other crucial parameters have not been modified in the GS procedure. These hyperparameters either self-adjust (such as various mutation and reproduction coefficients) based on the set values of the crossover coefficient, are automatic (such as parsimony coefficient) or are set to constant (such as early stopping value, set at 0.1) [10]. An additional hyperparameter used is the operation set, which defines which mathematical operations may be used in the modeling process – which was limited to basic arithmetic and basic trigonometric operations.

The evaluation of candidate solutions (models) is performed using MAE, given by the equation [11]:

$$MAE = \sum_{i=0}^{n} |y_i - \hat{y}_i|, \tag{3}$$

Where n is the total number of data points in the dataset y_i is a real value of the output for datapoint i, and $\hat{y}_i$ is the predicted value for the same data point.

3. Results and Discussion

The results are presented in Table 4. MAE for each of the outputs ($q_{1\text{-}6}$) is given in the Table, along with the vector of hyperparameters used to generate the solution which achieved the given error value. Only the best solutions were selected.

Joint	MAE [°]	Population Size	Generations	Crossover Rate	Sample Size
q_1	0.32	1000	50	0.9	10
q_2	0.23	2000	100	0.9	75
q_3	0.41	2000	100	0.95	50
q_4	0.48	2000	50	0.9	100
q_5	0.13	500	50	0.95	10
q_6	0.34	1000	200	0.9	50

Table 4. Results of best achieved models per joint, with model hyperparameter values.

It can be noted that all the errors are below 0.5 degrees, which indicates a successful regression. Observing the utilized hyperparameters it can be noted that population size trended towards the larger end of the possible values, while the opposite is true with the generations. The crossover was also high, indicating a low tendency to randomize the solution pool via the mutation operations. No noticeable trend can be discerned for Sample Size. The resulting models (equations) are too large to be presented in the manuscript but can be generated using the provided hyperparameters and methodology.

4. Conclusions

It can be concluded that GP can be used to successfully regress the models for inverse kinematics of a robotic manipulator, using an artificially generated set of data. The results achieved show that errors below 0.5 degrees can be reached, with further improvements being possible through the finer tuning of hyperparameter values (in cases such as Generations or

Sample Size which generally select between two possible values), or inclusion of larger possible values (in cases of Population Size and Crossover Rate hyperparameters which tend to the largest possible values). The main issue of the GP application determined by the authors is that models – while practically generated in the shape of equations, are quite large – with equation elements numbering in hundreds. Authors believe this may be addressed through the finer hyperparameter tuning within the GP algorithm (namely of the Parsimony Coefficient hyperparameter) or through the post-processing of obtained equations which could be simplified or approximated on the used ranges with relatively low errors, offering significantly easier use.

Acknowledgments

This research has been (partly) supported by the CEEPUS network CIII-HR-0108, European Regional Development Fund under the grant KK.01.1.1.01.0009 (DATACROSS), project CEKOM under the grant KK.01.2.2.03.0004, and University of Rijeka scientific grant uniri-tehnic-18-275-1447.

References

[1] Tian, X., Xu, Q. and Zhan, Q. *An analytical inverse kinematics solution with joint limits avoidance of 7-DOF anthropomorphic manipulators without offset.* Journal of the Franklin Institute, 358(2), pp.1252-1272. 2021.

[2] Solowjow, E., Ugalde, I., Shahapurkar, Y., Aparicio, J., Mahler, J., Satish, V., Goldberg, K. and Claussen, H., *Industrial Robot Grasping with Deep Learning using a Programmable Logic Controller (PLC).* In 2020 IEEE 16th International Conference on Automation Science and Engineering (CASE) (pp. 97-103). IEEE. 2020.

[3] Toquica, J. S., Oliveira, P. S., Souza, W. S., Motta, J. M. S., & Borges, D. L. *An analytical and a Deep Learning model for solving the inverse kinematic problem of an industrial parallel robot.* Computers & Industrial Engineering, 151, 106682. 2021

[4] Baressi Šegota, S., Anđelić, N., Lorencin, I., Saga, M. and Car, Z., *Path planning optimization of six-degree-of-freedom robotic manipulators using evolutionary algorithms.* International Journal of Advanced Robotic Systems, 17(2), p.1729881420908076. 2020.

[5] Wei, Y., Nie, X., Hiraga, M., Ohkura, K. and Car, Z., *Developing end-to-end control policies for robotic swarms using deep Q-learning.* Journal of Advanced Computational Intelligence and Intelligent Informatics, 23(5), pp.920-927. 2019.

[6] Wei, Y., Hiraga, M., Ohkura, K., & Car, Z. *Autonomous task allocation by artificial evolution for robotic swarms in complex tasks*. Artificial Life and Robotics, 24(1), 127-134. 2019.

[7] Koza, J.R. and Koza, J.R., Genetic programming: on the programming of computers by means of natural selection (Vol. 1). MIT press. 1992

[8] *ABB IRB 120 Operating Manual*, Zurich, Switzerland, ABB Ltd. 2017.

[9] Hastie, T., Tibshirani, R. and Friedman, J., *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media. 2009.

[10] Ferreira, J., Pedemonte, M. and Torres, A.I., 2019. A genetic programming approach for construction of surrogate models. In Computer Aided Chemical Engineering (Vol. 47, pp. 451-456). Elsevier.

[11] Willmott, C.J. and Matsuura, K., Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. Climate research, 30(1), pp.79-82. 2005.