IRSTI 28.23.27 DOI: https://doi.org/10.26577/JMMCS.2022.v115.i3.011

E.B. Zhantileuov[1*], A. Aibatbek[1], A.M. Smaiyl[1], A. Albore[2], E.A. Aitmukhanbetova[1], Sh.M. Saimassayeva[1]
[1]Astana IT University, Kazakhstan, Astana
[2]IRT Saint Exupery, France, Toulouse
*e-mail: zhantileuov.eldiyar@Astanait.edu.kz

Multi-Agent Learning for the Inverse Kinematics of a Robotic Arm

This paper presents a solution to the inverse kinematics problem for robotic manipulator based on the Adaptive Multi-Agent System (AMAS) approach. In this research, multi-agent system is in charge of controlling a robot arm with four degrees of freedom (DOF) and two motorized wheels, giving appropriate commands, such as rotation angles and velocities, to reach the desired position and orientation of the end effector. The calculation of commands is directly related to the solving of forward and inverse kinematics. Before the learning process of AMOEBA, the rotational angles, θ values, are encoded into a single number N, this parameter is the desired value that we are going to predict in the predicting stage. During the learning phase, the Agnostic MOdEl Builder by self-Adaptation (AMOEBA) builds context agents, which has local models and is able to self-adapt. After the getting the predicted value, N_{pred}, it will be decoded back to get the set of rotational angles that is given to robot end effector. In addition, the robot with all its physical parameters is modeled and simulated in the Robot Operating System (ROS) environment

Key words: Forward kinematics, inverse kinematics, adaptive multi-agent system, agnostic model builder by self-adaptation.

Е.Б. Жантилеуов[1*], А. Айбатбек[1], А.М. Смайыл[1], А. Альборе[2], Э.А. Айтмұханбетова[1], Ш.М. Саймасаева[1]
[1] Astana IT University, Қазақстан, Астана қ.
[2]IRT Saint Exupery, Франция, Тулуза қ.
*e-mail: zhantileuov.eldiyar@Astanait.edu.kz

Робот қолының кері кинематикасы үшін мультиагенттік оқыту

Бұл жұмыста бейімделетін көп агенттік жүйе (Adaptive Multi-Agent System) тәсіліне негізделген роботтық манипуляторға арналған кері кинематика мәселесінің шешімі ұсынылады. Бұл зерттеуде мульти-агенттік жүйе төрт еркіндік дәрежесі (DOF) бар робот қолы мен екі дөңгелегін басқаруға жауапты. Роботтық қажетті позиция және бағдарына жетуі үшін, оның қолы мен дөңгелектеріне айналу бұрышы мен жылдамдық тәрізді тиісті командалар беріледі. Командаларды есептеу тура және кері кинематика есебін шешумен тікелей байланысты. AMOEBA-ның үйрену кезеңіне дейін θ айналу бұрыштары бір N санына шифрланады. Бұл параметр болжау кезеңіндегі біздің болжам жасайтын негізгі мән болып табылады. Үйрену кезеңінде Agnostic MODEl Builder by self-adaptation (AMOEBA) жергілікті үлгілері бар және өзін-өзі бейімдей алатын контекстік агенттерді құрады. Болжамды мән, N_{pred}, есептелініп алынғаннан кейін, айналу бұрыштарының жиынтығын алу үшін кері бағытта шифр ашылады. Бұл жиынтық роботтық атқарушы механизмі, яғни робот қолының саусақ ұшы, қажетті позиция және бағдарға жетуі үшін төрт еркіндік дәрежелі қолы мен екі дөңгелегіне команда ретінде беріледі. Сонымен қатар, робот өзінің барлық физикалық параметрлерімен Robot Operating System (ROS) ортасында модельденеді және имитацияланады.
Түйін сөздер: Кинематика, кері кинематика, адаптивті көп агенттік жүйе, өзін-өзі бейімдеу, агностикалық модель.

Е.Б. Жантилеуов[1*], А. Айбатбек[1], А.М. Смайыл[1], А. Альборе[2], Э.А. Айтмуханбетова[1], Ш.М. Саймасаева[1]
[1] Astana IT University, Казахстан, г. Астана
[2]IRT Saint Exupery, Франция, г. Тулуза
[*]e-mail: zhantileuov.eldiyar@Astanait.edu.kz

Мультиагентное обучение для обратной кинематики роботизированной руки

В данной статье представлено решение обратной задачи кинематики для робота-манипулятора на основе подхода Adaptive Multi-Agent System (AMAS). В этом исследовании мультиагентная система отвечает за управление манипулятором робота с четырьмя степенями свободы (DOF) и двумя моторизованными колесами, давая соответствующие команды, такие как углы поворота и скорости, для достижения желаемого положения и ориентации исполнительного механизма, то есть концевого эффектора. Расчет команд напрямую связан с решением прямой и обратной кинематики. На этапе обучения Agnostic MOdEL Builder путем самостоятельной адаптации (AMOEBA) создает агенты контекста, которые имеют локальные модели и способны к самостоятельной адаптации. Перед процессом обучения AMOEBA, углы поворота, θ значения, кодируются в одно число N, этот параметр является желаемым значением, которое мы собираемся предсказать на этапе прогнозирования. После получения предсказанного значения N_{pred}, оно будет декодировано обратно, чтобы получить набор углов поворота, заданный концевому исполнительному механизму робота. Кроме того, робот со всеми его физическими параметрами моделируется и симулируется в среде Robot Operating System (ROS).
Ключевые слова: Прямая кинематика, обратная кинематика, адаптивная мультиагентная система, независимый построитель моделей путем самостоятельной адаптации.

1 Introduction

Nowadays the study and development of intelligent robots are becoming an essential part of robotics. Many methods and approaches are aimed at making the robots fully automated and independent of external impacts, such as neural networks and multi agent systems. Major attention is paid to the motion of the robot, which, in turn, involves the study of kinematics. The general objective of this research is to reach the desired point or target with end-effector of robot with precise accuracy. In order to reach the goal, both forward and inverse kinematic problems must be solved. The forward kinematics (FK) involves determining the position and orientation of the robotic end-effector by giving values for each individual joint of robotic manipulator. Vice versa, by knowing the position and orientation of the end effector, the inverse kinematics (IK) is in charge with determination of values that must be set to the joints, in other words, inverse kinematics is the inverse problem of forward kinematics. In comparison with forward and the inverse kinematics, the solution of inverse kinematics is much more complicated. The FK can be easily solved by performing linear algebraic operations on homogeneous transformation matrices and has a unique solution. However, due to the complex IK equations, which is strongly nonlinear, there is no single solution for IK. As we mentioned, the IK is the main issue of robotics, and several methods are proposed for its solution [1]. Many approaches to this problem lie on the analytical, algebraic, or iterative methods, which give approximate results. Recently, much attention has been paid to artificial networks and self-adaptive multi-agent systems. The controlling of the robotic arm is considered as real-world complex problem and it cannot be solved by predefined model and needs learning and self-adaptation. 'Multi-agent systems are particularly suitable to design and implement self-organizing systems' [2]. In this paper, Self-Adaptive Context Learning

(SACL) recurrent pattern is applied to our problem. It consists of two mechanisms: Adaptive mechanism, which perceives information from the environment and dynamically builds a model describing the current context and its transformation Exploitation mechanism, which decides what actions to perform over the environment [2].

For building a dynamic model in adaptive mechanism, Agnostic MOdEl Builder by self-Adaptation (AMOEBA) is used. AMOEBA is based on AMAS approach. In order to be able to build a model, AMOEBA must learn on data provided by simulation or FK problem, which makes it supervised learning. In the final application, the multi-agent system will be integrated with machine learning, the function of which is to process an image, taken from the robot's camera, identify the target point and compute its distance and position relative to the camera. The integration of a machine-learning application with multi-agent system is another key feature of the project. The position of the button and the robot with all its physical parameters are simulated in ROS environment. Motivating Example. Figure 1 shows the real problem of the work. Consider a robot inside an elevator, the starting position and orientation of which are known. The robot's camera, which is attached on the end-effector, takes a picture of buttons in the elevator and the robot needs to press the desired button. Once the picture of elevator buttons is taken, the machine learning software identifies the desired button and calculates its position $(x,\ y,\ z)$ with respect to the camera. The coordinates of the button are then sent to the multi-agent system. Multi-agent system is responsible to control the 6 servo motors: 4 for robot arm and 2 for wheels. Taking the positions received from ML as input data, the multi-agent system solves IK problem to get rotation angles for each joint, $\theta_0,\ \theta_1,\ \theta_2,\ \theta_3$. The servo motors are given an angle setpoints and they rotate and maintain to reach this setpoint:

$$\text{CAMERA} \xrightarrow{image} \text{coord} \xrightarrow{x,y,z} \text{AMAS} \xrightarrow{\theta_0,\theta_1,\theta_2,\theta_3} \text{Robot Arm.}$$

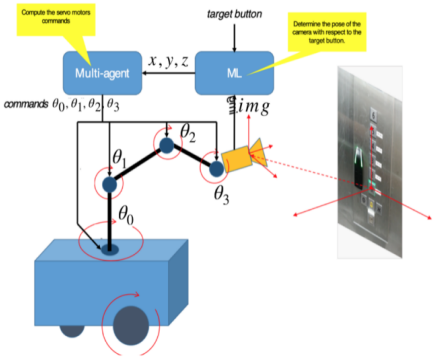

Figure 1: The robot in an elevator, identifying the desired target and tries to reach it

Figure 2 contains a snapshot of the real robot with four degrees of freedom (DOF) arm which is placed on the platform. The platform has two motorized wheels and one castor wheel.

Figure 2: The picture of the real robot with 4 DOF named TwIRTee

2 Simulating Physical Model of Robot under Robot Operating System (ROS)

This section describes the simulation model of the TwIRTee robot under ROS/Gazebo. This model includes the robot chassis with its two motorized wheels, the robot arm, and the Light Identification Detection and Ranging (LIDAR). It gives the procedure to setup the environment and to interact with the simulation via the com-mand line and programmatically. The two wheels are identical, so they are modeled using a macro with a parameter, "tY"that

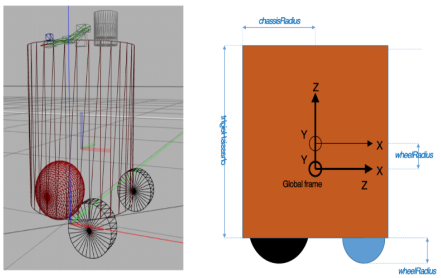

Figure 3: The local frame for the chassis definition

gives the translation of the wheel with respect to the Y axis. Each wheel is drawn in a local frame that is obtained by a rotation of $\frac{\pi}{2}$ radians along the Y and Z axis with respect to the joint reference frame (see Figure 3). The robot is a set of links (such as the chassis described previously) and joints. Let's take the example of the robot arm that is fitted on top of the robot, as show on Figure 3, with a close-up view on Figure 4.

The arm is composed of: 4 servo motors (the green boxes): link0, link1, link3 and link5; two sets of "bars"(brown colored): link2 and link4; camera (in blue); "finger"(in red, at the tip of the arm):
"link1_joint"joints "link1"and "link2"with a "revolute"joint;
"link3_joint"joints "link2"and "link3"with a "revolute"joint;
"link4_joint"joints "link3"and "link4"with a "fixed"joint.
In the model, the camera is represented by a simple blue box (see Figure 5).

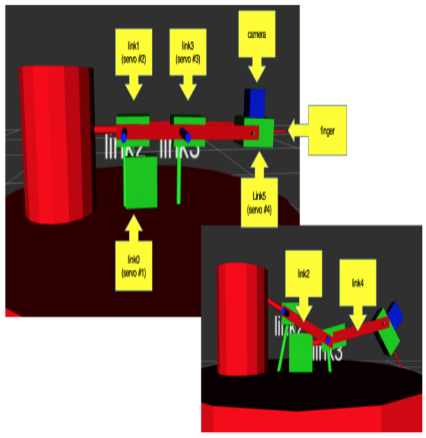

Figure 4: The robotic arm closed view

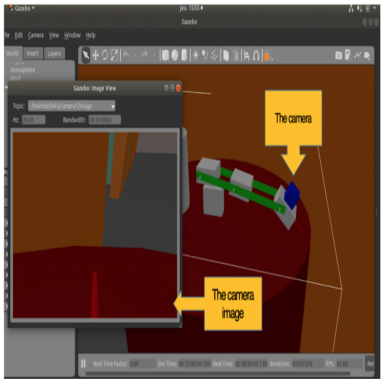

Figure 5: The illustration of the camera attached to the end-effector (blue box)

This environment allows the complete dynamics of the system to be simulated, in-cluding the effect of inertia: the simulator receives the angles for each joint and com-putes the position of the arm and camera. Figure 6 shows the general idea of integrating the machine learning part with the multi-agent system in ROS environment. There are many related works with image processing and object detection and ML for image processing is quit out of this paper. The main task is to tackle with multi-agent system, to make the multi-agent system learn and self-adapt with precise accuracy.

3 Forward Kinematics

In robotics, forward kinematics is responsible for determining the final coordinates and the direction of the end-effector relative to the global coordinate space. Let's consider that the initial position and orientation of each servo motor is known. Ho-mogeneous transformation matrices with a dimension of 4x4 will be constructed from the base frame to the end effector frame [3]. These matrices consist of a 3x3 rotation matrix, that describes the orientation of

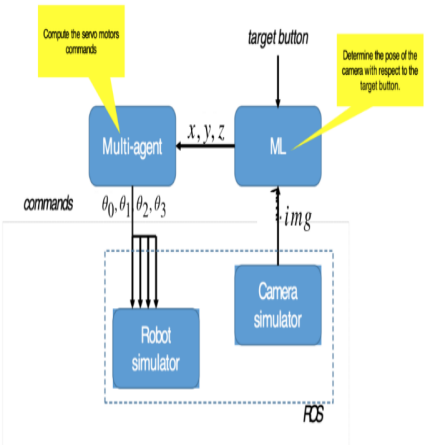

Figure 6: The illustration of integration of ML and AMAS in ROS environment

joints and their behaviors, and trans-lation vector. Further, linear algebra operations will be performed on matrices to ob-tain FK results. In this section, more detailed solutions are provided for the robot arm.

3.1 Kinematics for 4 DOF Robotic Arm

In our case, the robotic arm has 4 degrees of freedom (DOF). The robot is articulated vertically with 4 joints. It has a stationary base, shoulder, elbow and wrist, where the base joint rotates around the z-axis and the other three rotate around the y-axis. The position of joints is represented in the three-dimensional Cartesian coordinate system and a local reference frame is assigned to each joint. The coordinate frame assign-ment is shown in Figure 7. In addition, it is necessary to assign a global coordinate frame to the base of the robot [4] (see Figure 8). The servo motors in three-dimensional space can have movements of

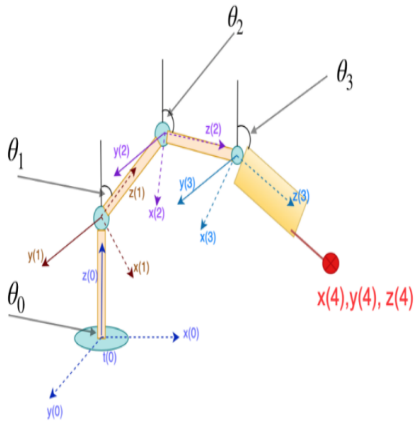

Figure 7: The coordinate frame assignment of robotic arm

rotation and translation. The homogeneous transformation matrix (H.T.M) with dimension of 4x4 is constructed separately for each joint to describe its position and orientation relative to the world coordinate system. The H.T.M is composed of 3x3 rotation matrix and 3x1 translation vector:

$$\left| \begin{array}{cccc} . & . & . & . \\ . & R_{3\times3} & . & t_{3\times1} \\ . & . & . & . \\ 0 & 0 & 0 & 1 \end{array} \right| \tag{1}$$

– Rotational matrix describes the rotation of joints in Euclidean space. The rotation is done about $z,\ y\ and\ x$ axes through a counterclockwise angle θ. The axis rotation matrices for a rotation about $z,\ y\ and\ x$ axes given, respectively [5]:

$$R_z(\theta) = \left| \begin{array}{ccc} 1 & 0 & 0 \\ 0 & \cos(\theta) & \sin(\theta) \\ 0 & -\sin(\theta) & \cos(\theta) \end{array} \right| \tag{2}$$

$$R_y(\theta) = \left| \begin{array}{ccc} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{array} \right| \tag{3}$$

$$R_x(\theta) = \left| \begin{array}{ccc} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{array} \right| \tag{4}$$

– Translation or displacement vector shows the location of each joint in the base frame. The final translation vector is the answer of the FK problem. In order to obtain the final translational vector, the transformation matrices of each joint are multiplied. The sequence of multiplication is important, as it results in a trajectory generation step [5].

The transformation matrices of each joint are represented as ${}^{i}_{j}T$ (see Fig.8.):

1. Transformation matrix of base joint, rotates about z-axis:

$${}^{0}_{1}T = \left| \begin{array}{cccc} 1 & 0 & 0 & x_0 \\ 0 & \cos(\theta_0) & \sin(\theta_0) & y_0 \\ 0 & -\sin(\theta_0) & \cos(\theta_0) & z_0 \\ 0 & 0 & 0 & 1 \end{array} \right| \tag{5}$$

2. Transformation matrix of shoulder joint, rotates about y-axis:

$${}^{1}_{2}T = \left| \begin{array}{cccc} \cos(\theta_1) & 0 & \sin(\theta_1) & x_1 \\ 0 & 1 & 0 & y_1 \\ -\sin(\theta_1) & 0 & \cos(\theta_1) & z_1 \\ 0 & 0 & 0 & 1 \end{array} \right| \tag{6}$$

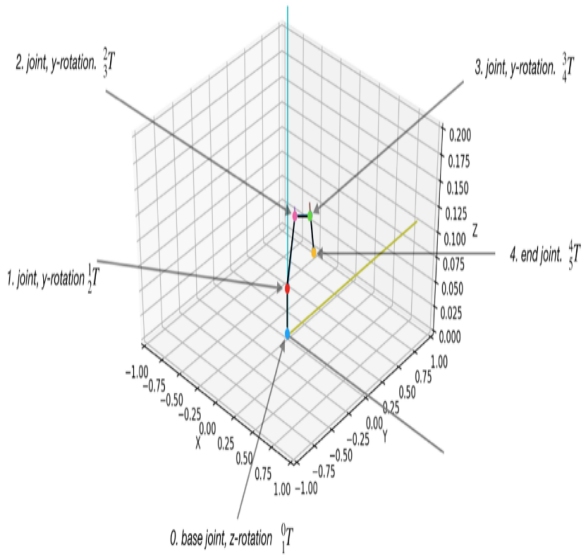

Figure 8: The coordinate frame assignment of robotic arm in world space

3. Transformation matrix of elbow joint, rotates about y-axis:

$$ {}^{2}_{3}T = \begin{vmatrix} \cos(\theta_2) & 0 & \sin(\theta_2) & x_2 \\ 0 & 1 & 0 & y_2 \\ -\sin(\theta_2) & 0 & \cos(\theta_2) & z_2 \\ 0 & 0 & 0 & 1 \end{vmatrix} \tag{7} $$

4. Transformation matrix of wrist joint, rotates about y-axis:

$$ {}^{3}_{4}T = \begin{vmatrix} \cos(\theta_3) & 0 & \sin(\theta_3) & x_3 \\ 0 & 1 & 0 & y_3 \\ -\sin(\theta_3) & 0 & \cos(\theta_3) & z_3 \\ 0 & 0 & 0 & 1 \end{vmatrix} \tag{8} $$

5. Transformation matrix of end joint:

$$ {}^{4}_{5}T = \begin{vmatrix} 1 & 0 & 0 & x_4 \\ 0 & 1 & 0 & y_4 \\ 0 & 0 & 1 & z_4 \\ 0 & 0 & 0 & 1 \end{vmatrix} \tag{9} $$

Finally, the desired transformation matrix is obtained by multiplying all ${}^{i}_{j}T$ matrices:

$$ {}^{0}_{5}T = {}^{0}_{1}T \cdot {}^{1}_{2}T \cdot {}^{2}_{3}T \cdot {}^{3}_{4}T \cdot {}^{4}_{5}T \tag{10} $$

where ${}^0_5T$ has the form of:

$$\begin{vmatrix} . & . & . & x \\ . & R_{3\times3} & . & y \\ . & . & . & z \\ 0 & 0 & 0 & 1 \end{vmatrix}. \tag{11}$$

The $(x,\ y,\ z)$ is the answer to the FK problem. The $(x_4,\ y_4,\ z_4)$ is the position of the end-effector in the local coordinate space (see Figure 7) and $(x,\ y,\ z)$ is the position of the end-effector on the world system, in another words, the coordinates of the end-effector are translated to the global coordinate system.

The Implementation of FK on 4 DOF Robotic Arm Implementing the FK to determine the final position and orientation of the end-effector is done in Python.

Suppose the arm of the robot is raised up initially. The rotation angles are given to each servo motor, i.e. the rotation angle setpoints, $(\theta_0,\ \theta_1,\ \theta_2,\ \theta_3)$, are sent to the base, shoulder, elbow and wrist. Depending on the given angles, the motors begin to rotate. The final location of the end-effector is determined by the translation of the coordinate from the local coordinate system to the global one. Input data is angular setpoint, the output is the coordinates of the end-effector on the global system:

$$\theta_0,\ \theta_1,\ \theta_2,\ \theta_3 \xrightarrow{FK\_4Dof} x,\ y,\ z.$$

The result of the problem is illustrated in Python on Figure 9. Initially, the robot arm is

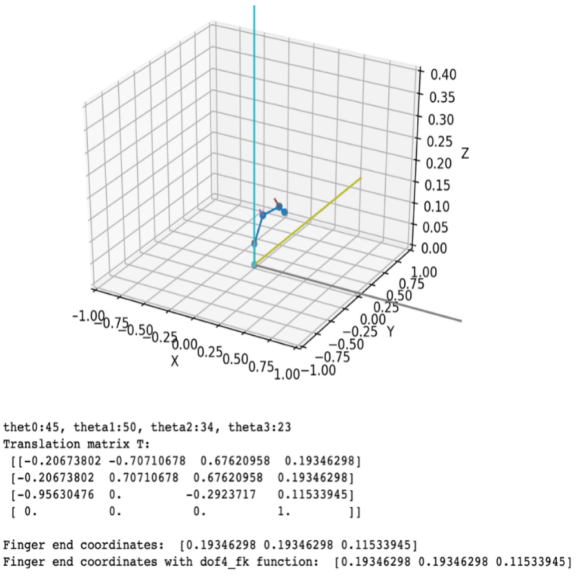

Figure 9: The result of the example to check the correctness of implementation FK.

raised up and $\theta_0 = 45,\ \theta_1 = 50,\ \theta_2 = 34,\ \theta_3 = 23$ is given to the motors. The dark blue curve is the final position and orientation of the arm manipulators. The final position of the end-effector computed by FK is (0.19346298, 0.19346298, 0.11533945).

4 Inverse Kinematics

Summing up the previous section, we can say that solving forward kinematics for robotic manipulators is a fairly simple task, only linear algebra operations are performed on matrices to determine the final position and orientation of the end-effector. The problem has one and only one solution. However, when the final position and orientation of the manipulators is initially given, and the task is to find the rotation angles for each joint, (θ_0, θ_1, θ_2, θ_3), the problem becomes non-linear and complex. This kind of task in robotics is called Inverse Kinematics Problem. There are many approaches to solving inverse kinematics problem, e.g. analytical solution, numerical methods, artificial neural networks and self-adaptive multi-agent systems. In this paper, we propose Adaptive Multi-Agent System based solution for solving IK problem.

5 Adaptive Multi-Agent System

To solve IK problem, we need to prepare a model which is responsible to predict the revise the degrees of liberty and "rotation time". However, due to the complexity of the problem, it is difficult and expensive to solve using a predefined model; instead, we will use several agents, an autonomous entities, responsible for predicting the result. A system where the agents are plugged-in should be able to adapt to the environment and learn independently. The Adaptive Multi-Agent Systems (AMAS) approach has been applied to designed and developed self-adaptive multi-agent system. This approach aims at solving problems in dynamic non-linear environments by a bottom-up design of cooperative agents, where cooperation is the engine of the self-organization process [7].

6 The Self-Adaptive Context Learning Pattern

Our self-adaptive system is connected with a dynamic environment by a cycle of observations. The main task of system is to receive the observations coming from the environment and find a proper actions for the current state of inputs, which, in turn, is called the context [8]. This is a context mapping problem. The Self-Adaptive Context Learning (SACL) is recurrent pattern, based on the AMAS approach, the key feature of which is to solve the context-mapping sub-problem. It is composed of two mechanisms, that interacts with the environment:

- Adaptation mechanism, is dynamically building a model, that describes the current context and possible actions in it. [2, 8] It is related to the learning phase of the system and its changes.
- Exploitation mechanism, is in charge with the selecting the most appropriate action in the current context.

7 AMOEBA: Agnostic MOdEl Builder by Self-Adaptation

The building of the model in adaptation mechanism is performed by using Agnostic MOdEl Builder by Self-Adaptation (AMOEBA), based on the AMAS approach. The model explains

the interaction that occurs between the mechanism of exploitation and the environment [2]. The model receives a set of input data, we call it percepts, and produces one output. We call the obtained result as prediction and the actual, correct result is called oracle.

There are two types of agents in AMOEBA [9]:

- Percept agents are responsible for the perceiving information from the environment.
- Context agents are in charge of determination the context, where a specific output would be a good one.

AMOEBA learning phase is done by building the context agents. Each context agent has its own validity range and local model. The validity range of the context agent is the interval, where a specific output will be relevant [9]. If the received value of the percept agent is included in the validity range interval, we say the validity range is valid for this percept. The context agents have rectangular shape in two-dimension space (see Figure 10). The local model is built separately for each context agent. When the validity range of the

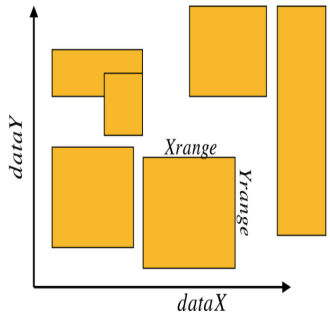

Figure 10: The context agents in AMOEBA

context agent is valid for the current perceived value, the output is calculated by using the local model of that context agent. In this paper, the linear regression is used as a model. The linear regression function computed using a set of points [9]:

$$\sum_{n=1}^{p} x_n v_n + a \tag{12}$$

where p is the number of percepts, x_n and v_n are the coefficients, a is the real number.
The creation of the context agent, the self-organization, the changing of the validity ranges, the changing local model and the destroying itself is deeply described in reference [9].
Working Principle of AMEOBA. At first, AMOEBA must learn from examples with the correct outputs. This approach of learning is called supervised learning. Once, the AMOEBA is learned, it starts to predict the result for a new inputs.

Let's look at the illustration taken from reference [9]:

1. During the learning phase, AMOEBA uses incoming data to adapt and improve itself. The specific data set with the correct result is given to AMOEBA. However, at the

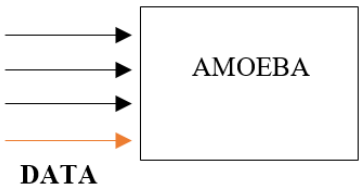

Figure 11: Learning phase of AMOEBA, with the given oracle (red arrow)

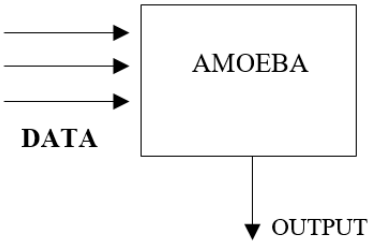

Figure 12: The exploitation step of AMOEBA, without labeled data

beginning, the oracle, actual result, is "hidden" from AMOEBA. The valid context agent tries to predict the output, and checks the predicted value with the oracle. If it was wrong, it adapts and improves itself by reducing its validity range or changing the local model (see Figure 11).

2. During the exploitation step, AMOEBA receives a set of data without an oracle. Based on the previous knowledge it provides an output (see Figure 12).

AMOEBA for the Inverse Kinematics Problem. In IK problem for the robot arm, the input data are the final coordinates of the end-effector, $(x,\ y,\ z)$, remember that in practice these coordinates are taken from machine learning software. The output is a set of rotation angles for each servo motor, $(\theta_0,\ \theta_1,\ \theta_2,\ \theta_3)$. Then the servo motors execute the given commands to achieve the $(x,\ y,\ z)$ target position. This means that we must predict 4 parameters for the robot arm. However, AMOEBA learns to predict only one parameter at the time. So, using four independent AMOEBAs to perform the learning of each parameter can give physically unreachable commands to the robot arm because the correlation between each parameter would be lost; e.g. to reach point (x, y, z) the arm has many ways to reach desired point by varying its angles, a single solution is given by an arm configuration expressing four angles which depend from each other in each configuration. If the learning process for each angle is independent, the prediction for the angle will be non-correlated to the one of the other angles, resulting in an "impossible" arm configuration. Therefore, in order to preserve the correlation between the joint's positions of the robotic manipulators, we decided to encode the four angles $(\theta_0, \theta_1, \theta_2, \theta_3)$ to one single number N. This number N is used as an oracle in the learning process. Then, when the (x, y, z) coordinate is provided to the trained system, the number encoding the joints angles is given as output, and decoded for the final application.

In order to AMOEBA to predict values, the system need to be trained. Therefore, a training data set should be provided.

Training Data for AMOEBA. The learning data for AMOEBA is built in Python programming language. Several training sets of 100, 1000 and 5000 examples respectively, are randomly generated in different files. The angle values uniformly cover the fallowing ranges:

$$\theta_0 \in (0;\ \pi)\,,\ \theta_1 \in \left(0;\ \frac{\pi}{2}\right),\ \theta_2 \in \left(0;\ \frac{\pi}{2}\right),\ \theta_3 \in \left(0;\ \frac{\pi}{2}\right).$$

For each example, the final position and orientation of the end-effector is calculated by solving FK problem (see Figure 13). The result of FK problem is exact and stored in vector $(x,\ y,\ z)^{\mathrm{T}}$ form.

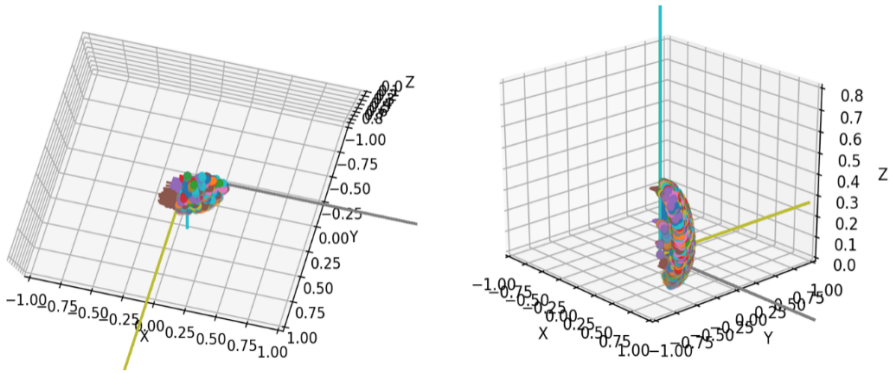

Figure 13: The resulting positions of the end-effector for 5000 randomly generated set of joint angle values

Finally, each example used in the training file is a row in a table consisting in $\theta_0,\ \theta_1,\ \theta_2,\ \theta_3,\ x,\ y,\ z$ parameters, and the respective encoding of the joint positions, given that the learning ability of AMOEBA is limited by only one parameter. This means that, we feed AMOEBA with data, that has the correct answers or oracles.
Encoding and Decoding of θ *Values.* The process of encoding θ values into a single number, N, occurs before the learning process of AMOEBA. The number N is used as an oracle at the learning stage (red colored) and at the predicting stage, this is the value that we aim to predict. The value of $N_{predict}$ is then decoded to retrieve $\theta_0 pred,\ \theta_1 pred,\ \theta_2 pred$ and $\theta_3 pred$ (see Figure14).

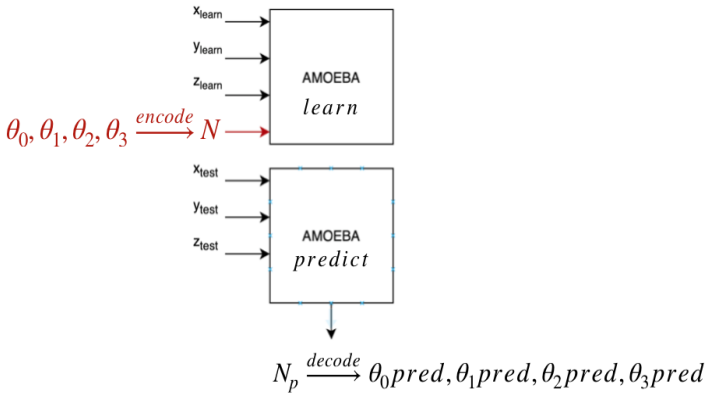

Figure 14: The role of encoding/decoding in the learning and predicting stages of AMOEBA.

Let's see how θ values are encoded. The movement of joints are limited within the following

ranges:

$$\theta_0 \in (0;\, \pi)\,,\, \theta_1 \in \left(0;\, \frac{\pi}{2}\right),\, \theta_2 \in \left(0;\, \frac{\pi}{2}\right),\, \theta_3 \in \left(0;\, \frac{\pi}{2}\right),$$

and the maximum value that angle can assume is 180^0. This value, incremented by 1, is called the base ($B = 181$). Finally, the value N is calculated:

$$N = \theta_0 \times B^3) + (\theta_1 \times B^2) + (\theta_2 \times B^1) + (\theta_3 \times B^0) \tag{13}$$

To decode N, we divide it by base B. The value in remainder is θ_3. In order to get θ_2, θ_1 and θ_0, the division process is repeated, but instead of N, quotient of previous division is used.
Example 1. Let's encode and decode the set of angles:

θ_0=45^0;θ_1=23^0; θ_2= 54^0;θ_3=89^0

N= $\left(45\times181^3\right)$ + $\left(23\times181^2\right)$ + $\left(54\times181^1\right)$ + $\left(89\times181^0\right)$ = 267601711

The four values of θ are encoded in one N.
The decoding of N :

267601711÷181 = 1478462 (*remainder* 89)

1478462÷181 = 8168 (*remainder* 54)

8168÷181 = 45 (remainder23)

45÷181= 0 (remainder45)

The values in remainders are our angles, which we encode earlier.

Returning to our training data, let's encode all the joint angles. Table 1 represents several lines from the real dataset.

Table 1. A training data for AMOEBA

$\boldsymbol{\theta_0}$	$\boldsymbol{\theta_1}$	$\boldsymbol{\theta_2}$	$\boldsymbol{\theta_3}$	$\boldsymbol{x}$	$\boldsymbol{y}$	$\boldsymbol{z}$	$\boldsymbol{N}$
7	12	78	19	0.216441	0.026575	0.13153	5207239
72	18	36	74	0.059224	0.182274	0.20249	52637114
62	27	11	60	0.087914	0.165342	0.25034	45417750
53	48	83	72	0.101095	0.134158	-0.027521	39033342
...	...	...	...	...	...	...	...
19	83	53	48	0.189071	0.065102	-0.095592	14528118

Now instead of fourfold training for each θ, AMOEBA will be trained once on the values of N.

Learning Phase of AMOEBA. In the problem of inverse kinematics for the robot arm, AMOEBA starts learning by mapping ($x,\ y,\ z$) into cartesian plane. Note that: the oracle is N. For each point, AMOEBA randomly produces a value N_{pred}. If this value is closer to the oracle, the validity range of context agent expands, and vice versa, if the difference between the exact value of N and the predicted value of N is large, the range becomes smaller. If

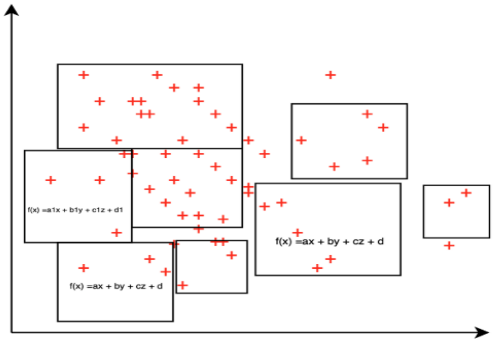

Figure 15: The illustration of context agents (red crosses are percept agents; the rectangles are context agents). Each.

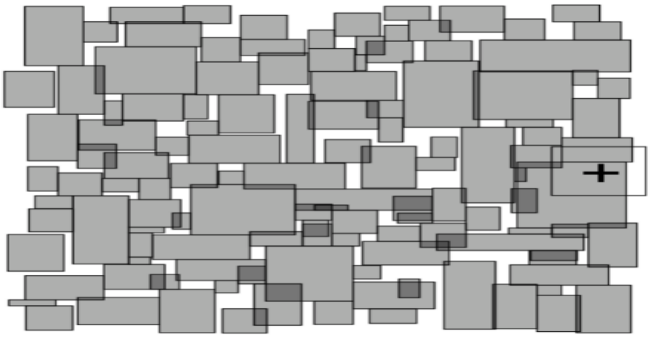

Figure 16: 2D visualization of learning AMOEBA in JAVA.

the validity range of context agent is too small, AMOEBA decides that it is useless and the context agent will be self-destroyed. Context agents with their local regression models are illustrated on Figure 15.

Validation Phase. To estimate how well AMOEBA was trained, we need to provide a testing dataset. Just like in training dataset, this file consists of 100 lines of $\theta_0,\ \theta_1,\ \theta_2,\ \theta_3,\ x,\ y,\ z,\ N$ *values. However*, at this stage we will use the oracle only to calculate model error. This means that the input for AMOEBA is only $x,\ y,\ z$, remember that at the learning stage, the input was $x,\ y,\ z$ and oracle N. Based on previous knowledge, AMOEBA predicts the value of N, the output is N_{pred}. This output is then decoded to get $\theta_0 pred,\ \theta_1 pred,\ \theta_2 pred$ and $\theta_3 pred$. Next, we simply solve FK problem for predicted joint angles and obtain the predicted coordinates of the end-effector, $(x_{pred},\ y_{pred},\ z_{pred})$. These steps are described in the following scheme:

$$x,\ y,\ z \overset{input}{\longrightarrow} AMOEBA \overset{predic}{\longrightarrow} N_{pred} \overset{decode}{\longrightarrow} \theta_0 pred,\ \theta_1 pred,\ \theta_2 pred,\ \theta_3 pred$$

$$\overset{solve\ FK}{\longrightarrow} x_{pred},\ y_{pred},\ z_{pred}.$$

The performance of AMOEBA was determined based on the Euclidean distance of two points and the mean squared error (MSE) between the predicted output and the expected

output.

8 Experimental results

First of all, I generated 3 training datasets for AMOEBA with 100, 1000 and 5000 rows of θ values. To find the corresponding localizations, the problem of forward kinematics has been solved and for each row, the values of θ were encoded into a single N (see Tab.2). These data were then transmitted to AMOEBA, so that it could learn. After each training with the data of different sizes, another testing dataset is given, to check the correctness the model.

Table 2. An example of learning data for AMOEBA

θ_0	θ_1	θ_2	θ_3	x	y	z	N
7	16	19	19	0.15	0.02	0.3	5234329
81	65	63	54	0.03	0.2	-0.05	59581224
78	9	68	17	0.04	0.21	0.18	56941037
30	21	13	12	0.13	0.08	0.3	22041282
...	...	...	...	...	...	...	...
55	2	82	87	0.09	0.13	0.116	40118667

Once AMOEBA is trained, the validation phase is conducted. Tables 3, 4 and 5 show the results of validation stage after training with 100, 1000 and 5000 data rows, respectively.

Table 3. After training AMOEBA with 100 rows of data, the mean Euclidean distance is 0.33 and the $\sum MSE = 0.03$.

x	y	z	N_{pred}	$\theta_0 p$	$\theta_1 p$	$\theta_2 p$	$\theta_3 p$	x_p	y_p	z_p	ED	MSE
0.15	0.02	0.3	9720459	13	30	5	9	0.17	0.04	0.3	0.02	0.001
0.03	0.2	-0.05	55375708	75	86	45	58	0.05	0.2	-0.1	0.05	0.002
0.04	0.21	0.18	59999713	82	27	33	43	0.03	0.22	0.2	0.03	0.001
0.13	0.08	0.3	27920717	38	27	0	17	0.12	0.09	0.31	0.02	0.00
...	...	...	...	...	...	...	...	...	...	...	...	...
0.09	0.13	0.116	37979806	52	8	77	76	0.11	0.14	0.11	0.02	0.01

Table 4. After training AMOEBA with 1000 rows of data, the mean Euclidean distance is 0.12 and the $\sum MSE = 2.66$.

x	y	z	N_{pred}	$\theta_0 p$	$\theta_1 p$	$\theta_2 p$	$\theta_3 p$	x_p	y_p	z_p	ED	MSE
0.15	0.02	0.3	5250410	7	18	17	80	0.16	0.02	0.25	0.05	0.003
0.03	0.2	-0.05	59594295	81	67	28	75	0.04	0.25	0.03	0.09	0.009
0.04	0.21	0.18	57170504	78	38	7	74	0.04	0.21	0.21	0.03	0.001
0.13	0.08	0.3	21849462	29	87	41	72	0.18	0.10	-0.1	0.39	0.147
...	...	...	...	...	...	...	...	...	...	...	...	...
0.09	0.13	0.116	40903374	56	9	71	84	0.10	0.15	0.13	0.02	0.001

Table 5. After training AMOEBA with 5000 rows of data, the mean Euclidean distance is 0.1 and the $\sum MSE = 3.12$.

x	y	z	N_{pred}	$\theta_0 p$	$\theta_1 p$	$\theta_2 p$	$\theta_3 p$	x_p	y_p	z_p	ED	MSE
0.15	0.02	0.3	5229175	7	15	51	85	0.19	0.02	0.16	0.14	0.021
0.03	0.2	-0.05	59609975	81	69	23	5	0.05	0.29	0.07	0.15	0.023
0.04	0.21	0.18	56898316	78	4	43	46	0.03	0.16	0.25	0.08	0.008
0.13	0.08	0.3	22157307	30	35	42	27	0.22	0.13	0.15	0.18	0.033
...	...	...	...	...	...	...	...	...	...	...	...	...
0.09	0.13	0.116	40903374	56	9	71	84	0.10	0.15	0.13	0.02	0.001

With the help of testing data, we can compute the Euclidean distance between two points, (x, y, z) and $(x_{pred}, y_{pred}, z_{pred})$. With an increase in the training data, the mean Euclidean distance decreased, and the sum of the mean squared error increased (see Figure 17). The explanation for this is closely related to the number of context agents. When we try to train AMOEBA with more data, it also tries to build a perfect model. Thus, it breaks down the initial context agents into several small ones. When we have more context agents than necessary, our model becomes overfitted. On the other hand, we can notice that the values of

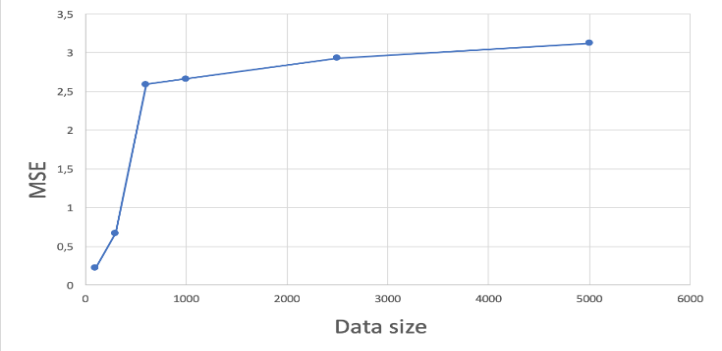

Figure 17: The graph of mean squared error of different data size.

N_{pred} are approximated to the real values of N and that AMOEBA always perfectly coincides with the first angle. So, I found that the order of θ values at the encoding stage is highly important, since when encoding, the first value is multiplied by the highest base accordance with equation (12).

Returning to the problem, at the encoding stage, we need to encode so that each θ is occurred first in order (see Figure 28).

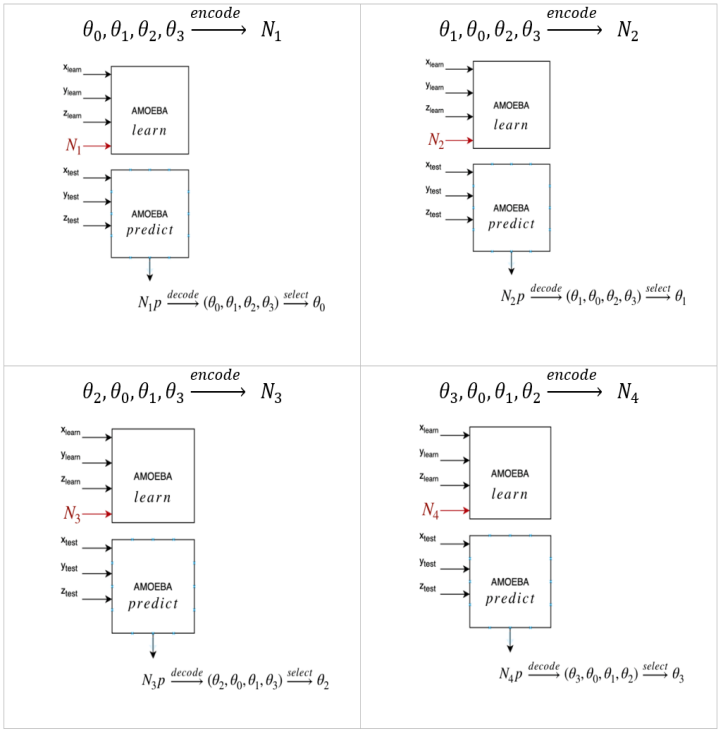

Figure 18: At the encoding step, we encode so that each θ will be first in order.

For each N, we create 4 independent AMOEBAs. Note that in this case all links and relations will be preserved between θ parameters. Further, all other steps will be the same for this learning. Only, at predicting stage, we select the better values of θ from each independent learning. Table 6 shows the testing phase results after training AMOEBA with 5000 data rows.

Table 6. The result of validation phase, after training AMOEBA with 5000 lines of data. To obtain this results, learning is done independently of each other with the oracle N.

x	y	z	$\theta_0 p$	$\theta_1 p$	$\theta_2 p$	$\theta_3 p$	x_p	y_p	z_p	ED	MSE
0.08	0.18	-0.05	67	50	69	31	0.09	0.21	0.00	0.06	0.004
0.04	0.17	-0.11	78	85	56	65	0.04	0.17	-0.1	0.01	0.000
0.20	0.16	-0.04	39	69	26	37	0.22	0.18	0.04	0.09	0.007
0.18	0.11	0.19	31	34	18	56	0.19	0.11	0.21	0.03	0.000
...	...	...	...	...	...	...	...	...	...	...	...
0.27	0.10	0.10	56	9	71	84	0.15	0.06	0.29	0.23	0.052

The result of latter method is pretty impressive: after 5000 learning, $\sum MSE = 0.008$ and average Euclidean distance is 0.06.

Conclusion This study presented a detailed solution for inverse kinematics problem using an Adaptive Multi-Agent System approach.

To avoid parameter non-correlation, we encoded the output / input of the IK problem as one base-dependent number. Using this approach, we were able to predict the position and orientation of the robot arm joints, given a final position.

The results show that the size of the training set is relevant to the performances, as the bigger it is, the model becomes more complex and the MSE increases. To make the error less, four AMOEBAs were trained with the differently encoded labels.

This applications were aimed as a part of a more complex system, involving machine learning techniques to identified a goal for a robot, and multi-agent system to elaborate the robotic arm position to reach this goal.

References

[1] Samuel R. Buss "Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares methods"IEEE Journal of Robotics and Automation, 16(2004): 1-19.

[2] N. Verstaevel, J. Boes, J. Nigon, D. d'Amico, M. Gleizes "Lifelong Machine Learning with Adaptive Multi-Agent Systems In Proceedings of the 9th International Conference on Agents and Artificial Intelligence, 1(2017): 275-286.

[3] R.R. Serrezuela, A.F.C. Chavarro, M.A.T. Cardozo, A.L. Toquica, L.F.O. Martinez, "Kinematic modelling of a robotic arm manipulator using Matlab ARPN Journal of Engineering and Applied Sciences, 12:7(2017): 2037-2045.

[4] A. Mohammed "Forward and Inverse Kinematic Analysis and Validation of the ABB IRB 140 Industrial Robot"International journal of electronics, mechanical and mechatronics engineering, 7:2(2017): 1383-1401.

[5] Kwon3d.com. (1998). Rotation Matrix. [online] Available at: http://www.kwon3d.com/theory/transform/rot.html [Accessed 28 Aug. 2019].

[6] G. Dudek and M. Jenkin, "Computational Principles of Mobile Robotics"Cambridge University Press, USA, 2nd edition, 2010.

[7] J.-P. Georgé, M.-P. Gleizes, and V. Camps, "Cooperation In Di Marzo G. Serugendo, M.-P. Gleizes, and A. Karageogos, editors, Self-organising Software, Natural Computing Series, pages 7-32. Springer Berlin Heidelberg, 2011.

[8] J. Boes, J. Nigon, N. Verstaevel, M.-P. Gleizes, F. Migeon, "The Self-Adaptive Context Learning Pattern: Overview and Proposal International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT), 9405(2015) in LNAI, Springer, Larnaca, Cyprus, 91-104. .

[9] J. Nigon, E. Glize, D. Dupas, F. Crasnier, J. Boes, "Use Cases of Pervasive Artificial Intelligence for Smart Cities Challenges IEEE Workshop on Smart and Sustainable City (WSSC 2016) associated to the International Conference IEEE UIC (2016): 1021-1027.

Список литературы

[1] Samuel R. Buss "Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares methods"IEEE Journal of Robotics and Automation, 16(2004): 1-19.

[2] N. Verstaevel, J. Boes, J. Nigon, D. d'Amico, M. Gleizes "Lifelong Machine Learning with Adaptive Multi-Agent Systems In Proceedings of the 9th International Conference on Agents and Artificial Intelligence, 1(2017): 275-286.

[3] R.R. Serrezuela, A.F.C. Chavarro, M.A.T. Cardozo, A.L. Toquica, L.F.O. Martinez, "Kinematic modelling of a robotic arm manipulator using Matlab ARPN Journal of Engineering and Applied Sciences, 12:7(2017): 2037-2045.

[4] A. Mohammed "Forward and Inverse Kinematic Analysis and Validation of the ABB IRB 140 Industrial Robot"International journal of electronics, mechanical and mechatronics engineering, 7:2(2017): 1383-1401.

[5] Kwon3d.com. (1998). Rotation Matrix. [online] Available at: http://www.kwon3d.com/theory/transform/rot.html [Accessed 28 Aug. 2019].

[6] G. Dudek and M. Jenkin, "Computational Principles of Mobile Robotics"Cambridge University Press, USA, 2nd edition, 2010.

[7] J.-P. Georgé, M.-P. Gleizes, and V. Camps, "Cooperation In Di Marzo G. Serugendo, M.-P. Gleizes, and A. Karageogos, editors, Self-organising Software, Natural Computing Series, pages 7-32. Springer Berlin Heidelberg, 2011.

[8] J. Boes, J. Nigon, N. Verstaevel, M.-P. Gleizes, F. Migeon, "The Self-Adaptive Context Learning Pattern: Overview and Proposal International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT), 9405(2015) in LNAI, Springer, Larnaca, Cyprus, 91-104. .

[9] J. Nigon, E. Glize, D. Dupas, F. Crasnier, J. Boes, "Use Cases of Pervasive Artificial Intelligence for Smart Cities Challenges IEEE Workshop on Smart and Sustainable City (WSSC 2016) associated to the International Conference IEEE UIC (2016): 1021-1027.