Artificial Neural Network Based Path Planning of Excavator Arm

Nga T. T.Vu, Do D. Bui, and Hieu T. Tran
Department of Automatic Control, Hanoi University of Science and Technology, Hanoi, Vietnam
Email: nga.vuthithuy@hust.edu.vn, {duydovg, hieutranbk3e}@gmail.com

***Abstract*—This paper presents a solution in path planning for a robotic arm based on the artificial neural network (ANN) architecture, particularly a Static (Feedforward) Neural Network (SNN). The inputs of the network are the sample sets that are obtained from some specific requirements of the desired trajectory. After training, the outputs of the network are the smooth curves that will be used as the reference trajectory for the joints of the excavator arm. The capabilities of the designed neural network in solving the path planning problems are clearly demonstrated through a simulation conducted with a complex trajectory for the excavator.**

***Index Terms*—Artificial Neural Network (ANN), Static Neural Network, Feedforward Neural Network (FFNN), Excavator, Manipulator**

I. Introduction

As one of the important construction machines, an excavator is widely used in various engineering construction fields. However, the traditional excavator is operated by humans directly, and thus, it cannot be used in dangerous construction fields. Therefore, developing an unmanned excavator which can work independently or can be controlled remotely has recently attracted the attention of many scientists and researchers because of its advantages [1–3]. In designing and developing an excavator that can perform the excavation process autonomously, trajectory planning is one of the important roles [4].

The trajectory of the robot arms can be built in both joint space and Cartesian space. In the Cartesian space, the generated trajectory can meet the geometric constraints directly, but it is quite complicated because of inverse kinematics [5]. Therefore, most of the desired trajectories are planned in the joint space [6–15]. In [6], a time-minimum algorithm is used to generate the path for m-joint mechanical manipulators based on the interpolated technique. The generated trajectory is global time-minimum; however, this is an unconstraint algorithm. The hybrid algorithms that combine two or more optimal requirements are presented in [7–17]. In these schemes, besides the time-minimum, some additional characteristics are considered, such as energy optimization and jerk optimization. Moreover, these topologies also take into account the mechanical constraints of the system, i.e., the limitation of the joint position, velocity, acceleration, and jerk.

Recently, the neural network is considered an effective way to generate a trajectory for a manipulator robot. In [16], a dynamic neural network is used to plan the path in an abstract manner. Using a neuron-dynamic system with feedback, the scheme can be used to adaptively generate complex trajectories. However, the training process for this system is quite abstruse. In [17], a neural network is used to identify the properties of soil. The output of the network will be combined with other factor such as bucket volume, reachability, and time efficiency to generate the trajectory of the excavator arm.

This paper proposes a scheme to plan the reference trajectory of the excavator arm based on the Feedforward Neural Network (FFNN). The principle behind this solution is based on the fact that the path planning problem can be defined as generating a geometric path from an initial to a final point, passing through pre-defined via-points which are obtained from some specific requirements of the desired trajectory. The ability to recreate a smooth curve from just a small set of training data of FFNN is well suited for this problem. The capabilities of the designed neural network in solving the path planning problems are clearly demonstrated through a simulation conducted with a complex trajectory for the excavator arm.

This paper is organized as follows. In Section 2, the dynamic model of the system is introduced. Section 3 presents the algorithm background of the neural network and the proposed solution, and Section 4 shows how to design an FFNN and discusses the simulation results. Finally, the conclusion is drawn in Section 5.

II. Dynamic Model of the Excavator

The block diagram of the excavator is shown in Fig. 1. The system can be considered a subclass of a wheeled mobile manipulator with a wheeled mobile robot (excavator base) and a mounted multi-degree-of-freedom manipulator (excavator arm) [18].

Manuscript received July 5, 2018; revised December 8, 2018.

doi: 10.18178/ijmerr.8.1.12-17

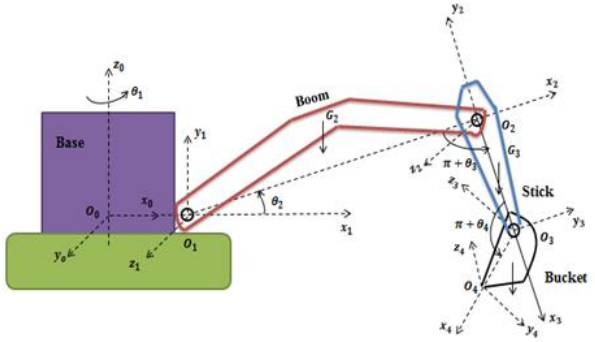


Figure 1. Block diagram of an excavator.

$$D(\theta)\ddot{\theta}+C(\theta,\dot{\theta})\dot{\theta}+G(\theta)+B(\dot{\theta})=\Gamma\tau-F_L, \quad (1)$$

where $\theta=[\theta_1 \ \theta_2 \ \theta_3 \ \theta_4]^T$ is the vector of measured joint angles, $D(\theta)$ denotes the inertia matrix, $C(\theta,\dot{\theta})\dot{\theta}$ is the vector of Coriolis and centrifugal forces, $G(\theta)$ is the vector of gravitational forces, $B(\dot{\theta})$ denotes friction, Γ is the corresponding input matrix, vector $[\tau_1 \ \tau_2 \ \tau_3 \ \tau_4]^T$ specifies the torque acting on the joint shafts, and F_L denotes the interactive torque between the bucket and the environment during the digging operation.

The equations of $D(\theta), C(\theta,\dot{\theta}), G(\theta), \Gamma$ are given as follows [20]:

$$D(\theta)=\begin{bmatrix} D_{11} & D_{12} & D_{13} \\ D_{21} & D_{22} & D_{23} \\ D_{31} & D_{32} & D_{33} \end{bmatrix}, \quad C(\theta,\dot{\theta})=\begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix},$$

$$\Gamma=\begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix},$$

$$G(\theta)=[G_1 \quad G_2 \quad G_3]$$

$$B(\dot{\theta})=\left[B_{bo}\dot{\theta}_2 \quad B_{st}\dot{\theta}_2 \quad B_{bu}\dot{\theta}_2\right],$$

in which

$$D_{33}=I_{bu}+M_{bu}r_4^2$$

$$D_{22}=D_{33}+I_{st}+M_{st}r_3^2+M_{bu}\left[a_3^2+2a_3r_4\cos(\theta_4+\alpha_4)\right]$$

$$D_{11}=D_{22}+I_{bo}+M_{bo}r_2^2+M_{st}\left[a_2^2+2a_2r_3\cos(\theta_3+\alpha_3)\right]$$
$$+M_{bu}\left[a_2^2+2a_2a_3c_3+2a_2r_4\cos(\theta_{34}+\alpha_4)\right]$$

$$D_{23}=D_{32}=D_{33}+M_{bu}a_3r_4\cos(\theta_4+\alpha_4)$$

$$D_{13}=D_{31}=D_{23}+M_{bu}a_2r_4\cos(\theta_{34}+\alpha_4)$$

$$D_{12}=D_{21}=D_{13}+I_{st}+M_{st}\left[r_3^2+a_2r_3\cos(\theta_3+\alpha_3)\right]$$
$$+M_{bu}\left[a_3^2+a_2a_3c_3+a_3r_4\cos(\theta_4+\alpha_4)\right]$$

$$C_{11}=-M_{st}a_2r_3\dot{\theta}_{23}\sin(\theta_3+\alpha_3)-M_{bu}a_2a_3\dot{\theta}_{23}s_3-$$
$$M_{bu}a_2r_4\dot{\theta}_{234}\sin(\theta_{34}+\alpha_4)$$

$$C_{12}=-M_{st}a_2r_3\dot{\theta}_{23}\sin(\theta_3+\alpha_3)-M_{bu}a_2a_3\dot{\theta}_{23}s_3-$$
$$M_{bu}a_2r_4\dot{\theta}_{234}\sin(\theta_{34}+\alpha_4)$$

$$C_{13}=-M_{bu}a_2r_4\dot{\theta}_{234}\sin(\theta_{34}+\alpha_4)$$

$$C_{21}=a_2\dot{\theta}_2\left[M_{bu}a_3s_3+M_{st}r_3\sin(\theta_3+\alpha_3)\right]-$$
$$M_{bu}a_3r_4\dot{\theta}_{234}\sin(\theta_4+\alpha_4)$$

$$C_{22}=-M_{bu}a_3r_4\dot{\theta}_{234}\sin(\theta_4+\alpha_4)$$

$$C_{23}=-M_{bu}a_3r_4\dot{\theta}_{234}\sin(\theta_4+\alpha_4)$$

$$C_{31}=M_{bu}r_4\dot{\theta}_2\left[a_2\sin(\theta_{34}+\alpha_4)+a_3\sin(\theta_4+\alpha_4)\right]+$$
$$M_{bu}a_3r_4\dot{\theta}_3\sin(\theta_4+\alpha_4)$$

$$C_{32}=M_{bu}a_3r_4\sin(\theta_4+\alpha_4)$$

$$C_{33}=0$$

$$G_1=(M_{bu}+M_{st})ga_2c_2+M_{bo}gr_2\cos(\theta_2+\alpha_2)$$

$$G_2=M_{bu}ga_3c_{23}+M_{st}gr_3\cos(\theta_{23}+\alpha_3)$$

$$G_3=M_{bu}gr_4\cos(\theta_{234}+\alpha_4)$$

III. Path-planning-based Ann

A. Background

An artificial neural network (ANN) is based on a collection of connected nodes called artificial neurons. Each connection defines the network topology, and it allows the neurons to transmit a signal from one neuron to another. The receiving neuron can process the signal(s) using the transfer function before passing these processed signal(s) to the neurons connected to it. A typical neuron with R inputs is shown in Fig. 3. The individual inputs p_1, p_2,…, p_R are each weighted by corresponding elements $w_{1,1}$, $w_{1,2}$,…, $w_{1,R}$ of the weight matrix **W** (a more detailed explanation is presented in Ref. [21]).

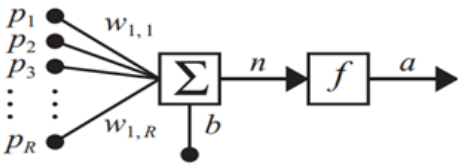


Figure 2. Multiple-input neural network [21].

The output $a \in R$ of the neural network is calculated as follows:

$$a=f\left(\sum_{i=1}^{R}\mathbf{w}_{1,i}\times\mathbf{p}_i+b\right)=f(\mathbf{Wp}+b), \quad (2)$$

where $p_i \in R$ is the i^{th} input of the neuron, $w_{1,i} \in R$ is the weight of the i^{th} input of this neuron, and $b \in R$ is the bias of the neuron. $f(\cdot) = R \rightarrow R$ is the transfer function, and it is different for each neural network that solves different problems. This function can be chosen among some main types of transfer function, such as hyperbolic tangent, sigmoid, and linear functions.

In this paper, we will use the most famous neural network topology, **FFNN.** Its basic structure is shown in Fig. 3.

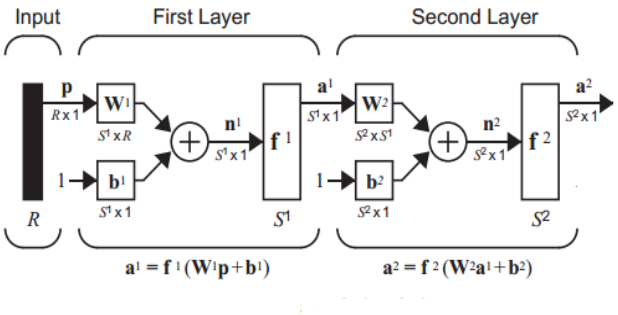


Figure 3. FFNN with two layers [21].

In Fig. 3, the superscript denotes the layer, S^i is the number of neurons in the layer i^{th}, and R is the number of inputs. A layer whose output is the network output is called the output layer. The other layers are called hidden layers. The network shown above has an output layer (layer 2) and a hidden layer (layer 1).

B. Problem Description and Proposed Solution

In an intelligent robotic system, the path planning problem is defined as generating a geometric path, which can either be in the operating space or the joint space, from an initial to a final point, passing through pre-defined via-points. The simplest situation is when the path is to be planned in a static and known environment; however, more generally, the path planning problem can be formulated for any robotic system subject to kinematic constraints in a dynamic and unknown environment [7].

For an excavator, during digging operation, in order to satisfy the technical and economic requirements (mechanical constraints, time – jerk – energy optimization, etc.), the path of joints usually go through some given points. From these via-points, it needs to be calculated to generate a smooth enough trajectory for each joint.

To deal with this problem, the neural network solution is used in this paper. The steps in executing the algorithm are summarized as follows:

Step 1: Given the via-points in the operating space, a kinematic inversion is applied to get a sequence of via-points in the joint space. This will be used as a training dataset for the neural network.

Step 2: Design the structure and parameters of the neural network.

Step 3: Train the network accordingly based on the training dataset. After this step, we will obtain a neural network that can interpolate a trajectory from the via-points.

Step 4: Use the trained network to recreate the trajectory to verify the network.

IV. Simulation and Results

A. Network Creation and Training

The excavator arm includes three joints, i.e., boom, stick, and bucket, and the trajectory of each joint is generated from one neural network. These networks are trained separately, independent of each other, so only one training process is considered, and the training process for the other joints is the same but with different training sample sets. In the simulation, 51 samples for each joint are used, with a sampling time of 0.5 s.

For each joint, a two-layer FFNN with one hidden and one output layer is utilized. The transfer function of the hidden layer is f_1, and that of the output layer is f_2:

$$f_1(n) = tansig(n) = \frac{e^n - e^{-n}}{e^n + e^{-n}} \quad (3)$$

$$f_2(n) = purelin(n) = n, \quad (4)$$

where $n = 0, 1, 2,\ldots, n \in N$.

The number of neurons of the output layer is set to 1, and the number of neurons of the hidden layer is adjusted to obtain the best result. In this paper, we show the training results for two cases with 5 and 20 neurons in the hidden layer, respectively. The learning rate is set to 0.00001, and the number of epochs is set to 1000. The training method is Levenberg–Marquardt. The time it takes to run the simulation is less than 2 min.

The network structure is shown in Fig. 4.

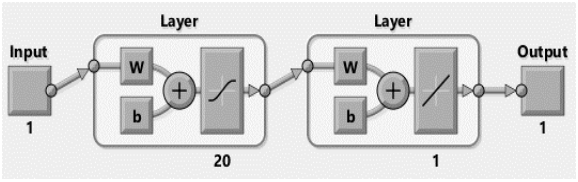


Figure 4. Network structure.

B. Simulation Results

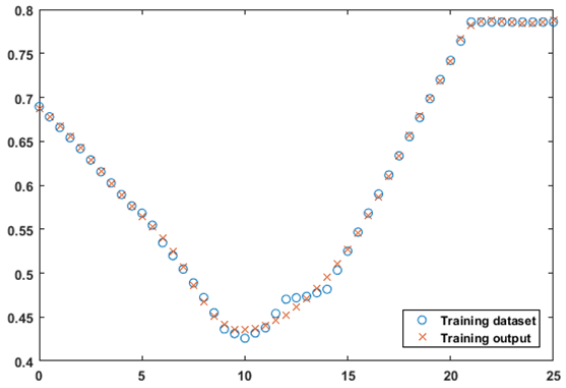
To verify the reliability of the FFNN in the path planning of the excavator arm and the effect of each factor to the training results, the simulation is executed for two cases:

- **Case 1**: The number of samples for each joint is 51, the sampling time is 0.5 s, the learning rate is 10^{-5}, the epochs are 1000, and the number of neurons for the hidden layer are 5.
- **Case 2**: The number of samples for each joint is 51, the sampling time is 0.5 s, the learning rate is 10^{-5}, the epochs are 1000, and the number of neurons for the hidden layer are 20.

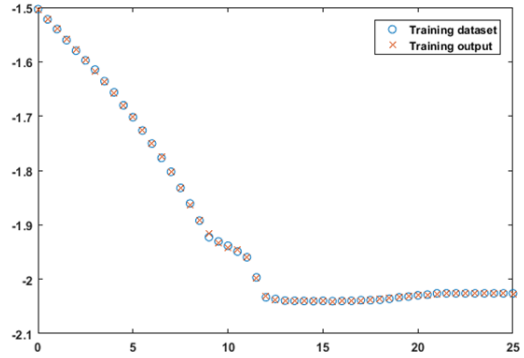
The results of the training process are illustrated in Figs. 5–8, in which Figs. 5–6 show the results for Case 1 and Figs. 7–8 for Case 2. Figs. 5a and 7a show the training results for the first joint (the boom), Figs. 5b and 6b show the results for the second joint (the stick), and the results for the third joint (the bucket) are shown in Figs. 5c and 7c.

For the first case (five neurons in the hidden layer), it can be seen that, for all joints, output matching is not good (Fig. 5), with the training value sometimes not fit with the sample value. Fig. 6 shows the training result of the system in the work space. From the response of each neural network for the joints, using forward kinematics, the desired trajectories in the Cartesian space are obtained. From these, it is easy to plot the trajectory in the work space. It can be seen from Figs. 5 and 6 that, as the shape of the desired trajectory is complicated (in this case, the path changes the direct suddenly), the training data

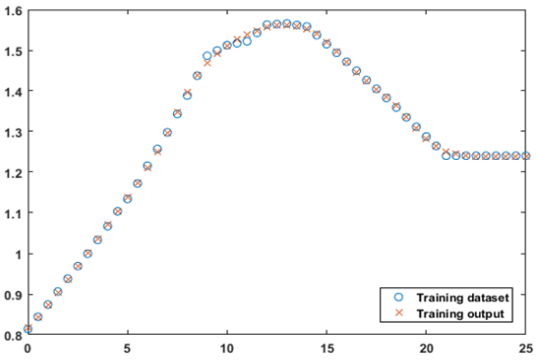
cannot fit with the sample data in all joints (Fig. 5), which makes the error between the generated trajectory and the desired one tolerable.



a. The first joint (the boom)



b. The second joint (the stick)



c. The third joint (the bucket)

Figure 5. Output matching of three joints.

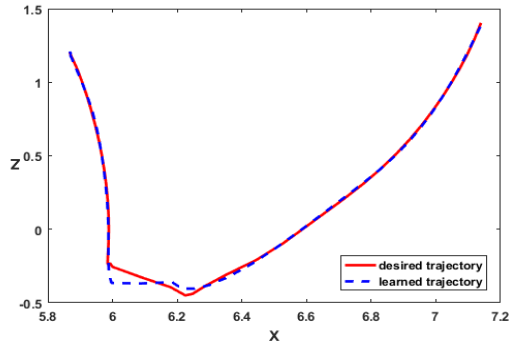
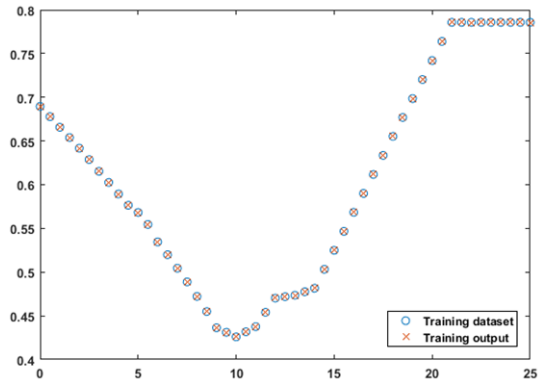
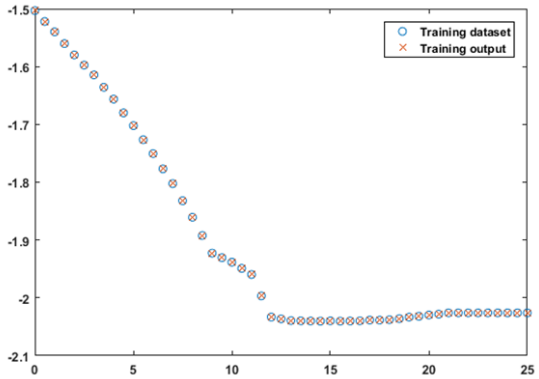


Figure 6. The learned and desired trajectories.

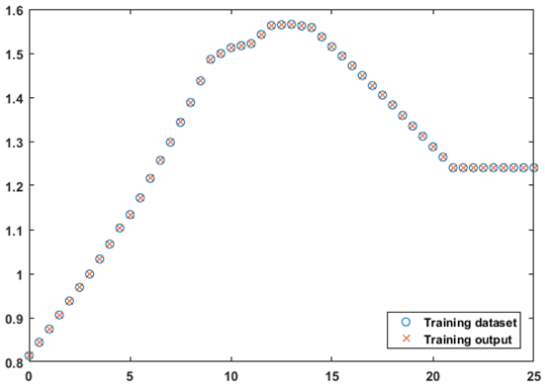
The second test is done with 20 neurons in the hidden layer. The results are shown in Figs. 7–8. Similar with the first case, Fig. 7 shows the output matching of the joints, and Fig. 8 shows the trajectory in the work space. Fig. 7 shows that all training data are the same with the sample, which means that the training output of the neural network matches perfectly with the training dataset. Consequently, the generated trajectory in the work space is almost a duplication of the desired one (Fig. 8). In comparison with Fig. 6, it is demonstrated clearly that, by increasing the number of neurons in the hidden layer, the training results can be improved significantly.



a. The first joint (the boom)



b. The second joint (the stick)



c. The third joint (the bucket)

Figure 7. Output matching of three joints.

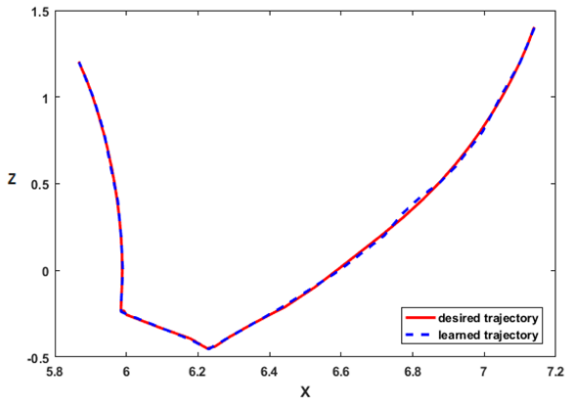


Figure 8. The learned and desired trajectories.

From the simulation results, we can clearly see that, as the number of neurons in the network increases, output matching gets better. It means that, by choosing a suitable network configuration, the FFNN can generate a smooth and exact trajectory for the excavator arm from some given via-points in the desired trajectory.

V. Conclusion

A neural network methodology for trajectory planning of the excavator arm has been described in this paper. Firstly, some sampled points in the desired trajectory in the joint space are derived from specific requirements. These via-points are the training set for the neural network. Next, the neural system built using sub-neural networks is designed and trained. Finally, the simulation test is performed for two cases to verify the effectiveness of the proposed system. The simulation results showed that, by choosing a suitable number of neurons, the presented system can generate a smooth trajectory from some via-points with perfect matching.

Acknowledgments

This work is funded by Ministry of Education and Training (MOET) under the grand number B2018-BKA-70.

References

[1] H. Feng, C. B. Yin, W. W. Weng, W. Ma, J. J. Zhou. W. H. Jia, and Z. L. Zhang, "Robotic excavator trajectory control using an improved GA based PID controller," *Mechanical System and Signal Processing*, vol. 105, pp. 153-168, 2018.

[2] Q. H. Le, J. W. Lee, and S. Y. Yang, "Remote control of excavator using head tracking and flexible monitoring method," *Automation in Construction*, vol. 81, pp. 99-111, 2017.

[3] F. A. Bender, S. Goltz, T. Braunl, and O. Sawodny, "Modeling and offet-free model predictive control of a hydraulic mini excavator," *IEEE Transaction on Automation Science and Engineering*, vol. 14, pp. 1682-1694, 2017.

[4] A. Gasparetto, P. Boscariol, A. Lanzutti, and R. Vidoni, "Path planning and trajectory planning algorithms: A general overview," *Motion and Operation Planning of Robotic Systems*, vol. 29, pp 3-27, 2015.

[5] J. M. Hollerbach, *Robot Motion: Planning and Control*, USA: the MIT press, 2012, ch. 2, pp. 221.

[6] A. Piazzi and A. Visioli, "Global minimum-time trajectory planning of mechanical manipulators using interval analysis," *International Journal of Control*, vol. 71, no. 4, pp. 631–652, 1998.

[7] S. A. Bazaz and B. Tondu, "Minimum time on-line joint trajectory generator based on low order spline method for industrial manipulators," *Robotics and Autonomous Systems*, vol. 29, pp. 257-268, 1999.

[8] K. J. Kyriakopoulos and G. N. Saridis, "Minimum jerk path generation," presented at the IEEE international conference on robotics and automation, Philadelphia, USA, 1988.

[9] A. Piazzi and A. Visioli, "An interval algorithm for minimum-jerk trajectory planning of robot manipulators," in *Proc. 36th IEEE Conf.*, San Diego, USA, 1997, pp. 1924–1930.

[10] A. Piazzi and A. Visioli, "Global minimum-jerk trajectory planning of robot manipulators," *IEEE Trans. on Industrial Electronics*, vol. 47, no. 1, pp. 140–149, 2000

[11] D. Verscheure, B. Demeulenaere, J. Swevers, J. D. Schutter, and M. Diehl, "Time-energy optimal path tracking for robots: a numerically efficient optimization approach," *Advanced Motion Control*, vol. 2, pp. 727–732, 2008.

[12] A. Gasparetton, A. Lanzutti, R. Vidoni, and V. Zanotto, "Experimental validation and comparative analysis of optimal time-jerk algorithms for trajectory planning," *Robotics and Computer-Integrated Manufacturing*, vol. 28, no. 2, pp. 164-181, April 2012.

[13] R. Zhao and S. Ratchev, "On-line trajectory planning with time-interval motion constraints for industrial robot manipulators," presented at IEEE International Conference on Robotics and Automation; 2017.

[14] Z. Zhang, Y. Lin, S. Li, Y. Li, Z. Yu, and Y. Luo. "Tricriteria optimization-coordination motion of dual-redundant-robot manipulators for complex path planning," *IEEE Control System Society*, vol., no. 99, pp. 1-13, 2017.

[15] G. Atmeh and K. Subbarao, "A dynamic neural network with feedback for trajectory generation," *International Federation of Automatic Control*, 2016, pp. 267-272.

[16] S. Lee, D. Hong, H. Park, and J. Bae, "Optimal path generation for excavator with neural networks based soil models," in *IEEE Intenation Conf. Multisensor Fusion and Integration for Intelligent* System, 2008, pp. 632-637.

[17] A. J. Koivo, M. Thoma, E. Kocaoglan, and J. Andrade-Cetto. "Modelling and control of excavator dynamics during digging operation," *Journal of Aerospace Engineering*, vol.9, no. 1, pp. 10 -18, 1996.

[18] C. P. Tang, P. T. Miller, and V. N. Krovi, "Kinematic control of a nonholonomic wheeled mobile manipulator a different approach," presented at the ASME Dynamic Systems and Control Conference, Michigan, USA 2008, pp. 799-806.

[19] Y. Liu, M. S. Hasan, and H. Yu, "Modeling and remote control of an excavator," *International Journal of Automation and Computing*, vol. 7, no. 3, pp. 349–358, 2010.

[20] H. Yu, "Robust combined adaptive and variable structure adaptive control of robot manipulators," in *Robotica*, vol. 16, no. 6, pp. 623–650, 1998.

[21] M. T. Hagan, *Neural Network Design*, 2nd ebook ed., 1996, ch. 2, pp. 900-903

[22] M. T. Hagan, *Neural Network Design*, 2nd ebook ed., 1996, ch. 22, pp. 900-903.

Nga Thi -Thuy Vu received the B.S. and M.S. degrees in electrical engineering from Hanoi University of Science and Technology, Hanoi, Vietnam, in 2005 and 2008, respectively, and the Ph.D. degree in electronics and electrical engineering from Dongguk University, Seoul, Korea, in 2013. She is currently with the Department of Automatic Control, Hanoi University of Science and Technology, Hanoi, Vietnam, as a Full Lecturer. Her research interests include DSP-based electric machine drives and control of distributed generation systems using renewable energy sources.

Do D. Bui is a final-year student in automatic control engineering at Hanoi University of Science and Technology. His research interests include nonlinear control and neural network optimization.

He is currently participating in ERASMUS+ 2018 exchange program and taking part in research activities within Department of Industrial Engineering in University of Trento, Italy.

Hieu T. Tran is also a final-year student in automatic control engineering at Hanoi University of Science and Technology. Currently, he taking part in an internship program at Vietnam Autotech Machinery Company, researching about Designing Controller and Monitoring in Mobile Manufacturing and also participating in renewable energy courses belong to ENCORED program.

His research interests include renewable energy and neural network optimization.