WIRELESS KINECT-NAO FRAMEWORK BASED ON TAKAGI-SUGENO FUZZY INFERENCE SYSTEM

A. Lekova, A. Krastev, I. Chavdarov

Institute of Systems Engineering and Robotics, Bulgarian Academy of Sciences, Sofia, "Acad G. Bonchev" str., bl.2, tel. +3599793232, e-mails:
alekova@ifi.uio.no; aikrastev.iser.bas@gmail.com; van_chavdarov@dir.bg

Abstract: In the context of learning new skills by imitation for children with special educational needs, we propose Wireless Kinect-NAO Framework (WKNF) for robot teleoperation based on Takagi-Sugeno (T-S) Fuzzy Inference System. The innovative solutions here are related to view invariant teleoperation, work online, smoothness in motion retargeting by median filter and fuzzifying of depth Kinect data, motion mapping of Human to NAO movements by featured angles rather than direct Inverse Kinematics angles. The nonlinearity of the observed 3D Kinect angles in different offsets is linearly approximated by T-S fuzzy rules of zero and first order that have local support in 2D projections. The feasibility of framework has been proven by real experiments.

Key words: Teleoperation, Telerobotics, Takagi–Sugeno fuzzy system, View invariance, Microsoft Kinect V2, NAO robot.

INTRODUCTION

Imitation involves a child's ability to copy others and helps children to learn new things and movements. Unfortunately, children with special educational needs often have difficulty with imitation. They lack the ability to share a focus on humans, however are attracted to robots and computerized technologies. Therefore, we seek for joyful play environment exploiting assistive technologies to enhance children's ability to imitate. In the context of the project [1], a Microsoft Kinect sensor [2] is used for motion-sensing and mediating the process of doing things by teleoperation to replicate the human movements on the robot. Nowadays, teleoperation (or also motion retargeting) is achieved by sensors on the human or by external observations over time. Since the marker based motion capturing systems are expensive and require careful calibration, a lot of work has been done to study imitation by external observations for extracting 3D poses from an image sequence. Tracking the human motion is an attempt a kinematic model of the robot to be recovered from the video sequences and to be an input for the kinematic modules of the robot. The teleoperation process has two stages: the operator's calibration stage and motion mapping to the robot.
In the present study, Kinect V2 sensor is used for teleoperation of Aldebaran NAO humanoid robot V 2.1.4 [3]. These two technologies used for imitation in the context of a play are expected to be easily set-up at day-care centers or schools from people without engineering skills. So, the calibration of their integration should be easy. Observing the changes in the human and robot movement we faced a variety of types of problems during design and implementation coming from the following requirements: (1) need of Kinect data smoothing and filtering; (2) need of real-time operation; (3) smooth function for motion retargeting without false sudden shifts that cause the robot to move abruptly; (4) view-invariance and scalability of the Inverse Kinematics solutions.
Changes in body movements should be incorporated online and in real-time. There are two Aldebaran's Inverse Kinematics (IK) functions on the robot side to control the arm's joints and move the hand point to a given position [3] by passing the coordinates of a hand end positions or by using transformation matrices to pass parameters to direct control of arm's joints. However, Aldebaran's IK function works correctly only after passing the orientation of the hand point from Kinect. Additionally, the noisy Kinect readings cause continuously changing joint angles and results in abrupt movement of the arms even in steady state. The Kinect joint positions data are not perfectly precise, meaning that they are scattered around the correct joint positions in each frame and are accurate within a centimetre range, not millimetre [4]. When the functions using these data are not smooth (e.g. in calculating forearm size that participate in formulae for angles) the sudden Kinect spikes will cause the robot to move abruptly while the user is barely changing pose.
Existing Solutions. The noise and variations of readings are taken care of by the margin of error [5] or by filtering the Kinect data [6]. The double exponential smoothing filter [4] is the most used smoothing filter for Kinect data smoothing [6]. During calibration a complex reference coordination between Kinect and robot coordinate frames requires a lot of code and skills that therapists or educators do not have. Often the operator stays at predefined area in front of the Kinect view [7] or complex transformation matrices are used for calibration between Kinect and NAO coordinate systems since different areas of the input space require different compensations and scale independence.
An adaptive neuro-fuzzy inference system (ANFIS) for motion mapping is proposed in [5], where three methods to imitate human upper body motion are implemented on a NAO robot and compared: (1) direct angle mapping method (2) IK using fuzzy logic and (3) IK using iterative Jacobian. The direct method requires the coordinates of three joints (shoulder, elbow and wrist) to determine NAO angles. However, continuously changing angles for the positions result in jerky movement. The IK using ANFIS method requires only two joint coordinates and is found to be more efficient and fast because the fuzzy system is trained a priory and there are not many computations involved in mapping the coordinates of the end-effector to the joint angles. However, the training of the ANFIS can take a long time depending on the amount of training data. Moreover, the first method is used to train ANFIS and training phase could not be performed from inexperienced persons. Solving the IK problem iteratively using the Jacobian pseudo-inverse requires two joints as well, but is found to be inefficient because a lot of iterations are required at each step and the response of the robot is very slow. This method also gets

stuck in singularities. To the best of our knowledge we didn't find related works for Kinect–robot teleoperation based on fuzzy depth data processing and motion retargeting. The Takagi-Sugeno (T-S) fuzzy systems are mainly used for robot control, not for approximation reasoning over sensor data, as we use it.
Considering the above problem requirements, we propose a Wireless Kinect-NAO Framework (WKNF) for teleoperation based on Takagi-Sugeno Fuzzy Inference System (T-SFIS). The wireless framework connects middleware on Kinect side and NAO robot side to transmit data to robot actuators. IK and T-SFIS algorithms are online and lightweight in order not to block the wireless connection to robot Python scripts in real time. We have made a trade-off between latency and precision in teleoperation by approximation reasoning. T-SFIS is chosen as universal approximator since it presents a low time response using a set of simple functions that require low CPU and memory resources.
During the design and implementation of WKNF we found out several innovative solutions, proposed in the next Section. WKNF is view invariant considering the parallax effect and normalize the calculated angles and distances. It works well online, the smoothness in motion retargeting is ensured by median filter of depth Kinect data and fuzzyfication of distances and angles calculated by vector algebra. Motion mapping is performed by featured angles rather than direct angles. Featured angles are stable in the vision area of Kinect and at least one of the joints establishing the vectors are not quickly changing joints, resulting in less scattered Kinect readings. We exploit trigonometric functions that stop amplifying the data noise. T-S fuzzy rules of zero and first order that have local support in 2D projections according to offsets of body parts are used.

PROPOSED SOLUTIONS

In WKNF for motion retargeting we have to determine the joint angles for the robot actuators to set a desired trajectory on the basic of positions and orientations of the robot end-effectors.

Processing Kinect body data to solve Inverse Kinematics task. We analyse the Kinect depth and body stream data and identify the 3D positions of upper body limbs over time. The important joints for motion retargeting of upper limbs are left and right shoulder, elbow, wrist and hand. During the movement, the length and angles between each joint are changing at each frame and some of them are considered as important features to map the Kinect angles to angles of robot actuators. To reduce the spikes, we apply Median filter and we use it only for joints that are more often in "Inferred State", such as hand tips and thumbs. This filter doesn't introduce latency because it doesn't take advantage of the statistical distribution of data or noise.
First we have tried to solve the problem for motion retargeting using direct angle analytical method. However due to its poor performance and ambiguous results, we searched for unique angles that can feature the movement by θ_1 to θ_5 over time and map them to NAO actuator angles. Featured angles are stable in the vision area of Kinect and the joints forming the vectors are not quickly changing joints, resulting in less scattered Kinect readings. The extra joints we use for motion retargeting are head, thumb and tip (see Figure 1 a).
The angle between two vectors defined by the three joints P^{th}, Q^{th} and R^{th} with coordinates: (x_p, y_p, z_p) (x_q, y_q, z_q) and (x_r, y_r, z_r), then the two vectors are $\overrightarrow{PQ}$ and $\overrightarrow{QR}$. The angle between them is calculated by using equation (1).

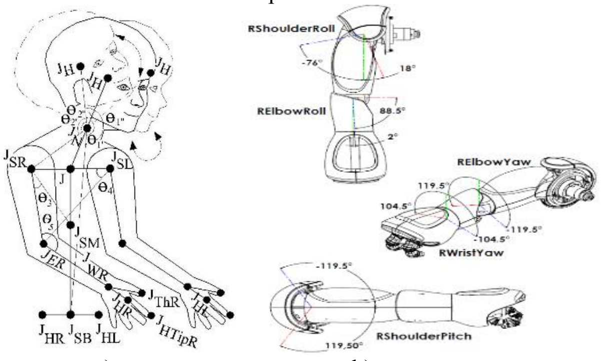
$$\theta_{pqr} = \cos^{-1}\left[\frac{\overrightarrow{PQ}\cdot\overrightarrow{QR}}{\|\overrightarrow{PQ}\|\,\|\overrightarrow{QR}\|}\right] \tag{1}$$

where $\overrightarrow{PQ}\cdot\overrightarrow{QR}$ is the dot product (Eq.2) and $\|\overrightarrow{PQ}\|$ and $\|\overrightarrow{QR}\|$ are the lengths (Eq.3) .

$$\overrightarrow{PQ}\cdot\overrightarrow{QR} = (x_q - x_p)*(x_r - x_q) + (y_q - y_p)*(y_r - y_q) + (z_q - z_p)*(z_r - z_q) \tag{2}$$

$$\|\overrightarrow{PQ}\| = \sqrt{(x_q - x_p)^2 + (y_q - y_p)^2 + (z_q - z_p)^2} \tag{3}$$

During the implementation we established that the noisy Kinect readings result in noisy calculations for distances (we use Euclidean formula) and 3D angles, different in the different area of the input space. One of the reasons is that operations to calculate body part sizes and relative positions such as addition, subtraction and multiplication, amplify the noise. Trigonometric functions which are typically used for calculating joint angles affect the noise in different ways. We use arccosine that results in small radian values and even decrease the noise. For unavoidable dependency on offsets of body parts, we implement fuzzy rules that have local support in slit planes in 2D projections (see Figure 4). Another reason for incorrect calculations is the observed parallax effect.



a) b)
Figure 1. a) Important joints for motion retargeting of upper limbs b) 3D model of the Nao right upper limb.

View invariance and scalability. Parallax arises due to change in viewpoint occurring to motion of the observer. Since the visual angle of an object projected onto the retina decreases with distance, this information can be combined with previous knowledge of the object's size (etalon) to determine the absolute depth of the object. By measuring angles, and using geometry, one can determine the length of an object – *x* if *r* is the radius between the object and the eye. From the Kinect's eye perspective, the changes in sizes a Kinect would see when the user move far and close we also explain with motion parallax effect. Consequently, the depth coordinate Z affects angle calculations because the lengths of body parts participate in the formulae. However, if we have an etalon (size(s) taken during the Kinect calibration stage) we can measure the change in the angle relative to human position in front of Kinect and to correct angles on the Kinect side according to current visual angle (α_{cur}) using Equations 4. In WKNF, the 2D distances for closeness are normalized either by the depth value Z (for tip and thumb joints) or by dividing to α_{cur} (for bigger limbs such as hip and elbow).

$$\alpha_{etallon} = x/r * 180/\pi; \qquad \alpha_{curr} = x_{curr}/r_{curr} * 180/\pi$$
$$\alpha_{correct} = \alpha_{etallon} - \alpha_{curr} \tag{4}$$

The Need of Fuzzy-Based Reasoning. Kinect and NAO have different coordinate systems, as well as NAO has joint limitations (see Figure 6 a). Because of the context of the application (play for children) we made a trade-off between the precision of motion retargeting, easy calibration and NAO response delay. We applied approximate reasoning to handle Kinect noise readings and provide a universal approach for motion mapping

instead to do a complex calibration between Kinect and NAO coordinate systems.
Fuzzy logic is a superset of classical logic with the introduction of "degree of membership". Uncertainties are presented as fuzzy sets (A_i), which are often expressed by words and interpreted by their membership functions $\mu_{A.}$ We exploit the Takagi-Sugeno (T-S) fuzzy model as a universal function approximator [8]. Its structure consists of rules in the form:

$$R_i : IF\ x \quad is\ A_i\ THEN\ y_i = a_0^i + \sum_{k=1}^{n} a_k^i x_k \tag{5}$$

where $x = (x_1, x_2, ..., x_n) \in D$ is a vector representing the inputs defined on *D*. A_i is a fuzzy set defined on certain domain (*D)*; y_i is a scalar output corresponding to rule i ; a_k^i are the consequence parameters associated to rule *i*. $i \in \{1,...,p\}$, where *p* is the number of rules. For a zero-order T-S model, the output level y is a constant ($a_0 = a_k = 0$). The rules are aggregated and defuzzified by using the fuzzy-mean formula:

$$y(x) = \sum_{i=1}^{p} \mu_{A_{ik}}(x) * y_i \Bigg/ \sum_{i=1}^{p} \mu_{A_i}(x) \tag{6}$$

where $\mu_{A_{ik}}(x)$ is the degree of fulfillment of *i*-th rule.

$$\mu_{A_i}(x) = \mu_{A_{ik}}(x_1, x_2, ..., x_n) = \underset{k=1}{\overset{n'}{T}} \mu_{A_{ik}}(x_k),$$

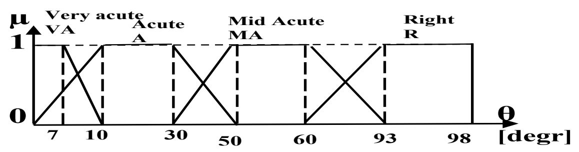
where n' is the number of input variables in *i*-th rule ($n' \le n$), ***T*** is a type of t-(co)norm as minimum, product, etc.

During the design we identify premise and consequence parameters and write a fuzzy-rules base (FRB) for mapping angles from Kinect to NAO coordinate frames over time. We use fuzzy trapezoidal membership functions (simple and fast for calculation), where the upper base of a trapezoid takes care of small scattering, i.e. ignoring the scattering in centimetre range that cause the robot to move abruptly while the user is barely changing pose. We observed nonlinearity during the mapping for some of the 3D Kinect to NAO angles in different offsets. For instance, during the mapping of the angle values to move the hand to a given location, NAO right shoulder roll angle (see Figure 3b) has constant values in different hand-body offsets, while the 3D Kinect angle θ4 is changing in dependence on distances between 2D joint positions of the right hip and elbow. The proposed solution is to partition the 3D input space in several projections such that a linear approximation of unknown mapping relation is possible in these ranges with accepted error less than 5%. The goal is to obtain the premise and consequence parameters of T-S and the required model error ε. If the error is less than 5% zero-order T-S is used, otherwise first-order, second, etc., till the error gets below 5%. The less rules used (i.e. coarse fuzzy partition) the bigger error. The first step is in several 2D projections to identify the group of states having similar linear approximation of mapping, and to distinguish these states as fuzzy sets. Their membership functions will classify the Kinect angle in degrees into these fuzzy sets. The second step is to find the local approximation linear solutions and to aggregate and defuzzifying them by equation (6).

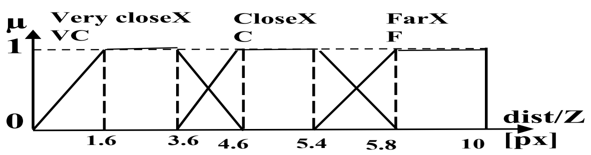
IMPLEMENTATION AND EVALUATION

Technical Specification. Kinect V2 sensor is connected to a laptop with Intel(R) Core (TM) i7-5500U CPU@ 2.40 GHz and transmits sensor data to a software application running on the laptop, built in C#, referencing Microsoft Kinect library (SDK 2.0) and performing data pre-processing and motion retargeting. Apart from the robot hardware and Kinect sensor, no other hardware is used. Three middleware are connected: Kinect SDK2 [2], NAO.NET on the Kinect side [9] and Naoqi 2.1.4 on the robot side [3]. Apart from those middleware, the framework is built only in C# on the Kinect side and Python scripts on the robot side. These codes run in parallel - first running on a laptop and connected to the Kinect sensor to provide body joint tracking and fuzzy logic processing, and second running on the robot side - waiting to receive data from Kinect side via Wi-Fi and use it to make the move of the robot actuators. The latency of fuzzy reasoning (how much time it takes for robot output to catch up to the actual human joint position when there is a movement in a joint) is not introduced in CPU time it takes for executing the T-SFIS. In general, the delay depends on the number of the simultaneously open Python scripts and the number of per frame passing parameters to these scripts. The reasonable trade-off between precision and latency is proven by real experiments (Figure 2), where we infer the angles for the robot actuators over time by fuzzy reasoning via numerically processing of the information in the fuzzy rules.
On Kinect side we process the 3D coordinates of joints at each consecutive frame. The applied Median filter output is the median of the last N inputs (joint positions). It latency depends on N and we use it only for N=20 Kinect frames. We calculate the joint angles by analytical IK per frame. The Kinect tracking algorithm operates in 3D camera space, however for some offsets (such as closeness), limb sizes and angles we use the Kinect Coordinate Mapping to project 3D points from camera space to a row/column location in the depth space with origin x=0, y=0 corresponding to the top left corner of the depth image. Thus, we operate only with positive values for joint coordinates.
Modelling of uncertainties consists of information about linguistic variables, domains, constraints as fuzzy sets (Figure 3) and fuzzifiers. The values of constraints, normalized in the range [0÷1], take part in the premise parameters in IF-THEN fuzzy rules, such as Kinect angles and/or 2D distances. The rules map the input values to the output space in terms of implication relation between fuzzy sets in "IF" and functions in "THEN" parts. Fuzzified input data trigger one or several rules in the fuzzy model to calculate the result. Two type of functions for motion mapping of Kinect angles to NAO space in "THEN" parts are designed: function approximation using first-order T-S model and zero-order T-S model.

Figure 2. Real experiments with children



a) for angle θ_3



b) for distances in XZ slit

Figure 3. Premise parameters

To cope with the nonlinearity of the observed Kinect angles in X and Y split planes where the 3D Kinect angles vary (see Figure 4), we implemented the proposed solution to partition the 3D input space and decompose the mapping of Kinect to NAO in the slit planes in XZ and YZ projections and use first-order T-S Linguistic values (semantically close values) for angles in different slit planes according to the closeness of the hand to the body and normalized by current parallax angle, form the premise parameters participating in the fuzzy rules that are presented in Figures 5.

Consequences are the equations of the trend lines.

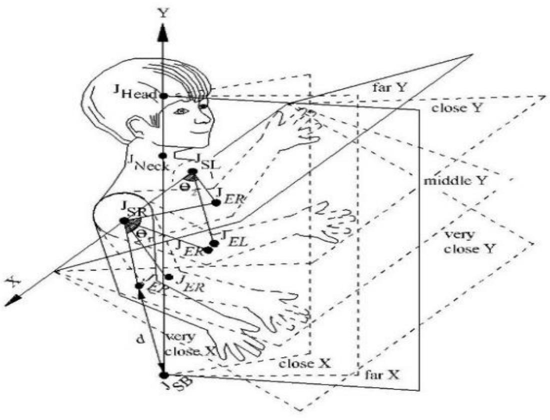


Figure 4. Local support of features in XZ and YZ projections

```
TSFS_RSR.addRule(new TagakiSugenoFuzzyRule(-4.4322 * the-
          ta4 + 641.34, new string[] { “All”, “very_closeY” }));
     TSFS_RSR.addRule(new TagakiSugenoFuzzyRule( -1.3333*
  theta4 + 190 , new string[] { “RightObtuse”, “med_closeY” }));
     TSFS_RSR.addRule(new TagakiSugenoFuzzyRule(-5.6129 *
   theta4 + 834.95, new string[] { “Obtuse”, “med_closeY” }));
          TSFS_RSR.addRule(new TagakiSugenoFuzzyRule(-74,
              new string[] { “RightObtuse4”, “med_closeY” }));
     TSFS_RSR.addRule(new TagakiSugenoFuzzyRule( -0.8333*
      theta4 + 115, new string[] { “RightObtuse”, “closeY” }));
     TSFS_RSR.addRule(new TagakiSugenoFuzzyRule(-2.6761 *
  theta4 + 394.57, new string[] { “RightObtuse3”, “closeY” }));
     TSFS_RSR.addRule(new TagakiSugenoFuzzyRule(-2.5037 *
             theta4 + 366.96, new string[] { “All”, “farY” }));
```

Figure 5. Kinect θ4 mapping to NAO RShoulderRoll angle

Decomposition is used also to handle anti-collision limitations due to potential shell collision at the head level. They are implemented by fuzzy rules as well. As it can be seen in Figure 6a), the Pitch motion range is limited according to the Yaw value. First we map the angle HeadYaw by T-S of zero order and then HeadPitch Min and Max according to the first order T-S functions presented in Figure 6b) and 6c).

Figures 6. b) and c) show how collision relations between HeadYaw and HeadPitch angles are approximated linearly in four semantically closed groups (forming the premise parameters), the trend lines (forming consequence parameters for each semantic group), their equations and squared errors. If the error is not acceptable, the first step is repeated until deriving premise parameters. Having premise and consequence parameters, we write fuzzy rules for linearly approximation of non-linearity. Two example fuzzy rules of zero and first-order T-S derived from Figure 6. c) are presented below.

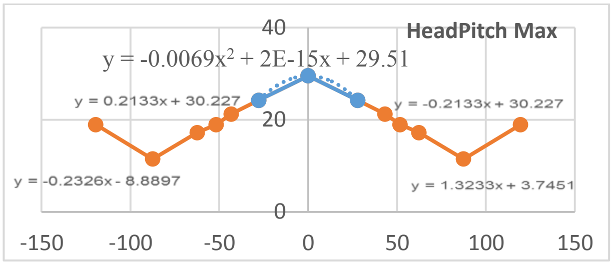
R_3 : If μ_{θ_1} is acute and μ_{θ_1} is right *then* $\theta_1 = 40$

R_4 : If μ_{θ_1} is acute and μ_{θ_1} is obtuse then $\theta_1 = 0.3225\,\theta_2$ - 45.787

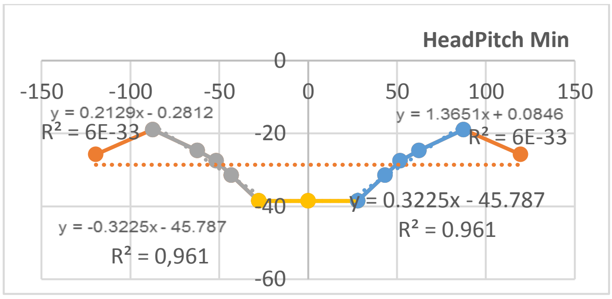
By using first-order T-S rather than zero order, we limit the movements and reduce the number of fuzzy sets used, respectively the number of fuzzy rules in the whole system.

HeadYaw	HeadPitch Min	HeadPitch Max
-119.52°	-25.73 °	18.91 °
-87.49 °	-18.91 °	11.46 °
and so on		

a) Anti-collision limitation due to Aldebaran Table [3]



b) Consequence Parameters



c) Consequence Parameters

Figure 6. Anti-collision limitations at the head level

The feasibility and real time operation of the proposed WKNF has been proven by videos in [1], Section Results> Games for motor and cognitive rehabilitation> Imitation. The overall latency is less than 100 milliseconds that is suggested for developers. The problems we aim to solve are for imitation of the whole body. In the future, we intend to incorporate in the fuzzy reasoning the NAO center of mass for robot stability.

CONCLUSION

The proposed wireless Kinect-NAO framework for teleoperation has been implemented and tested with children and adults. Its feasibility and usability have been proven by real experiments with required accuracy, view invariance and response in real time. Structured expressions of natural language as linguistic IF-THEN rules, flexible fuzzy sets and easy for implementation wireless connection to NAO middleware allow the framework to be tried by others without detailed knowledge for video processing, inverse Kinematic task in Robotics and fuzzy logic.

ACKNOWLEDGMENTS

This research is supported by the EEA grants, BG09 N D03-90/27.05.2015, Measure “Projects for Interinstitutional Cooperation”.

REFERENCES

1. http://iser.bas.bg/METEMSS/en/

2. http://www.microsoft.com/en-us/kinectforwindows

3. Aldebaran NAO humanoid robot, URL: https://www.ald.softbankrobotics.com/en/cool-robots/nao

4. Skeletal Joint Smoothing White Paper. URL: http://msdn.microsoft.com/en-us/library/jj131429

5. Mukherjee S., et all, Inverse kinematics of a NAO humanoid robot using Kinect to track and imitate human motion, Int. Conf. RACE 2015, 18-20 Feb, Chennai, India, 1-7.

6. Wenbai C. et all, Human's Gesture Recognition and Imitation Based on Robot NAO, Signal Processing, Image Processing and Pattern Recognition, Vol.8, N12, 2015, 259-270

7. Rodriguez I, et al. Humanizing NAO robot teleoperation using ROS, 14th IEEE-RAS Int. Conf. on Humanoid Robots, November 18-20, 2014. Madrid, Spain, 179-186

8. Takagi T., Sugeno M. Fuzzy Identification of Systems and Its Applications to Modelling and Control. IEEE Trans. Sys. Man Cyber. 1985, 15:116–132

9. Gamal T., NAO.NET. http://www.codeproject.com/Tips/

1002530/NAO-NET-Python-programs-to-NET