Simulation and Control of a Robotic Arm Using MATLAB, Simulink and TwinCAT

Wei-chen Lee-*IEEE Member*, Shih-an Kuo

Abstract—**It is challenging to develop robot applications without viewing the robot movement. Besides, it is tedious to establish motion paths and adjust controller parameters using the real robot if there is no simulation program available. To resolve the issues for a low-cost robot, we developed a system that integrated kinematics and motion control simulation using MATLAB and Simulink. The system can then be connected to a real robot by using TwinCAT to verify the simulation results. Case studies were conducted to demonstrate that the system worked well and can be applied to those robotic arms without simulators.**

I. Introduction

It is always beneficial for the users of robotic arms to be able to first simulate the robotic behavior before deploying the robotic arms in the product line. However, not all the robotic arms manufacturers can provide such simulation programs. In the past, many scholars have developed techniques on robotic arm simulation. Corke [1] proposed a toolbox for robotic arm simulation based on MATLAB. It contains the forward kinematics, inverse kinematics, and other functions. Breijs et al. [2] proposed an automatic modeling and controller design environment in MATLAB/Simulink for robotic arms. Falconi et al. [3] developed a tool integrating MATLAB and Simulink to help users quickly create a robotic arm simulation environment. The MATLAB functions developed by Corke [1] were also included in their tool. The kinematic and dynamic models of a robotic arm can be obtained using the tool and then simulated in Simulink. Gil et al. [4] proposed a MATLAB robotics toolbox for teaching students the basic concepts of robotics. Many robotic arms on the market were available in the toolbox for easy implementation. In addition to the forward kinematics, inverse kinematics, and path planning, this toolbox can also be used to tune the PID parameters of the controller for the joint control.

In addition to the simulation virtually, many studies have incorporated real robotic arms with the simulation results to demonstrate the streamline implementation. Chinello et al. [5] established a MATLAB toolbox for the KUKA robotic arms, integrated all functions into a user-friendly interface, and connected to the KUKA controller from a computer via TCP / IP. The toolbox contains functions such as forward and inverse kinematics, point-to-point joint control, trajectory generation, etc. González-Palacios et al. [6] proposed a simulation interface called SnAM (Serial n-Axis Manipulators), which was developed under the ADEFID (ADvanced Engineering platForm for Industrial Development) framework to solve the forward kinematics and inverse kinematics. It can help users complete tasks such as trajectory planning and collision avoidance. The integration of SnAM and the adaptive prototype also can help test the performance of serial manipulators of various designs. Mohammed Abu et al. [7] proposed visualization software to solve the problem of insufficient robots in schools. They combined MATLAB/Simulink and AutoCAD to establish forward/inverse kinematics and trajectory planning. Dean-Leon et al. [8] proposed a new MATLAB modeling and simulation toolbox that automated the modeling process. Users only needed to know the D-H (Devanit-Hartenberg) parameters and basic linear algebra to be able to use the toolbox.

For the controller design simulation, many researchers proposed various robotic arm controllers. Piltan et al. [9] realized the PUMA 560 manipulator position control simulation through MATLAB/Simulink and designed the manipulator controller based on the computed torque control (CTC) methods. Through the test results, the authors demonstrated that the PID-CTC controller was better than the PD-CTC controller in terms of the performance of overshoot, steady-state error, and root mean square error. Jamali et al. [10] proposed a structure of modeling, simulation, and optimization based on SolidWorks, MATLAB, and Simmechanics, in which they optimized the controller to reduce the joint errors by using the Genetic algorithm.

Regarding the controllers, many researchers and engineers have used TwinCAT, the windows control and automation technology, in related applications. Shuai et al. [11] proposed an elderly-assisted four-axis robotic arm based on Beckhoff's industry PC. In their study, TwinCAT was used as a communication protocol between the PC computer and motor drivers to achieve the purpose of real-time control. Proskuriakov [12] used TwinCAT to build a real-time control environment for their hydraulic servo system. Fan et al. [13] proposed a welding robot combining a computer and a virtual NC (Numerical Control) to design a device control layer through a Beckhoff controller and TwinCAT.

Based on the discussion above, we can understand that integrating a simulator with a real robotic arm is still needed for those robotic arms without simulators. To address the issue, the objective of this research was to develop a software system to integrate robot simulation and robot control. The robot

This work was financially supported by the "Center for Cyber-physical System Innovation" from The Featured Areas Research Center Program within the framework of the Higher Education Sprout Project by the Ministry of Education (MOE) in Taiwan.

W. C. Lee is with the Department of Mechanical Engineering and the Center for Cyber-physical System Innovation, National Taiwan University of Science and Technology, Taipei, 10607, Taiwan, (phone: 886-2-2737-6478; fax: 886-2-2737-6464; e-mail: wclee@ mail.ntust.edu.tw).

S. A. Kuo is with the Department of Mechanical Engineering, National Taiwan University of Science and Technology, Taipei, 10607, Taiwan, (e-mail: M10603107@mail.ntust.edu.tw).

manufactured by igus (Cologne, Germany) was used as our control target, and we used TwinCAT as the software to control the robotic arm. Using the techniques proposed in this paper, we can quickly establish a simulator for a robotic arm, and use the output of the simulator to control the real one.

II. Robot Description and Kinematics

A. Robot Description

This research used the RL-D-RBT-3322S-BC five-axis robotic arm from igus as our simulation and control target. This robotic arm is articulated and its five are all rotation axes. To make the simulation more realistic, parameters such as the weight, center of mass, and moment of inertia of each link was measured or calculated.

Regarding the motor driver, the SVR-K112 stepper motor drivers manufactured by Taiwan Pulse (Taichung, Taiwan) were used. This stepper motor driver is a single-axis drive with A and B two-phase design. It can be closed-loop controlled through the motor encoder's feedback to reduce the position errors. The communication the driver uses is EtherCAT architecture for real-time control between the server and the device. Devices in the network can be divided into master and slave. The controller of the channel is responsible for sending packets to each slave to read slave's data or control slave behavior.

B. Forward Kinematics

For the coordinate transformation of forward kinematics, D-H notation was used. First, the coordinate structure was defined according to the length and rotation axis of each link of the robotic arm. Then the coordinate transformation relationship was obtained through D-H notation, and the homogeneous transformation matrix between two links was obtained. Finally, the homogeneous transformation matrix of each rotation axis was multiplied to obtain the coordinate of the endpoint of the robotic arm.

C. Inverse Kinematics & Path Generation

We used algebraic approach to solve the inverse kinematics for the robotic arm. When we wanted to generate the trajectory of the endpoint from the selected points, we used the MATLAB function, cscvn, to generate cubic spline curves from the selected points, and MATLAB function, ppval, to obtain the interpolated points among the selected points for a smooth path. Then the we solved the rotations for the axes for each of the interpolated points by using the inverse kinematics. The equations derived from inverse kinematics are similar to those in [7]. There may be multiple solutions for the equations, and we need to limit the solutions for each axis within the physical limits so that we can obtain a unique solution.

III. System Structure

The system architecture of this study is shown in Fig. 1. It includes a visualization interface, TwinCAT server, TwinCAT devices, stepper motor drivers, and a five-axis robotic arm. The visualization interface, TwinCAT server, and devices were all built on the same PC. To facilitate the management, Beckhoff ADS (Beckhoff Automation Device Specification) communication architecture was used for data transmission. The motion control part adopted multi-axis independent control to reduce the impact of communication delay and excessive calculation. The EtherCAT communication architecture was used for the real-time communication between the PC and the motor drivers.

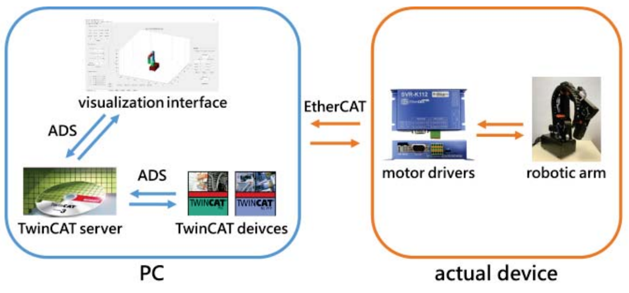


Figure 1. Integrated structure of the proposed system.

A. Simulation Environment

The visualization interface proposed by this research was mainly divided into two parts as shown in Fig. 2. We used MATLAB to perform kinematic simulation and Simulink for dynamic simulation. For kinematics simulation, it can be further divided into joint simulation, endpoint simulation and path planning simulation. The task simulation was to learn the dynamic characteristics of the robotic arm by taking into consideration of the parameters such as the weight and the moment of inertia of the links of the arm. We can also design and simulate the robotic arm controller in the environment.

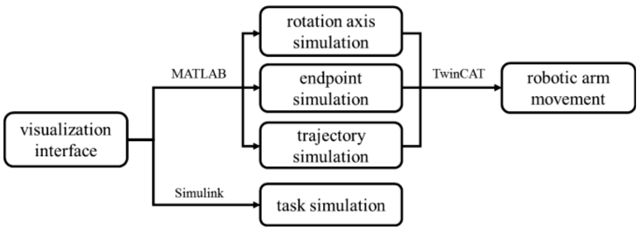


Figure 2. Structure of the simulation.

For the task simulation, the URDF (Unified Robot Description Format) was used to define the relationship between links in the virtual robotic arm. The 3D model with the coordinate system and the rotation axis set was converted into a URDF file through a SolidWorks to URDF exporter. After importing the converted URDF file into MATLAB through the function, importrobot, in the Robotics System toolbox, the information of RigidBodyTree was obtained. RigidBodyTree is a data file representing the structure of the robot. It contains the parent-child relationship of each rotational axis and the type of the rotational axis. It can also include arm parameters such as the mass, center of mass, and moment of inertia.

The kinematics used in task simulation was completed using the Robotics System toolbox, which contains kinematics and dynamics related functions. The Get Transform function block mainly completed the forward kinematics task. For the inverse kinematics, it was primarily done through the Inverse Kinematics function block.

The PID Controller function block in Simulink mainly established the PID controller used in this research. The motion was completed by using the Stateflow toolbox. The Chart function block was used to define and plan the movement flow of the virtual robotic arm. In addition to setting the robotic arm movement, it can also design some complicated scenarios, such as adding other objects and working with the other robotic arm together. Also, the MATLAB functions were used to assist the simulation process. In this study, the endpoint of the virtual robotic arm can move along the planned path. Because the endpoint obtained after kinematics calculations did not exactly match the target path point, it is necessary to define the tolerance, which was used to limit the calculation error. The virtual robotic arm can move only when the difference between the calculated endpoint and the target point was within the difference. Otherwise, the calculation continued until the tolerance requirement was met.

For the communication between the MATLAB-based simulation environment and TwinCAT-based hardware equipment, we adopted the Beckhoff ADS communication architecture. This architecture can connect the TwinCAT server to TwinCAT PLC, TwinCAT NC controller through TCP/IP communication protocol. For the proper connection, AdsPort needed to be set up. AdsPort represents the virtual device for designated communication. There are three communication methods between ADS devices: asynchronous, synchronous, and notification. In this study, the data reading part was written by using notification. The data receiving part established an ADS data stream to send the request to TwinCAT, and a notification was set for the variable to receive the updated value.

B. *Hardware Environment*

In the hardware environment, it was mainly constructed through a user interface, TwinCAT, and the motor drivers. The three communicated with each other through communication protocols to complete the entire process. First, the user performed the user interface operation. Then, according to the selected mode, the information such as the rotational angle and the variables set in TwinCAT PLC were transmitted to TwinCAT PLC using the ADS communication architecture. After receiving the data, TwinCAT PLC modified the corresponding function block value to turn on/off the TwinCAT NC PTP function. Then the TwinCAT NC PTP gave instructions to the motor drivers to move the robotic arm. It also read the driver data and sent the information back to the user interface.

From the description above, we can understand that the user interface played the role of sending requests, TwinCAT PLC played the role of receiving requests and arranging the control processes, while TwinCAT NC PTP gave the instructions to the motor drivers and the motor drivers executed the instructions.

In this study, we used three functions in in TwinCAT PLC, which were motor enable, rotation, and reset. We needed to set the parameters in the MATLAB environment so that we can control the robotic arm through the user interface directly. We also needed to define the necessary parameters for controlling the robotic arm in TwinCAT PLC. Finally, we connected the variables with the ADS communication program written in the MATLAB environment to complete the control of the TwinCAT PLC variables from the MATLAB environment.

We used four functions in the Tc2_MC2 library, which were reading the state of the motor, enabling the motor, controlling the rotation of the motor, and resetting the TwinCAT NC PTP state. Problems may occur at any time during the movement of the robotic arm. Therefore, monitoring can help users to obtain the current status. The MC_ReadStatus function block was used to read the status of each axis of the robotic arm. The MC_Power function block was used to control the motor on/off. In the point-to-point motion control, the MC_MoveAbsolute function block for position control was selected to construct the motor rotation control. Finally, the MC_Reset function block was used to reset TwinCAT NC PTP.

IV. Case Studies

A. *Visualization Interface*

The visualization interface in the MATLAB environment is shown in Fig. 3. It was designed based on the interface proposed by Peli [14]. This interface was established by using the MATLAB GUI. The functions are mainly divided into the visualization window, rotation axis movement, endpoint movement, and path planning. robotic arm control, and task simulation. We used igus robotic arm to demonstrate the operation and functions of the interface.

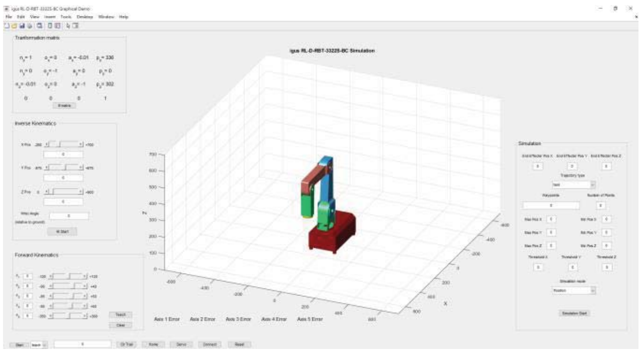
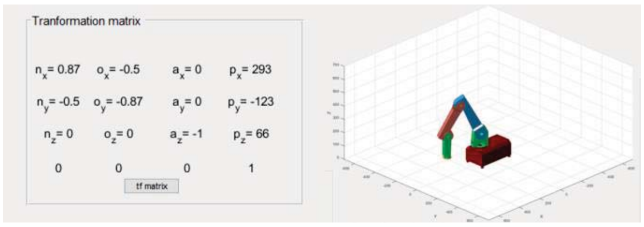


Figure 3. Visualization interface.

B. *Case Study–Kinematics Simulation*

The 3D model of the igus robotic arm was used to realize the operation of the various functions on the visualization interface. First, the target coordinates of the endpoint of the robotic arm were input and the rotation angles required to reach the target coordinates were calculated by using inverse kinematics calculated in MATLAB. Then the rotation angles were input in the visualization interface to obtain the actual endpoint coordinates using forward kinematics. To demonstrate this, we first input the target coordinates [293, -123, 67] to the endpoint movement simulation interface. The rotation angle for each axis obtained through inverse kinematics was used to obtain the endpoint coordinates as [293, -123, 66]. Fig. 4 shows the calculated results and the simulation of the robotic arm at the target position.


Tranformation matrix

$n_x = 0.87$	$o_x = -0.5$	$a_x = 0$	$p_x = 293$
$n_y = -0.5$	$o_y = -0.87$	$a_y = 0$	$p_y = -123$
$n_z = 0$	$o_z = 0$	$a_z = -1$	$p_z = 66$
0	0	0	1

Figure 4. Case study: endpoint simulation result.

From the simulation results, it can be seen that the coordinates of the endpoint of the input and the final endpoint of the output differed by 1 mm in the z-direction, which was caused by the numerical errors. Through the case and other cases (not presented in this paper) we performed in the kinematics simulation, it can be concluded that both the rotation axis simulation and the endpoint movement simulation can reach the specified targets.

Then we connected the robot to our PC with MATLAB/Simulink and TwinCAT so that we could use the calculation results obtained in the MATLAB simulation environment to control the real robotic arm. In the meantime, the visualization interface could display the simulated posture of the robotic arm, which was the same as the posture of the actual robot. Fig. 5 shows the endpoint movement in the simulated and real environment. The endpoint of the robotic arm moved from [377, -171, 145] to [377, 171, 145] through a planned path. It can be found that the simulated results were identical to the actual robot movements. Fig. 5(b) also shows that the movement was not smooth. This was due to the fact that not all the rotation axes started and stopped simultaneously.

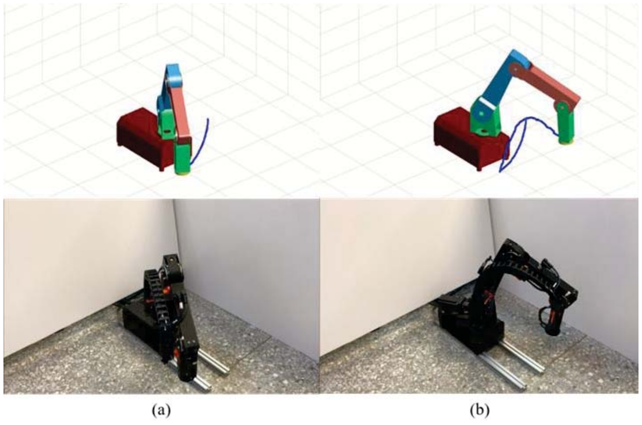


Figure 5. Case study: path simulation result and robotic arm movement. (a) Start position (b) Final position.

C. *Case Study–Task Simulation*

The task simulation case was to carry out the simulation of the production line environment in which a robotic arm and two conveyors worked together. This was similar to the simulation proposed by Miller [15]. In the simulation, first the robotic arm placed an object on a conveyor belt. The conveyor belt detected the object and then moved it to the designated location. When the object reached the location, the conveyor stopped, and then the robotic arm was requested to pick it up and move it according to the planned path to the other conveyor. Then the object was moved by the conveyor until the final location was reached.

To perform the simulation, we first tuned the controller for each axis by using the information of the weight, moment of inertia, and centroid location of each link. The PID parameters of the first to fifth axes were turned to the same values, which were $k_p = 25$, $k_i = 1.5$, and $k_d = 0.15$. In this setup, the virtual robotic arm could move stably. We then established the simulation environment consisting of two conveyors as mentioned previously. The weight of the object was set to 0.3 kgf, and then the path for moving the object between the conveyors was divided into 30 points. The simulation environment and points along the path are shown in Fig. 6.

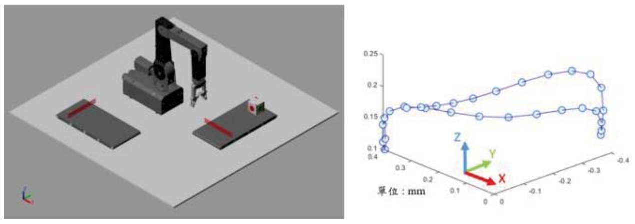


Figure 6. Case Study: Task Simulation environment (left) and the path (right).

For the simulation of pick-and-place, we installed a virtual gripper at the end of the robotic arm, and the endpoint coordinates needed to be offset to the center of the gripper. In this case, it was set to be 140 mm from the endpoint. Then we input the coordinates of the 30 points in the user interface, and limited the movement range to reduce the calculation of inverse kinematics. The range in the *x*-direction was from 38 mm to 400 mm, in the *y*-direction was from -410 mm to 0 mm, and in the *z*-direction was from 100 mm to 240 mm. The tolerance of the endpoint coordinates were set to ± 30 mm in each of the three directions.

The simulation results are shown in Fig. 7. The blue curves in Fig. 7 are the rotation angles of the five axes obtained by using the equations derived from the inverse kinematics. The orange curves are the rotation angles of the five axes obtained by using the inverse kinematic function blocks in MATLB's robotics system toolbox and the PID controller in Simulink. In Fig. 7, from 3.4 s to 4 s shows the motion while the robotic arm is gripping the object, and from 7.8 s to 8.2 s shows the motion while the robotic arm is placing the object. Comparing these two curves, we can see that the calculation results using two different approaches are very consistent, which verifies that the equations derived from inverse kinematics are correct and can be used to calculate the rotation angle for each axis of the robotic arm.

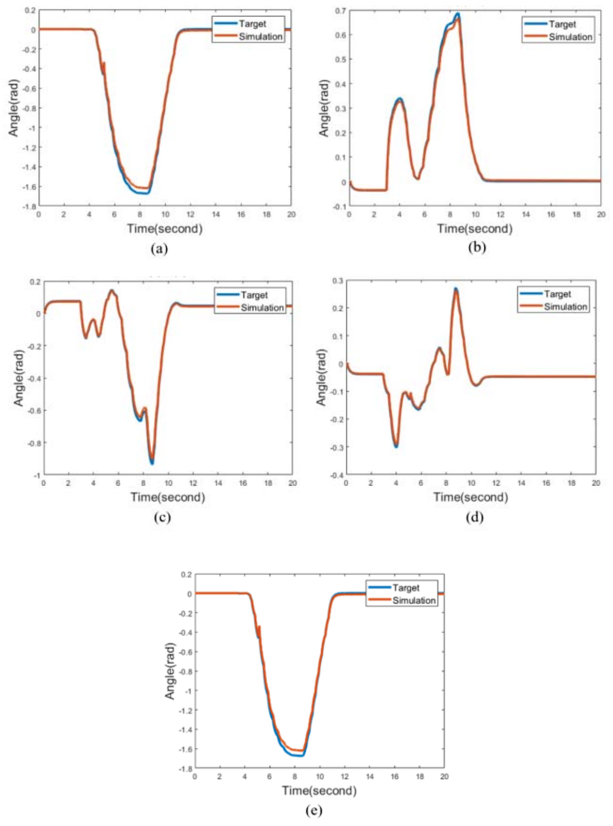


Figure 7. Case Study–Task Simulation result. Axis angle comparison between inverse kinematics and Simulink function in (a) joint 1; (b) joint 2; (c) joint 3; (d) joint 4; (e) joint 5.

V. Conclusion

This study presents a simulator that integrates a virtual environment and actual control functions. The simulation environment required for kinematics simulation was constructed by using MATLAB. The rotation axis simulation, endpoint simulation, and path simulation can be performed based on the forward kinematics and inverse kinematics equations.

Besides, through the simulation environment constructed by Simulink for the task simulation, the controller for the robotic arm with its physical parameters can be designed and simulated.

The outcome of the simulation can be used to control a real robotic arm by using TwinCAT. We demonstrated that the simulation of the motion of the robotic arm can be observed on the real robotic arm. The results of this research can be applied to any similar robotic arm to resolve the difficulties for using them without simulators.

Acknowledgment

The authors would like to thank igus Taiwan (Taichung, Taiwan) for providing the robotic arm for this research, and Taiwan Pulse Motion (Taichung, Taiwan) for providing the stepper motor drivers used to control the robotic arm.

References

[1] P. I. Corke, "A robotics toolbox for MATLAB," IEEE Robotics & Automation Magazine, vol. 3, pp. 24-32, 1996.
[2] A. Breijs, B. Klaassens, and R. Babuška, "MATLAB design environment for robotic manipulators," IFAC Proceedings Volumes, vol. 38, pp. 373-378, 2005.
[3] R. Falconi and C. Melchiorri, "RobotiCad: an Educational Tool for Robotics," IFAC Proceedings Volumes, vol. 41, pp. 9111-9116, 2008.
[4] A. Gil, O. Reinoso, J. M. Marin, L. Paya, and J. Ruiz, "Development and deployment of a new robotics toolbox for education," Computer Applications in Engineering Education, vol. 23, pp. 443-454, 2015.
[5] F. Chinello, S. Scheggi, F. Morbidi, and D. Prattichizzo, "KCT: a MATLAB toolbox for motion control of KUKA robot manipulators," in 2010 IEEE International Conference on Robotics and Automation, 2010, pp. 4603-4608.
[6] M. A. González-Palacios, E. A. González-Barbosa, and L. A. Aguilera-Cortés, "SnAM: a simulation software on serial manipulators," Engineering with Computers, vol. 29, pp. 87-94, 2013.
[7] Q. Mohammed Abu, I. Abuhadrous, and H. Elaydi, "Modeling and Simulation of 5 DOF educational robot arm," in 2nd International Conference on Advanced Computer Control, pp. 2010, 569-574.
[8] E. Dean-Leon, S. Nair, and A. Knoll, "User friendly Matlab-toolbox for symbolic robot dynamic modeling used for control design," in IEEE International Conference on Robotics and Biomimetics (ROBIO), 2012, pp. 2181-2188.
[9] F. Piltan, M. Yarmahmoudi, M. Shamsodini, E. Mazlomian, and A. Hosainpour, "PUMA-560 robot manipulator position computed torque control methods using MATLAB/SIMULINK and their Integration into graduate nonlinear control and MATLAB courses," International Journal of Robotics and Automation, vol. 3, pp. 167-191, 2012.
[10] P. Jamali and K. Heidari Shirazi, "Robot Manipulators: Modeling, Simulation and Optimal Multi-Variable Control," Applied Mechanics and Materials, vol. 232, pp. 383-387, 2012.
[11] G. Shuai, S. Zhuoyuan, W. Zhiyong, and M. M. Islam, "Beckhoff based arm control system design for Elderly Assisting Robot," in IEEE International Conference on Automation and Logistics, 2011, pp. 1-5.
[12] M. Proskuriakov, "Control of a hydraulic servo system using Beckhoff real time data acquisition computer based on MATLAB/SIMULINK platform," Master, School of Energy Systems, Mechanical Engineering, Lappeenranta University of Technology, 2018.
[13] L. Fan, J. Bai, F. Liu, C. Yang, and X. Qin, "Software Realization of Gantry Welding Robot Control System with PC+ Soft NC Mode," in 2nd IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference, 2018, pp. 251-255.
[14] Pasc Peli. (2016). Puma Robot Simulation. Available: https://www.mathworks.com/matlabcentral/fileexchange/59663-puma-robot-simulation
[15] Steve Miller. (2017). Robot Arm with Conveyor Belts. Available: https://www.mathworks.com/matlabcentral/fileexchange/61370-robot-arm-with-conveyor-belts