Normalization of Inputs and Outputs of Neural Network Based Robotic Arm Controller in Role of Inverse Kinematic Model

Michal Puheim[1], Ladislav Madarász[2]
Department of Cybernetics and Artificial Intelligence
Technical University of Kosice
Letná 9, 042 00 Košice, Slovak Republic
[1]michal.puheim@tuke.sk, [2]ladislav.madarasz@tuke.sk

Abstract*—Goal of this paper is to discuss the methods usable to normalize inputs and outputs of the neural network controller used to control the arm of the humanoid robot with 3 degrees of freedom. The task of the controller is to solve the inverse kinematic problem, i.e. to move the hand of the humanoid robot to the target location given in arbitrary coordinate system other than its own kinematic chain defined by joint angle vector. In order to train accurate model for the controller it is necessary to normalize the values of input and output data in the training dataset. Data normalization within certain criteria, prior to the training process, is crucial to obtain satisfactory results as well as to fasten the training process itself. To proceed with the normalization we need to reduce domains of the training data in advance. Despite this task may look trivial, especially if I/O domains are clearly given, in some applications, such as finding the solution to the inverse kinematics problem of the humanoid robotic arm, it may become more complex and challenging. In this paper we will analyze possible options to perform normalization using expert oriented, automatic and hybrid approaches.

Keywords—normalization, inputs, outputs, neural network, arm controller, humanoid robot.

I. Introduction

In our previous paper [1] we proposed the novel stereovision object tracking system and we used this system as cognitive unit for robotic arm controller based on the neural network [2]. With this combined system [3] we aimed to control the robotic arm in order to touch or grasp the tracked object, i.e. to solve the inverse kinematics problem using the neural network. The results of the system were satisfactory, but improvement is possible. Our motivation for this paper is to discuss possible refinement of the system using the improved method of normalization of input and output data used to train the neural network controller.

Study presented in [4] systematically examined the effect of different data normalization procedures in the performance of neural networks trained with this data. It was discovered, that adequate normalization of input and output variables, prior to the training process is very important to obtain good results and significantly reduces the calculation time. Results of [4] also shown that with proper normalization, relatively small networks can perform as well as more complex ones.

Proposed neural network arm controller [2] is based on simple multilayer feed-forward neural network and acts as compensator, i.e. forward controller without feedback. Such configuration naturally cannot be as precise as similar feedback based systems. In order to achieve good precision, it is necessary to focus on the quality of the neural network training. Main idea behind the discussion in this paper is that with proper training data optimization, even controller with such simple design can be comparable to more complex solutions.

II. Importance of Proper Data Normalization for Processing with Neural Networks

Process of training of the neural network can be described as finding of the minimum within the $n+1$ sized error space, where n dimensions are determined by n weights and $(n+1)^{th}$ dimension determines the error of the output of the neural network.

In order to understand why proper data normalization can enhance neural network training speed and performance, it is important to remember that the neural network usually initializes weights to random values in range from –1 to 1. Let us assume, that slope of the sigmoidal functions used for neuron activation is same for all neurons. We consider normalizations as linear scale transformations, and thus the minimum to which the network converges should always be the same, only shifted by the same linear transformation. The initial state of the backpropagation algorithm is always a point in the vicinity of coordinate space origin, while distance to the desired minimum is dramatically changed by the scale of the search space. So, normalizations that compress the searching space to a unitary hypercube reduce the distance to the minimum, effectively speeding up the backpropagation algorithm. Furthermore, if scales are very different for various values, the bigger ones will have a higher contribution to the output error, and so the error reduction algorithm will be focused on the variables with higher values, neglecting the information from the small valued variables [4].

The best situation should be when all input variables are in the same order of magnitude, especially if they are in the order of one. However, to perform this transformation one has to define the absolute maximum and minimum values for each input variable. This can be difficult in many real problems [4].

III. Basics of Inverse Kinematics

When controlling the robotic arm, we usually directly measure its inner kinematic parameters - joint coordinates. These coordinates measure the position of joints. Robotic arm can have more DOF and it could be described in number of ways. We are usually interested in the position of the end effector in some other coordinate system (e.g. real world Cartesian coordinate system). Therefore our goal is to perform the mapping between those two descriptions of the robotic arm position [5].

Direct (forward) kinematics is a mapping from joint coordinate space to space of end-effector positions. We know the position of all (or some) individual joints and we are looking for the position of the end effector. Direct kinematics could be immediately used in coordinate measurement systems. Sensors within the joints inform us about the relative position of the links, joint coordinates. The goal is to calculate the position of the reference point of the measuring system [5].

Inverse kinematics is a mapping from space of end-effector positions to joint coordinate space. In this case we know the position of the end effector and we are looking for the coordinates of all individual joints. To control the robotic arm, the calculation of these joint coordinates is necessary [5].

IV. Neural Network Based Arm Controller

In order to test the novel stereovision tracking application presented in [1], we designed dedicated controller [2] to control motion of the arm of the Nao robot [6]. Our approach to controller design is similar to [7]. Goal is to enable the robot to touch the object tracked by the proposed stereovision system using the information from the output of the tracking system. For this purpose we used the multilayer feed-forward neural network with three inputs and three outputs and two hidden layers, each consisting of four neurons.

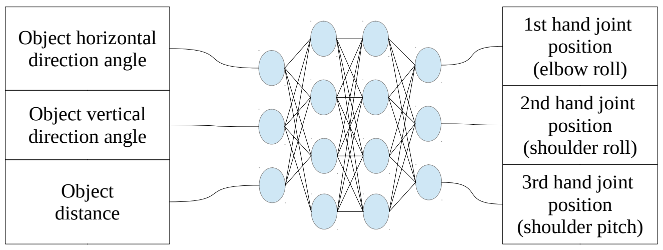
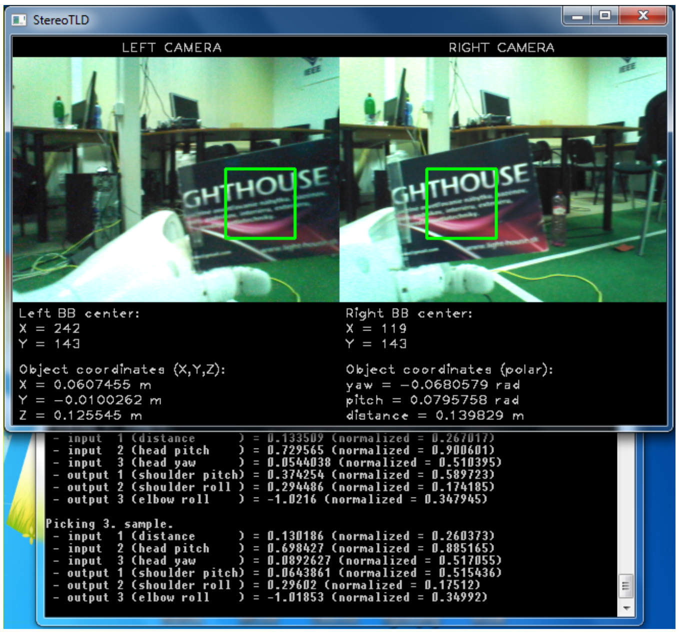

Object horizontal direction angle
Object vertical direction angle
Object distance
1st hand joint position (elbow roll)
2nd hand joint position (shoulder roll)
3rd hand joint position (shoulder pitch)

Fig. 1. Feed-forward neural network controller of robotic arm. Inputs are the information about the object. Outputs are the positions of the arm joints [2].

Inputs of the neural network include the available information obtained by the stereovision tracking system, such as distance to the object from cameras and angle difference from the central optical axis. Angles are given in horizontal and vertical orientation and are also adjusted according to the current neck joint turn of the robot. Outputs of the neural network are the values of the triplet of the motor positions which animate the joints of the robotic arm. In case of properly trained neural network we can move the arm of the robot simply by setting the motors to values given by the outputs of the network.

V. Training of the Neural Network Controller

In order to train the neural network it is necessary to gather training set of input and output data. In our case the output data are given by joint positions of the robotic arm. These positions can be read directly from the robot’s operating memory and we can also move robotic arm to any of these positions. The input data are given by stereovision tracking system. Our goal is to train the network to be able to convert the input data to the output data, i.e. to perform the *inverse* transformation of the coordinates of the tracked object to the corresponding joint positions. To generate the training data pairs we can exploit the *direct* transformation, where we use the tracking system to measure the coordinates. This can be done by mounting the tracked object to the robot’s hand, setting the motors to various random values and reading the information of joint positions from robot’s memory and object coordinates from the tracking system (see Fig 1.). More detailed description of training data acquisition can be found in [2].



```
Left BB center:                          Right BB center:
X = 242                                  X = 119
Y = 143                                  Y = 143

Object coordinates (X,Y,Z):              Object coordinates (polar):
X = 0.0607455 m                          yaw = -0.0680579 rad
Y = -0.0100262 m                         pitch = 0.0795758 rad
Z = 0.125545 m                           distance = 0.139829 m
```

```
 - input  1 (distance      ) = 0.133509 (normalized = 0.267017)
 - input  2 (head pitch    ) = 0.729565 (normalized = 0.900601)
 - input  3 (head yaw      ) = 0.0544038 (normalized = 0.510395)
 - output 1 (shoulder pitch) = 0.374254 (normalized = 0.589723)
 - output 2 (shoulder roll ) = 0.294486 (normalized = 0.174185)
 - output 3 (elbow roll    ) = -1.0216 (normalized = 0.347945)

Picking 3. sample.
 - input  1 (distance      ) = 0.130186 (normalized = 0.260373)
 - input  2 (head pitch    ) = 0.698427 (normalized = 0.885165)
 - input  3 (head yaw      ) = 0.0892627 (normalized = 0.517055)
 - output 1 (shoulder pitch) = 0.0643861 (normalized = 0.515436)
 - output 2 (shoulder roll ) = 0.29602 (normalized = 0.17512)
 - output 3 (elbow roll    ) = -1.01853 (normalized = 0.34992)
```

Fig. 2. Generation of input and output data, which are later used to train the neural network. Data are normalized to values from the interval [0, 1].

VI. Normalization of Inputs and Outputs of the Neural Network Controller

In neural networks all input and output values need to be normalized in range between 0 and 1. In order to normalize the value x of the data from interval $[a, b]$ to the value x_n from interval [0, 1], we can use the following equation:

$$x_n = \frac{x-a}{b-a} \,. \quad (1)$$

With consideration of results of [4] presented in section II. of this paper, we can assume that for successful training it is necessary for training data to cover whole range of the input and the output domains. In other words, it is suitable that the data after normalization will be equally distributed within the interval [0, 1], i.e. there will be approximately same number of examples close to 0 as the number of examples close to 1. Therefore, the problem is how to correctly determine the domain limits *a* and *b*.

A. Expert Oriented Normalization Based on the Construction Limits of the Robot

In case that domain of the given input (or output) is infinitely large, it needs to be appropriately bounded. For example the distance of the object from the robot can obtain values in range from 0 to ∞ and for purpose of the normalization it is necessary to limit its upper boundary to some suitable value. Due to the fact, that the neural network will control the arm of the robot, which is no longer than 30 centimeters, there is no point to consider values greater than this number.

In case of the arm joints we considered the upper and lower boundary from movement range limits given by the robot manufacturer [8]. These limits are depicted in Fig. 1. and described in Table I.

TABLE I. MOVEMENT LIMITS OF SELECTED JOINTS OF NAO ROBOT.

Joint	Range in degrees	Range in radians
LShoulderPitch	-119.5 to 119.5	-2.0857 to 2.0857
LShoulderRoll	-18 to 76	-0.3142 to 1.3265
LElbowYaw	-119.5 to 119.5	-2.0857 to 2.0857
LElbowRoll	-88.5 to -2	-1.5446 to -0.0349
LWristYaw	-104.5 to 104.5	-1.8238 to 1.8238
HeadYaw	-119.5 to 119.5	-2.0857 to 2.0857
HeadPitch	-38.5 to 29.5	-0.6720 to 0.5149

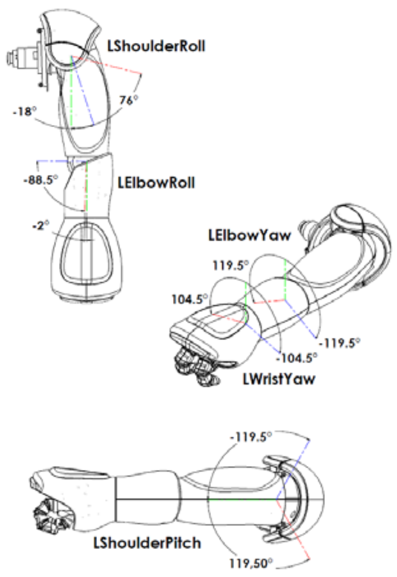


Fig. 3. Movement range of the shoulder, elbow and wrist joints of the left arm of the Nao humanoid robot [8].

Domains of the object direction angles were determined in a little different way. These angles are given by two components for each direction (horizontal and vertical). The components are head turn and additional shift of the object which occurs if the robot does not look straight to the object. For illustration, the horizontal object orientation angle is determined by sum of the HeadYaw joint value and the horizontal shift of the object from the center of the optical axes, which is provided by the stereovision tracking system. If the HeadYaw is set to maximum value, then we also need to consider the additional angle equal to the half of the horizontal view angle of the camera. Because of this the intervals which determine horizontal and vertical object angle domain boundaries were calculated as:

$$I_H = [-2.0857 - 0.5\lambda_h ; 2.0857 + 0.5\lambda_h] , \quad (2)$$

$$I_V = [-0.6720 - 0.5\lambda_v ; 0.5149 + 0.5\lambda_v] , \quad (3)$$

where I_h is horizontal angle interval, I_v is vertical angle interval, λ_h is horizontal view angle of the camera and λ_v is vertical view of the camera.

B. Automatic Normalization Using the Accessible Workspace

Even after the effort to manually determine correct intervals for input and output domains (described in section *A*), it is clear that in some situations it would not be possible to gather training data for whole domain range. For example if we mount the object in the left hand, large part of space on the right side of the robot will not be covered, because the hand simply cannot reach there, therefore we cannot obtain values in whole horizontal object angle domain. Likewise the robot can move its arm to such position, where the tracking system is unable to track the object anymore (see Fig 2.). As a result, for some joints it is not possible to gather data to cover whole domain range.

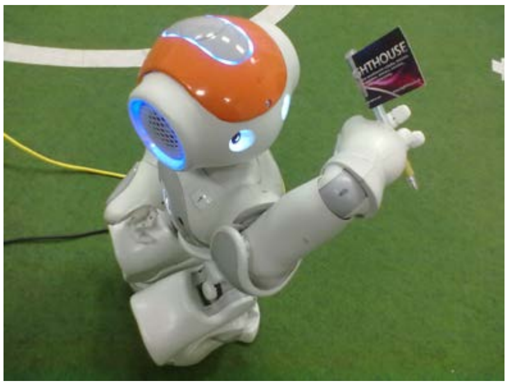


Fig. 4. Illustration of possible arm position, which is still within the construction limits given by the documentation. Robot is able to move the hand to this position but is no longer able to turn head in order to track the given object.

Better idea is to set domain range boundaries using the training dataset. In this approach we do not set domain boundaries in forward according to the documentation, but only after training data generation has finished. Domain boundaries for individual inputs and outputs are defined by respective maximum and minimum values within the training data. This way we can reduce the whole operational range of the neural network to actually accessible working space, which should enhance the speed of training and also resulting precision during actual control of the robotic arm.

C. Semi-Automated Normalization Using Lead-Through Learning

In previous section we solved the problem of proper normalization by finding the extremal values for input and output domains using the training data. This can be considered as optimal solution, but is dependant on quality of training data. In this section we focus on finding the way to optimize the data and further improve the process of training the neural network.

Problem with automatic training data generation is that it is random and some generated positions may not be natural. Furthermore it is well known, that the inverse kinematics problem (i.e. transformation from the real world coordinate system to joint coordinate system) can have multiple contradictory solutions for some positions [9]. Therefore automatic data generation can and will produce multiple outputs for the same input, which is confusing when used in training of the simple feed-forward neural network.

To solve this problem, we can use the lead-through approach similar to [10]. In this approach human operator moves the arm of the robot manually by hand with goal of showing the robot the positions potentially suitable for input/output data acquisition. It is important to move the arm in natural way and avoid difficult or unnecessary positions [11]. It is also proficient to avoid positions in which the robots tracking system clearly cannot see tracked object within the hand (e.g. positions close to the waist of the robot etc.).

Alternatively, it is possible to use automated technique, where the training data is gathered automatically, but examples with similar inputs and different outputs are filtered in such way, that only the optimal corresponding input/output example is preserved [12]. Optimal example can be chosen according to some heuristic function, e.g. the sum of absolute angle values of all joints, which would later lead the training algorithm of the neural network to most simple and natural arm positioning.

VII. CONCLUSIONS

In this paper we focused on the explanation of the importance of training data normalization and we further discussed the various methods of normalization and optimization of this data using the model application of simple neural network controller of humanoid robotic arm with three degrees of freedom.

There are multiple sophisticated neural network based solutions such as [9], which solve the inverse kinematic problem using complex system of multiple modular neural networks. Such methods often require complicated learning algorithms and lengthy implementation. Our goal in this paper was to discuss the idea that with proper training data optimization we can solve this problem even with simpler neural network, especially in cases where degree of freedom of the robotic arm is equal to number of dimensions of considered world coordinate system.

ACKNOWLEDGEMENT

The work presented in this paper was supported by VEGA, Grant Agency of Ministry of Education and Academy Science of Slovak Republic under Grant No. 1/0298/12 – “Digital control of complex systems with two degrees of freedom.” The work presented in this paper was also supported by KEGA under Grant No. 018TUKE-4/2012 – “Progressive methods of education in the area of control and modeling of complex systems object oriented on aircraft turbo-compressor engines.”

REFERENCES

[1] M. Puheim, M. Bundzel, P. Sinčák, L. Madarász: “Application of Tracking-Learning-Detection for object tracking in stereoscopic images”. Symposium on Emergent Trends in Artificial Intelligence and Robotics. Košice. Sep. 2013.

[2] M. Puheim, M. Bundzel, L. Madarász: “Forward Control of Robotic Arm Using the Information from Stereo-vision Tracking System”. 14th IEEE International Symposium on Computational Intelligence and Informatics. Budapest. Nov. 2013.

[3] M. Puheim.: “Application of TLD for object tracking in stereoscopic images”. Master diploma thesis (in slovak). Technical University of Košice. Faculty of Electrical Engineering and Informatics. 68 pages. 2013.

[4] J. Sola, J. Sevilla: "Importance of input data normalization for the application of neural networks to complex industrial problems," IEEE Transactions on Nuclear Science, vol.44, no.3, pp.1464,1468, Jun 1997

[5] V. Smutný: “Direct and Inverse Kinematics”, [Online]. 2013. Available: http://cmp.felk.cvut.cz/cmp/courses/ROB/roblec/serial-noteeng.pdf (Cited: 11. October 2013)

[6] Aldebaran Robotics: “Nao Website”, [Online]. 2013. Available: http://www.aldebaran-robotics.com/en (Cited: 22. April 2013)

[7] K. K. Dash, B.B. Choudhury, A. K. Khuntia, B.B. Biswal: "A neural network based inverse kinematic problem," Recent Advances in Intelligent Computational Systems (RAICS), 2011 IEEE, pp.471,476, 22-24 Sept. 2011.

[8] Aldebaran Robotics: “Nao Documentation v1.14.2”. [Online]. 2013. Available http://www.aldebaran-robotics.com/documentation/index.html (Cited: 24. April 2013)

[9] E. Oyama, T. Maeda, J.Q. Gan, E.M. Rosales, K.F. MacDorman, S. Tachi; A. Agah, "Inverse kinematics learning for robotic arms with fewer degrees of freedom by modular neural network systems," International Conference on Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ, pp.1791,1798, 2-6 Aug. 2005.

[10] T. L. Graf: “Lead-through robot programming system”, U.S. Patent 5880956 A, 9. Mar. 1999.

[11] Tar, J.K., Rudas, I., Kósi, K., Csapó Á., Baranyi, P.: “Cognitive Control Initiative”. 3rd IEEE International Conference on Cognitive Infocommunications, December 2-5, 2012, Košice, Slovakia. pp. 579-584. ISBN 978-1-4673-5188-1

[12] Nagy, I., Várkonyi-Kóczy, A.R., Dineva, A.: GA Optimization and Parameter tuning at the Mobile Robot Map Building Process. IEEE 9th International conference on Computational Cybernetics, July 8-10, 2013. Tihany, Hungary. pp. 333-338. ISBN 978-1-4799-0060-2.