On Solving the Inverse Kinematics Problem using Neural Networks

Akos Csiszar
ISW
University of Stuttgart
Stuttgart, Germany
akos.csiszar@isw.uni-stuttgart.de

Jan Eilers
Student
University of Stuttgart
Stuttgart, Germany
janeilers80@gmail.com

Alexander Verl
ISW
University of Stuttgart
Stuttgart, Germany
alexander.verl@isw.uni-stuttgart.de

***Index Terms*—robot; inverse kinematics; machine learning**

***Abstract*—Writing and solving the inverse kinematics equations of a robot is a cumbersome task. Furthermore, an analytic solution exists only for an ideal model and only if the structure of the robot meets certain criteria. If enhanced positioning precision is required, the robot needs to be calibrated. This eliminates most of the errors due to the differences in the ideal model and the real robot. Calibration methods require additional computations (usually of iterative nature) in the real-time cycle of the control system and are only capable of dealing with small differences between the ideal model and the real robot. In this paper a supervised learning based approach is proposed to solve the inverse kinematics problem and the calibration. Instead of creating and ideal model for a series of robots and calibrating each of them individually afterwards, the inverse kinematics function is learned using a neural network and so, it is tailored to one given robot, already including errors due to manufacturing and/or assembly tolerances. Moreover, it can also work for structures for which no analytic solution is possible. The preliminary results of this research are presented in this paper, covering two simple robot structures, a planar 2 DOF and a spatial 3 DOF structure, both with and without artificially introduced assembly errors (joint misalignments) which make analytical modeling unfeasible.**

I. Introduction

The number of industrial robots in use is continuously increasing [1]. Besides the industrial robots, hobby or toy roots for the ”do it yourself” community are also more and more common. Each time a robot’s mechanical structure is used in any way (modeling, simulation, control, etc.) the solution to the kinematic problem (either forward or reverse) is required. If the precision of the robot is of concern, calibration has to be included. Using calibration, the manufacturing and/or assembly tolerances of a robot are corrected (in some cases, temperature and load compensation might also be regarded). While the kinematic model is ideal and can be used for all robots of the same make and model, calibration is always unique to one robot and works similarly to a system identification process, for the exact geometry of the robot. It involves fixed, known references in the work space of the robot or a precise measurement system which gives accurate information about the pose of the end-effector and so, the differences between the real robot and the ideal kinematic model can be identified, usually by numerical iterative methods.

In this paper a machine learning approach is proposed to solve the (inverse) kinematics and calibration problems together. Instead of using the ideal kinematic model and improving the precision of the robot though calibration, we propose to learn the inverse kinematics (including the errors due to manufacturing and assembly tolerances) function, which is then associated with one single robot, not with a robot make and model.

Instead of a calibration process where only the differences to the ideal kinematic model are identified, the (inverse) kinematic function of the robot is learned. It is expected that this process will require more time and different equipment then the calibration process. However, the process itself is similar in nature to the calibration process. The robot, monitored by an external sensor system, executes predefined motions (e.g. space filling curves). Using this approach, the strict tolerances for manufacturing and assembly can be relaxed because the calibration method is not anymore restricted to small geometrical errors. By applying machine learning not just to the calibration problem but also to the inverse kinematics problem, the structure of the robot can also change, since special requirements that allow the analytical solution of the inverse kinematics problem (e.g. the intersection of the last three axes in one point) can be disregarded. The advantages and challenges of the this approach along with a possible applications have been presented in [4]. This paper presents preliminary results in solving the inverse kinematics problem using neural networks. Ideal robot kinematics models are altered to include errors due to assembly and manufacturing tolerances. Both ideal and altered models are used to generate the training data for the machine learning process. The neural networks are trained for solving the inverse kinematics problem and afterwards their performance is evaluated.

II. State of the Art

Many papers in the scientific literature deal with problems related to inverse kinematics and machine learning. Many of these target learning only a certain trajectory, not the general inverse kinematics function of a robot [3], [9], [5]. In these papers it is shown that by learning a trajectory a higher precision can be achieved as when in executing the pre-programmed trajectory the classical way. However, these

cannot be integrated into today's robot controllers which heavily rely in the inverse kinematics transformation [12]. Learning the inverse kinematics problem using neural networks can also be found in the scientific literature. In [8] and in [6] a 3 DOF planar inverse kinematics problem is learned, in [7] a 6 DOF robot using specialized neural network. However, in these works only the idealized kinematic model is considered.

Not only supervised learning is used for learning inverse kinematics in the scientific literature. In [2], using reinforcement learning, a goal seeking like behavior can be defined that can take a robot to is goal state with good precision. This and similar approaches [10] also cannot be integrated in a classical industrial robot control architecture.

The approach proposed in this paper for learning the inverse kinematics function of a robot is designed with today's robot controller structures in mind. This way existing robot controller software and hardware architectures can be left mostly unchanged but these can still benefit from this novel approach. Programming of the robot remains the same as before. The mathematical function currently based on implicit or explicit equations should be exchanged for a neural network, however the inputs and outputs of the function do not change. Furthermore, real-time execution capability is expected to be improved, when compared to iterative solution methods, given the deterministic nature of neural networks in the exploitation stage.

III. Concept

The main idea behind this research is to eliminate the need for developing (inverse) kinematic equation by hand and then adopting these to a given robot using calibration. Instead, in a way, the calibration process is extended. It is altered, so it does not only identifying the difference to the ideal kinematic model, it becomes a supervised learning process, learning the whole kinematic transformation of the robot, with manufacturing tolerances and assembly misalignments included. In figure 1 the overall architecture of the approach is presented. As it can be seen, the general architecture of a robot controller remains intact. The only changes are to the transformations and a further module is added which handles the tasks related to learning.

A. Obtaining The Training Set

In order to carry out the learning task, without any previous model, at a future stage of the research, a camera system is planned to be used to track the end-effector (from multiple points of view). (At the current stage of the research, errors are artificially included in simple forward kinematic models, to emulate assembly tolerances.) This measurement, together with the joint angles, based on signals coming from the joint sensors (usually encoders), will constitute the training set. In order to build up a training set which covers the whole work space, the motors are moved to describe so called space filling curves [11]. During these movements the joint angles and the end-effector coordinates are recorded by the Learning module and thus the training set is obtained. This training set includes all imperfections due to tolerances for a given robot. It is only valid for that given robot. The training set will contain duplicates in the Cartesian coordinate part, since a given end-effector pose in almost all cases and robot structures can be reached with more then one joint configuration. In this step everything is recorded and no filtering is applied to the data. It can expected that the training set requires a significant amount of memory. This is not considered a problem, since not the whole training set needs to be in memory, it can be written to a local hard disk or database on the network. Furthermore, memory is getting more and more available.

B. Supervised Learning

If the training set exists, the supervised learning process can be carried out. A first step is to select the supervised learning method. Support Vector Machines (SVN) and Neural Networks (NN) are popular supervised learning methods. The approach works regardless of the chosen method (although SVN might require significantly more memory). Since the training set has a serialized form, the training process does not necessarily has to be carried out on the robot controller. Running the training in the cloud or on GPU brings faster results and more available memory.

1) Multiple Solutions: As in the case when the inverse kinematics problem, the decision to use one or the other solution has to come from the user. For industrial robots this in many cases this is determined though the so called status and turn. These specify which solution should be calculated or, in other words, which formula has to be used to calculate a joint angle. In fact, the the inverse kinematics solutions coexists, but there is a selection made, based on external input, which of them is preferred in the current context.

The status and turn give an indication about which formulas have to be applied. This decision, made joint angle after joint angle can also be formulated in a more centralized manner. The choice to use one or the other solution can be made before or after the function definition. If made before, using all combinations of angles, a number of needed inverse kinematic functions result. Each of these contains only one solution and there are as many functions as possible inverse kinematic solutions. In the regions of the work space, where less solutions are possible some of these functions either return the same value as other (e.g. in singular poses), or these return complex solution (hence the target point is out of reach for one of the functions).

Dividing up the training set based on the joint angles (e.g. first joint angle negative or positive) results in a number of separate training data sets which do not carry multiple solutions. The ranges (both Cartesian and joint space) for which the training set has valid angles has to be stored. The training can be applied to each of these separate data set and the decision which one to choose happens exactly the same way as it does with current, analytic inverse kinematic functions.

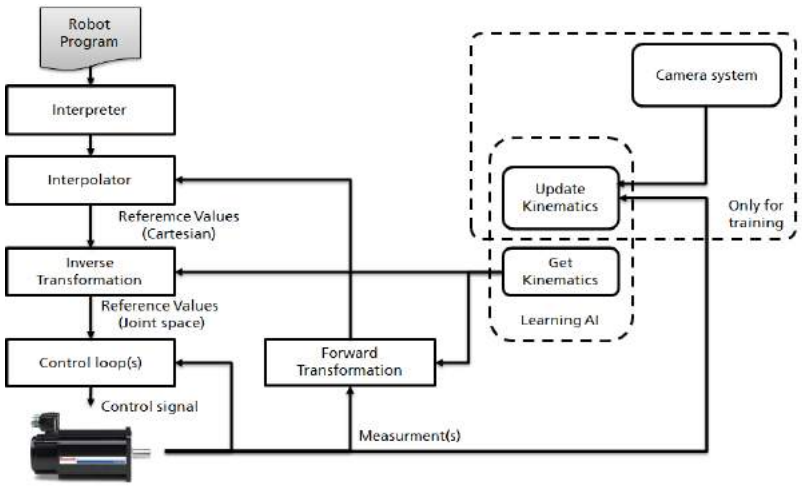

Fig. 1. The simplified overall architecture of the proposed concept

C. Precision Requirements

To be able to use the learned inverse kinematics it has to offer similar or better precision then its analytic counterpart. Please note that in this case the precision of the real robot is of interest not of the ideal model. Typical precision of robots is $\pm 0.1mm$, high accuracy robots have a precision of $\pm 0.05mm$.

IV. Implementation

At this stage of the research, simple examples have been considered, in order to examine how the NNs perform when learning the inverse kinematic function of a planar 2RR serial robot and a spatial 3RRR serial robot. Simulations have been carried out to compare the precision of the obtained NNs.

The selected neural network for learning the inverse kinematics problem is a feedforward network, with hidden layers, each having 50 neuron and TANSIG activation function, as presented in figure 2. The learning method was Levenberg-Marquardt with adaptive learning rate. The implementation and training has been carried out in Matlab, using the Neural Network Toolbox. The input to the neural network is the X and Y coordinate of the robot, additionally adding $\sqrt{X^2+Y^2}$ ad a third input. For the 3 DOF case X, Y and Z coordinates, as well a $\sqrt{X^2+Y^2+Z^2}$ are the inputs. Hence the 3 or 4 input neurons.

The output of the networks are not the joint angles directly. Instead, the sine and cosine functions of the 2 (or 3 in the 3 DOF case) are output from the networks, hence the 4 or 6 output neurons. Additionally, the $atan2()$ function is used to find the joint angle. At this stage no explicit feature scaling or batch normalization has been used. These can further improve the results. In order to avoid the over fitting problem, 60% of the available training set has been used for training the network, 20% for testing and 20% for validation.

A. Obtaining the Training Set

Four robot kinematic models have been considered. The first two models are based on a 2 DOF 2RR planar serial robot. DH parameters presented in table I, with and without artificially introduced manufacturing tolerances and joint assembly alignment errors which make analytical modeling of the structure unfeasible. The latter two cases are based on a 3 DOF 3RRR spatial serial robot, DH parameters presented in table II, also in this case, with and without artificially introduced joint alignment errors.

TABLE I
THE DH PARAMETER TABLE OF THE 2 DOF ROBOT

Linkage	θ	d	α	a
1	q_1	0	0	250
2	q_2	0	0	250

TABLE II
THE DH PARAMETER TABLE OF THE 3 DOF ROBOT

Linkage	θ	d	α	a
1	q_1	250	-π/2	0
2	q_2	0	0	250
3	q_3	0	0	50

The training set have been obtained using a homogeneous discretization of the joint space and the forward kinematics model. In the case of the models with introduced alignment errors, the following model has been used:

$$H = E \times T_1 \times E \times T_2 * E \times T_3 \quad (1)$$

where H is the overall $4x4$ transformation matrix, T_i are the DH transformation matrices and E is the misalignment $4x4$ transformation matrix, obtained as a 6 DOF transformation:

$$E = T_x \times T_y \times T_z \times R_x \times R_y \times R_z \quad (2)$$

having every translation $2mm$ and every rotation $2\,\mathrm{deg}$ in order to reflect a poorly assembles or manufactured robot.

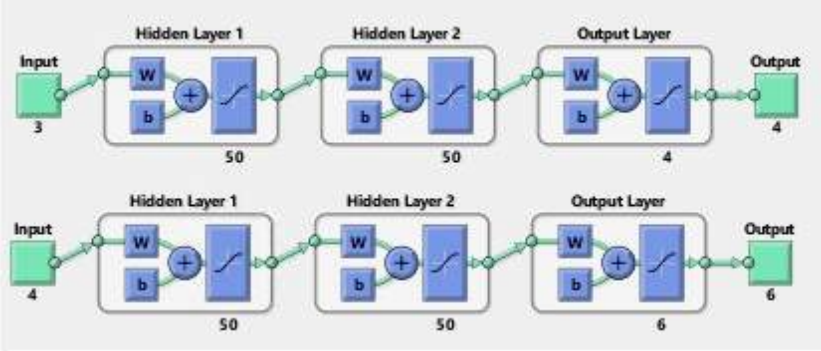

Fig. 2. The neural networks used for learning the inverse kinematic function, for 2DOF (upper) and 3DOF (lower)

The discretized area of the workspace for the 2 DOF case covers the range:

$$-\pi/2 < q_1 < 0; 0 < q_2 < \pi/2 \tag{3}$$

with a $0.01rad$ discretization, resulting in approximately 25.000 data points.

In the 3 DOF case, the covered workspace area is

$$-\pi/10 < q_1 < \pi/10; -\pi/2 < q_2 < 0; 0 < q_3 < \pi_2 \tag{4}$$

with a $0.03rad$ discretization, resulting in approx. 75.000 data points. As it can be observed, the area covered only presents 1 solution for the inverse kinematics problem. This is in line with the concept description above, solving the solution multiplicity problem was not the in the scope of these experiments.

B. Learning the IK Function

After obtaining the training data sets the NN have been trained to learn the inverse kinematics function. The training process have been carried out on PCs with 8 GB of RAM and Intel I7 processor, using Matlab Parallel Toolbox with 4 workers. The training performance has been measured by the mean squared error (MSE) function and an MSE value lower then $1e-09$ was the learning target in all cases. Learning times have 5 to 7 hours. It is assumed that this can be greatly improved by using GPU accelerated learning instead of parallel CPU learning.

V. Results

After training the neural networks, their performance has been compared to the forward kinematics function and the errors have been expressed as positioning errors in Cartesian space. Another set of test data has been generated, with a different grid size (not integer multiple) of the former discretization step. The joint angles have been converted to Cartesian space with the forward kinematics function. From here, the trained NN has been used to convert the Cartesian space values to joint space. These obtained joint space values have been converted back to Cartesian space using the same forward kinematics function, obtaining this way noter set of Cartesian positions. The difference between the two sets of Cartesian space values reflect the inaccuracies of the NN.

A. 2 DOF without joint misalignments

In this test case 99.16% of validation points have been below a Cartesian error threshold of 0.1 mm. 80.4% of point below a Cartesian error of 0.01 mm and 12% of points below a Cartesian error of 0.001 mm. Figure 3 shows the histogram of Cartesian errors and figure 4 shows the positioning errors in the workspace of the robot. As it can be observed the problematic area is the margin of the workspace, where due to homogeneous discretization in the joint space, only very few data points have been generated.

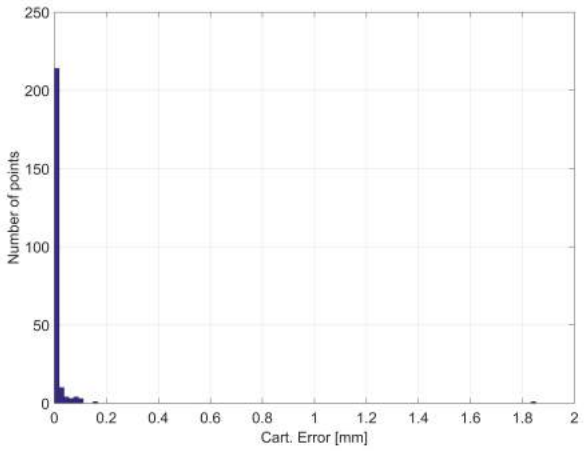

Fig. 3. Error histogram of the 2DOF case without joint misalignment

B. 2 DOF with joint misalignments

In this test case very similar results have been obtained. 98.98% of validation points are below a Cartesian error of 0.1mm. 83% of point are below a Cartesian error of 0.01 mm and 12% of points are below o Cartesian error of 0.001mm. Figure 5 shows the histogram of Cartesian errors and figure 6 shows the positioning errors in the workspace of the robot. Also here it can be observed the problematic area is the margin of the workspace, where due to homogeneous discretization in the joint space, only very few data points have been generated.

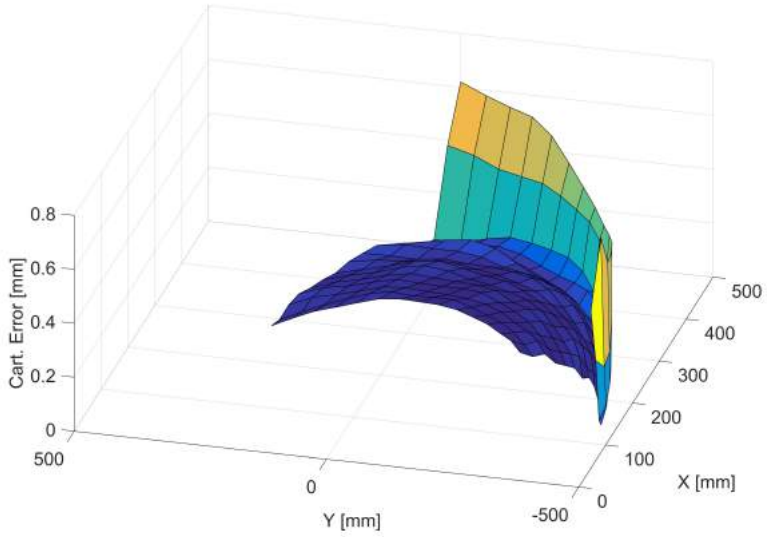

Fig. 4. Location and size of positioning errors in the workspace of the 2DOF case without joint misalignment

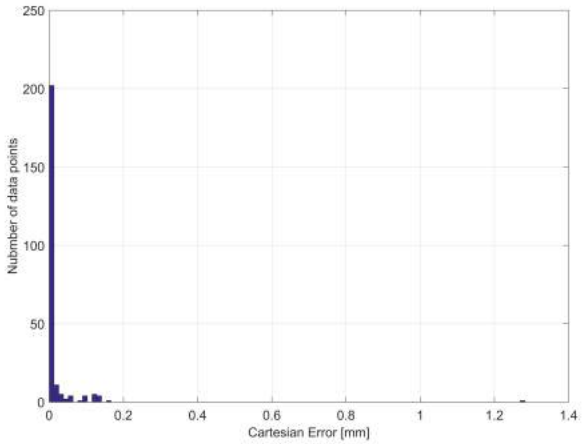

Fig. 5. Error histogram of the 2DOF case with joint misalignment

C. 3 DOF without joint misalignments

The results for this test case are statistically not different from the results of the next test case.

D. 3 DOF with joint misalignments

In this test case the results show that 99.95% of the validation points (not overlapping the training data) has a Cartesian error less then 0.1 mm, for 74.14% of points the positioning error using the NN inverse kinematics is smaller then 0.01 mm and for 3.25% of the test points it is smaller then 0.001 mm. The histogram showing the distribution of the positioning errors of the data set is presented in figure 7. The execution time of the trained network in Matlab (the NN as in-memory object, single threaded, without the Parallel Toolbox) varied between 0.009 s and 0.01 seconds. It is expected that in a real-time operating system the execution time and jitter will be significantly lower.

VI. Conclusions

When comparing the errors of the inverse kinematic functions with and without joint misalignment errors, it can be observed that for the neural network, from the results point of view, the misalignment does not make a difference. Hence further calibration methods will not be needed. The obtained precision shows that without prior network design and dimensioning promising results can be obtained.

In this paper a method to learn the inverse kinematic function of robots has been proposed. Furthermore, an architecture that is compatible with today's robot controller is suggested. This way, the way the robot is used and programmed does not have to be changed. Once the inverse kinematics is learned the robot will behave the same way as if it would be equipped with an analytic solution for the inverse kinematics problem. The learning process does not have to be carried out on the robot controller itself. Cloud based learning is expected to offer more memory and less computation time. The expected result of this approach is a more precise robot model, therefore a more precise robot, looser manufacturing tolerances and new robot structures, since no wrist joint is necessary for solving the inverse problem.

As the preliminary results are promising, the next step will be examine how neural networks size, network type, number of layers and activation function affect the training precision and execution time of the network. Furthermore, the homogeneous discretization in the joint space should be improved in order to improve the performance of the NN at the margins of the workspace.. The distant goal is to learn also higher order kinematic functions (velocities, accelerations) and add static

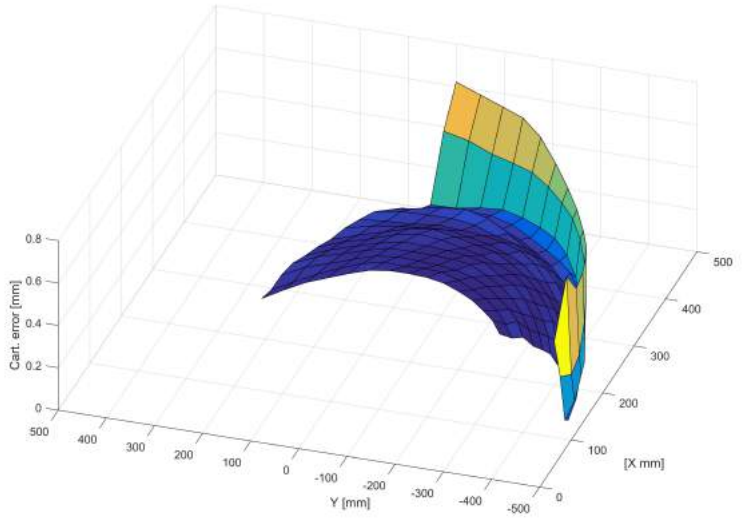

Fig. 6. Location and size of positioning errors in the workspace of the 2DOF case with joint misalignment

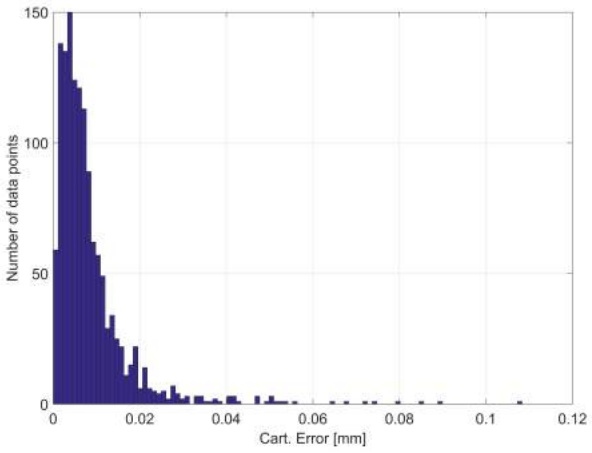

Fig. 7. Error histogram of the 3DOF case with joint misalignment

load compensation to the neural network. Network types other then feed forward networks are considered for these tasks.

Using this approach the need for kinematic modeling will be eliminated therefore any axis configuration of a serial chain will be usable as a robot structure. Furthermore, manufacturing and assembly tolerances will not play an important role, therefore robots will become less costly to produce. A robot screwed together from scrap metal can have the same precision as today's aluminum cast structures.

REFERENCES

[1] *World Robotics 2013*. IFR Statistical Department.

[2] Y. Ansari, E. Falotico, Y. Mollard, B. Busch, M. Cianchetti, and C. Laschi. A multiagent reinforcement learning approach for inverse kinematics of high dimensional manipulators with precision positioning. In *2016 6th IEEE International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, pages 457–463, June 2016.

[3] B. Bcsi, D. Nguyen-Tuong, L. Csat, B. Schlkopf, and J. Peters. Learning inverse kinematics with structured prediction. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 698–703, Sept 2011.

[4] A. Csiszar, P. Sommer, and A. Lechler. Ecobotics: Advantages and challenges of building a bamboo robot arm. In *2015 IEEE International Conference on Industrial Technology (ICIT)*, pages 192–197, March 2015.

[5] A. D'Souza, S. Vijayakumar, and S. Schaal. Learning inverse kinematics. In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, volume 1, pages 298–303 vol.1, 2001.

[6] Adrian-Vasile Duka. Neural network based inverse kinematics solution for trajectory tracking of a robotic arm. *Procedia Technology*, 12:20 – 27, 2014. The 7th International Conference Interdisciplinarity in Engineering, INTER-ENG 2013, 10-11 October 2013, Petru Maior University of Tirgu Mures, Romania.

[7] Rait Kker. A genetic algorithm approach to a neural-network-based inverse kinematics solution of robotic manipulators based on error minimization. *Information Sciences*, 222:528 – 543, 2013. Including Special Section on New Trends in Ambient Intelligence and Bio-inspired Systems.

[8] Rait Kker, Cemil z, Tark akar, and Hseyin Ekiz. A study of neural network based inverse kinematics solution for a three-joint robot. *Robotics and Autonomous Systems*, 49(3):227 – 234, 2004. Patterns and Autonomous Control.

[9] R. F. Reinhart and J. J. Steil. Reaching movement generation with a recurrent neural network based on learning inverse kinematics for the humanoid robot icub. In *2009 9th IEEE-RAS International Conference on Humanoid Robots*, pages 323–330, Dec 2009.

[10] M. Rolf and J. J. Steil. Efficient exploratory learning of inverse kinematics on a bionic elephant trunk. *IEEE Transactions on Neural Networks and Learning Systems*, 25(6):1147–1160, June 2014.

[11] Hans Sagan. *Space-filling curves*. Springer Science & Business Media, 2012.

[12] Bruno Siciliano, Lorenzo Sciavicco, and Luigi Villani. *Robotics : modelling, planning and control*. Advanced Textbooks in Control and Signal Processing. Springer, London, 2009. 013-81159.