Recurrent Neural Network-based Path Planning for an Excavator Arm under Varying Environment

Nga Thi-Thuy Vu
School of Electrical Engineering
Hanoi University of Science and Technology
Hanoi, Vietnam
nga.vuthithuy@hust.edu.vn

Nam Phuong Tran
School of Electrical Engineering
Hanoi University of Science and Technology
Hanoi, Vietnam
tranphuongnam3098@gmail.com

Nam Hoai Nguyen
School of Electrical Engineering
Hanoi University of Science and Technology
Hanoi, Vietnam
nam.nguyenhoai@hust.edu.vn

Abstract-This paper proposes an algorithm to generate the reference trajectory based on recurrent neural networks for an excavator arm working in a dynamic environment. Firstly, the dynamic of the plant which includes the tracking controller, the arm, and the pile is appropriated by a recurrent neural network. Next, the recurrent neural network combined with a Model Reference Adaptive Controller (MRAC) is used to calculate the reference trajectory for the system. In this paper, the generated trajectory is changed depending on the variation of the pile to maximize the dug weight. This algorithm is simple but effective because it only needs information about the weight at each duty cycle of the excavator. The efficiency of the overall system is verified through simulations. The results show that the proposed scheme gives a good performance, i.e. the dug weight always remains at the desired value (nominal load) as the pile changes its shape during working time.

Keywords-adaptive controller; excavator arm;neural network; path planning; uncertainties

I. INTRODUCTION

The automatic control of an excavator system is a major issue in the field of excavator research [1, 2]. By unmanned operation, the excavator systems not only keep the workers safe but also increase efficiency. However, in order to finish the task without an operator, the trajectory should be designed carefully. Thus, many researches on creating the trajectory for the general manipulator and the excavator have been published. In [3, 4], the 3D trajectory is built by using the information from sensors, cameras, and scanners. The advantage of these types of feedback signals is that they can track the change of the working environment. However, if the working environment lacks light or is dusty, the reliability of the obtained imagines can be reduced. In [5], a laser scanner is used to get the shape of the pile. From this information, the model of the pile is built and divided into small layers. The local path is designed according to these layers before creating the global path. In [6], the integrated physics-based model is presented for a mobile excavator. In this work, the current position of the excavator arm is returned to the control system to predict the trajectory for the next cycle. In [7, 8], a neural network is used to calculate the optimal trajectory for the excavator arm. However, these trajectories only work well in static environments. In [9], excavation trajectories are generated using the velocity and the acceleration of each hydraulic cylinder. The generated trajectories are optimal and stable but the velocity and the acceleration are difficult to measure. In [10], with the purpose of optimizing the efficiency for a semi-automated or fully automated excavator system, the trajectory is classified into 4 categories based on the location and the angle of the bucket. From these 4 trajectory types, the operator or the automatic controller will make a suitable decision as the environment changes. The problem of optimal working time and torque motion of the excavator in consideration with the boundary of the actuator's ability is the role of path planning [11]. In this scheme, the trajectory is planned based on B-spline technique using information about soil parameters and system dynamics.

Neural networks are known as a good way of dealing with path planning problems. In [12-14], neural networks are used to generate collision-free trajectories for robots. In [12], the robot works in a dynamic environment with U-shaped and varying obstacles. The reference trajectory of the robot is generated by using a topologically organized neural network. In this network, the dynamic of each neuron is characterized by a shunting equation. The same neural network topology is used in [13] for a multirobot system with moving obstacles. The trajectory planning for the manipulator robot based on a neural network model of the harmonic function is introduced in [14]. Trajectories are built based on neural networks to optimize the jerk in [15] or the working time in [16]. Neural networks are used in the field of path planning with many different purposes [17-19].

Corresponding author: Nga Thi-Thuy Vu

In this paper, an algorithm based on neural networks is proposed for an excavator arm working in a dynamic environment. The dynamic model of the inner loop which includes the tracking controller, the excavator arm, and the pile is approximated by a neural network. A second, Recurrent Neural Network (RNN) combined with the Model Reference Adaptive Controller (MRAC) algorithm is used to calculate the reference trajectory for the system. By this combination, the calculated trajectory can be adjusted after each duty cycle to adapt with the change of the pile and the dug weight remains around the nominal value despite the reduction of the pile. The effectiveness of the overall system is verified through simulations. The contributions of the proposed algorithm are concluded as:

- It can work in a dynamic environment, something that is restricted in [8].
- The dug weight remains in an acceptable range during the digging process although the material is reducing.

The proposed scheme uses only the feedback signal from the weight sensor to generate the path. This is more reliable than cameras or scanners [3-5] and it is easier to measure than velocity and acceleration [9].

II. PROBLEM DESCRIPTION

In order to maximize the efficiency of the excavator, one of the requirements is that during the digging process, the dug mass at each time must be maintained in a given acceptable range. However, the shape of the pile is changing while the excavator is digging up the material. Therefore, if the trajectory of the excavator remains the same and the dug stack declines over each period of the process, the requirement will not be guaranteed. Hence, the excavator driver has to observe the excavated weight and trajectories in the previous digging to choose an appropriate trajectory next time with an expectation that the mass in the next period will be acceptable. A trajectory generator based on this structure is proposed to replace the excavator driver, which will completely automate the digging process. Before going to solve the described problem, the following assumptions are made:

- The pile lies above the ground and has a triangular shape as shown in Figure 1.
- After a digging cycle, the material on the top of the pile will lie down and fill in the space that was taken, so the pile will maintain the triangular shape with a different slope.
- The dug weight in each period is limited by the volume of the bucket.
- The trajectory in each period is represented by a set of parameters.

The task of the trajectory generator is to observe the weight and the trajectory's parameters in the previous period in order to adjust suitably the trajectory parameters in the next one. In order to execute the algorithm, the excavator's trajectory has a parabolic shape, however, in fact, the excavator's trajectory often has a more complex shape due to many practical conditions. The parabolic shape is described by:

$$y - K = \frac{(x-h)^2}{4p} \quad (1)$$

where K, h and p are scalars. In this work, we have just change the parameter h, which corresponds to the coordinate of the vertex on the horizontal axis. Therefore, each trajectory will be represented by a single value h.

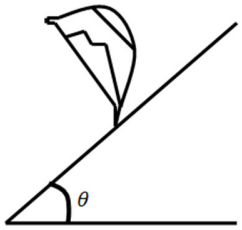

Fig. 1. The shape of the hypothetical pile model.

The block diagram of the overall system is illustrated in Figure 2. It can be seen that the desired trajectory which is generated by the module is denoted as q^*. The desired trajectory is a reference for the excavator controller, and the output of the excavator is q, which affects the dug weight. If the tracking controller has a good performance, the equivalent model of the controller and the excavator can be viewed as a dynamic model which has a static gain of one. In this paper, the equivalent model is considered as a first order system as follows:

$$G(s) = \frac{1}{5s+1} \quad (2)$$

This means that the trajectory error reduces exponentially over time.

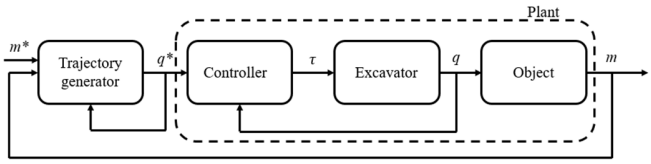

Fig. 2. Block diagram of the overall system.

III. ALGORITHM FOR GENERATING THE REFERENCE TRAJECTORY

In fact, one does not have complete information about the dynamics of the pile. Therefore, a decision on the appropriate trajectory can only be done in practice by observing the pile's response in accordance with the trajectories that the excavator made. Hence, the pile can be considered as a black box, and designing a controller for a black box leads to the idea of the neural network-based MRAC. The control structure of the neural-network based MRAC can be seen in Figure 3. In this Figure, the trajectory generator block can be seen as a controller of the outer loop. This controller is responsible for keeping the dug weight within the specific range as the shape of the pile changes.

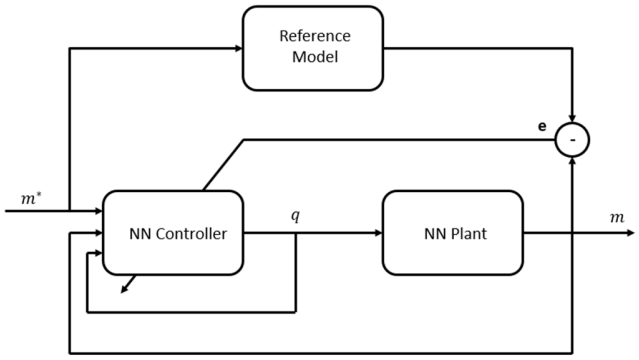

Fig. 3. Control structure of neural network-based MRAC.

This is an offline process, the first step of the design is identifying the plant with the usage of neural network, and the next step is creating a neural network-based controller according to the identified plant in the previous step. Note that, the plant now contains the pile, the excavator, and the tracking controller. The input of the plant is the desired trajectory q^*, whereas the practical trajectory created by the excavator is q because of the control error which should affect the dug weight. Then, the robot excavates the pile with the trajectory q and obtains a weight m. In the identification problem, because the user concerns only about the dug weight after each time the excavator finishes, the object can be viewed as a discrete-time system. A discrete-time system can be described as:

$$m_k = f(q_k^*,...,q_{k-n_u}^*, m_{k-1},...,m_{k-n_y}) \quad (3)$$

Discrete-time system identification can be carried out by using a nonlinear autoregressive network with exogenous inputs denoted as NARX. The NARX's architecture is illustrated in Figure 4, where the input and feedback output of the network pass through the Tapped-Delay-Lines (TDLs) which make up the dynamics of the network.

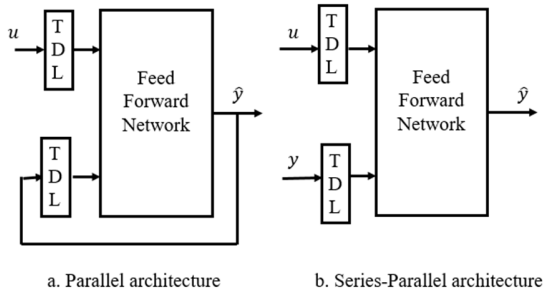

Fig. 4. Parallel and series-parallel architectures of the NARX network.

A feature that makes the NARX become regularly usable in identification problems is that it has only a feedback from the network's output. Therefore, in the training process, one can remove this feedback and consider the feedback output as a second input to the network. The opened architecture, referred as a series-parallel architecture (Figure 4), is used in one-step-ahead prediction. The training process of the opened network is much simpler than that of the closed network, because one can utilize the traditional back-propagation algorithm for the opened network training. After the opened-network training process is completed, the network can predict one-step ahead, however the objective is to identify the system, which means that the network can make a multi-step-ahead prediction. Therefore, the performance of the closed network is often not good enough. Hence, one should continue training the closed network based on the opened network. However, training a closed network can still confront the problem of gradient vanishing, which is a well-known problem in the training of RNNs. Thus, the original training set is divided into subsets which have a smaller size. The network is trained with these subsets. When the training process with these subsets is over, the size of each subset is increased by a small number, and the training restarts with the new bigger subset. These steps are repeated until the size of the subset becomes equal with the size of the original training set. The algorithm is shown in Figure 5, which begins with the size of each subset equal to the maximum number of delays (MD) in the TDL of the network plus one that is equivalent to the training opened network. In Figure 5, *MD* is the Maximum number of Delays of both input's TDL and feedback output's TDL. *Q, c, n* are the length of the original batch, the length of the mini batch, and the number of subsets' respectively, $\underline{x}$ is the network's parameters, e_{ik} represents the error which corresponds with the k^{th} element of the i^{th} subset, $l_i(\underline{x})$ is the squared error which corresponds to the i^{th} subset, $L(\underline{x})$ is the loss function which is the Mean Square Error of the original batch.

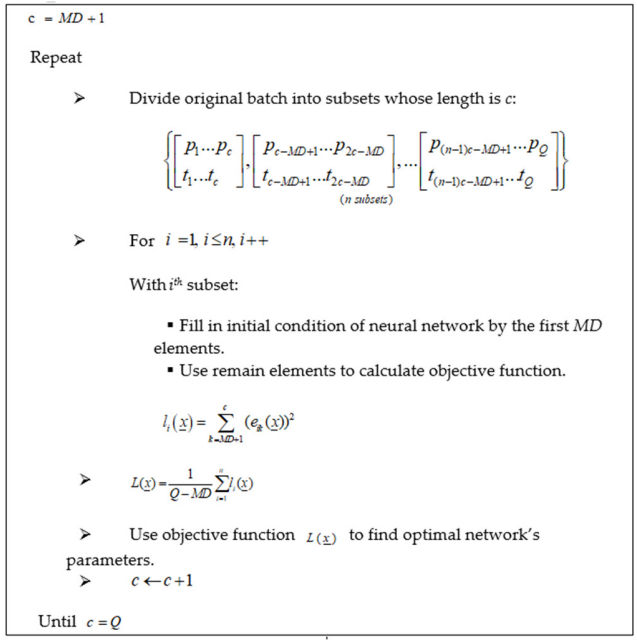

Fig. 5. Training algorithm of the closed network.

Note that each adjacent subset has some similar elements, with size equivalent to *MD* in the TDL. These elements are used as the initial conditions for the network. To guarantee that the objective function includes all elements in the original training batch, a subset is created to overlap the adjacent one. Once the closed network has finished training, we will create the controller network based on the object network. To train the

controller network, the whole structure that contains the plant network and the controller network must be considered as a major network. Then, this network is used to approximate the given reference model. However, during the training process, only the controller network's parameters are adjusted in order to optimize the objective function, while the object network's parameters are kept unchanged. The well-known back-propagation algorithm is used to train the major network, the error between reference model's output and this network's output back-propagates through layers of the network, and the controller network's parameters are updated.

IV. NETWORK TRAINING AND SIMULATION

A. Plant Network

Neural networks are known as a powerful technique to model a system with a mathematical model unknown or difficult to build [20, 21]. In this section, a neural network is used to model the plant which consists of the pile, the excavator, and the tracking controller. The first step of training the plant network is collecting data from the object. The performance of the network depends significantly on the way the data cover and describe the operating range of the object. In system identification problems, the data are often obtained by generating the input signal in the form of a sequence of step functions that have random durations and amplitudes. In this case, the reference input signal is produced with a random amplitude within the intervals [0.3, 0.5] and [0.5, 0.8] in the first 100 and the next 100 steps respectively. The duration of each step function is $d \in [1, 10]$. The main reason is that the collected data need to cover the operation range of the object. In the first 100 steps, because the size of the pile is still large, one should use a trajectory with small h. In contrast, in the next 100 steps, because a large amount of the pile is already taken, one needs to shift the range of the trajectory deep inside the pile with the expectation that the excavated weight is still acceptable. This signal is applied to the plant, then the plant's output is collected. The obtained data are shown in Figure 6.

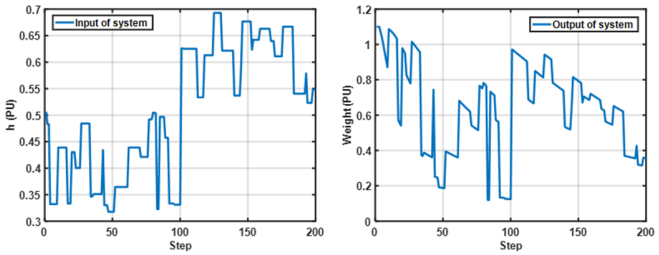

Fig. 6. A mini batch of the training data for the plant network.

Note that there are 40 mini batches whose length is 200 samples, however Figure 6 only shows an instance of the mini batches. Before training, the input data are normalized in the range [-1, 1]. Note that h is the representation of the desired trajectory, the control error in the controlling robot affects the practical trajectory, so the dug weight could not be the same as expected. In this case, the collected data include this control error. When the training data are available, the plant network can be designed. This network is illustrated in Figure 7. The input of the network passes through a 0:1 TDL. The TDL of the input has a current element (zero delay) due to the pile feature. When one adjusts the excavator's trajectory, the dug weight immediately changes depending on the shift of the trajectory, therefore the current component is added to demonstrate this feature. The TDL of network's feedback output is 1:5. The Bayesian regularization algorithm is used for training. The training result is shown in Figure 8. Then, the network is tested with the testing data. The test result is illustrated in Figure 9.

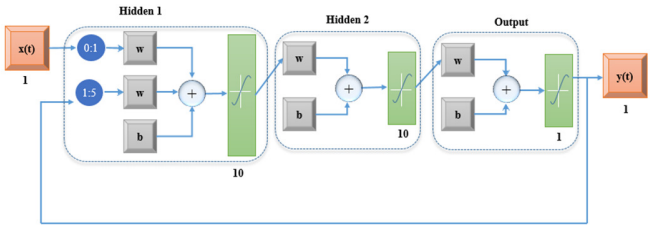

Fig. 7. Plant network's architecture.

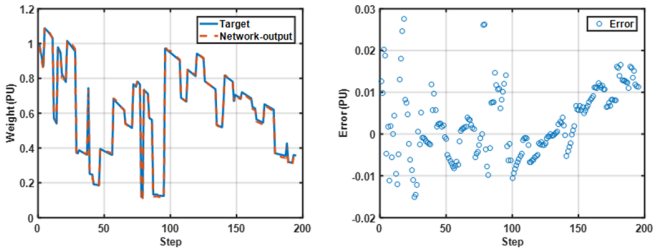

Fig. 8. Training result of the plant network.

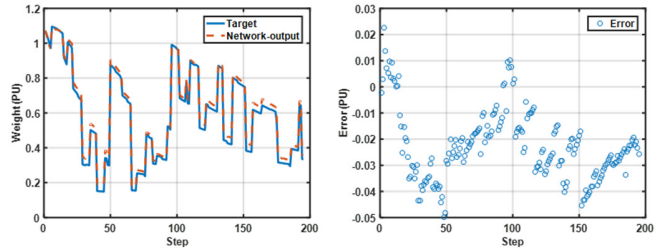

Fig. 9. Testing result of the plant network.

B. Controller Network

First of all, one must choose an appropriate reference model. Note that the plant network is of the 5th order, thus, a fifth-order dead-beat system is utilized as a reference model. The transfer function of the chosen reference model is:

$$Gm(z) = \frac{5z^{-1} + 4z^{-2} + 3z^{-3} + 2z^{-4} + z^{-5}}{15} \quad (4)$$

The next step is collecting data from the reference model for the controller training set. The training data for the controller is demonstrated in Figure 10. The left side in Figure 10 shows the input data of the training set. The right side shows the response of the reference model. Note that the initial value of the reference model's output is the limited dug weight, this means that the dug weight can reach the maximum value in the first period. It can be seen that although the controller only

operates with unit set-point, the input has different values that range from 0.4 to 1. The reason is that the controller has to work with different states of the object, so this type of input helps training the controller with practical states of the object, e.g. the controller can be trained to give a suitable trajectory that will help the dug weight increase from 0.5, which is its current value, to the desired value of 1. The training data have 8 mini batches, each mini batch having a length of 90 samples. Figure 10 illustrates an example of the total training set. The major neural network's architecture, which includes the plant network and the controller network, is shown in Figure 11. The transfer function of the output layer of the controller is the *tansig* function, which means that the output of the controller is within the range from -1 to 1 which is equal to the normalized input range of the plant network.

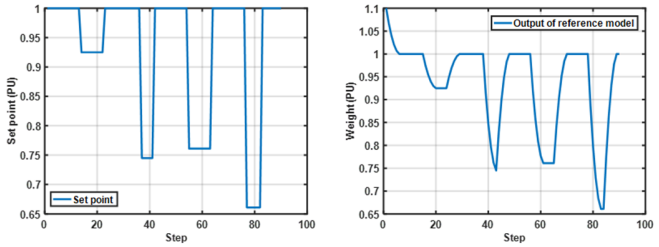

Fig. 10. Training data of the controller network.

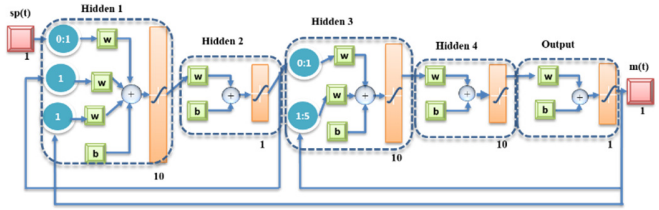

Fig. 11. Major network's architecture.

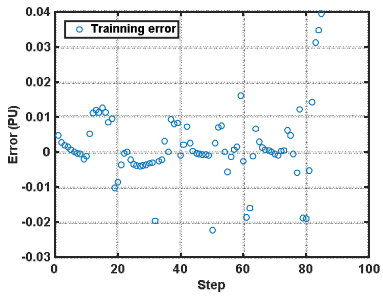

Fig. 12. Training result of the major network.

The used training algorithm is the Bayesian regularization. Figure 12 describes the training result of the major network. It can be seen that the training error is very small, therefore the controller can be tested on the practical object.

C. Stability Analysis

The system contains a plant neural network and a controller neural network. It is a five-layer discrete-time RNN. To analyze the stability of the RNN, there are several stability criteria [22-25]. In this case, the stability criteria from [22] were applied. To do so, the recurrent neural network will be put in the standard form as:

$$x(k+1) = f(W^1x(k)+W^2x(k+1)+b) \quad (5)$$

where $x(k)$ is defined as a vector of layers' output in the past (such as $a^1(k-1)$, $a^2(k-1)$, $a^2(k-2)$, $a^2(k-3)$ ..., where a^i is the output of layer i, i=1,2…5), W^1 and W^2 are the weight matrices of layer 1 and 2. The size of the state vector is 41 in this case. We did obtain the matrices W^1 and W^2, b and f, but the upper bound and lower bound matrices for the function vector $f(41\times1)$ must be found as the requirement of the stability criteria in [20]. This leads to the problem that the equilibrium point of the system in (5) must be determined. It is not easy to find the equilibrium point in this case because its size is 41 and the function vector includes nonlinear functions such as the *tansig*. Thus, in this work the stability of the system is verified through simulations.

D. Simulation Results

In order to verify the effectiveness of the proposed scheme, simulations are done in Matlab/Simulink. In the simulations, the initial path of the arm is set so that the dug mass is 0.4. The trajectory generator should calculate the reference trajectory for the plant to meet the requirement. The dug weight desired value is tracked during working time despite the change of the pile. Figure 13(a) shows the change of the pile during working time. Initially, the slope of the pile is about 62 degrees. Because of the digging, this slope will decrease gradually and reach the value of 46 degrees at the 42th step. Figure 13(b) shows the values of h for each step. The initial value of h is set at -0.9 and after 50 steps its value is 0.8. Corresponding with these 50 values of h, 50 trajectories will be generated to adapt to the change of the pile.

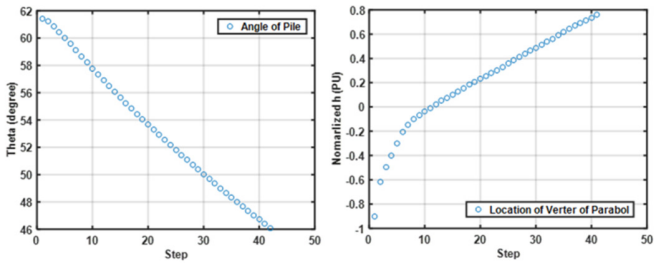

Fig. 13. The change of system during working time. (a) θ, (b) h.

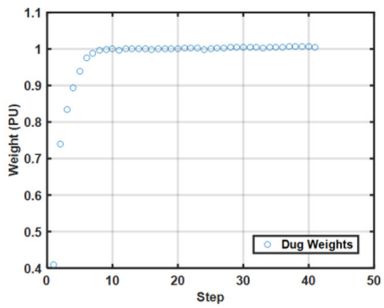

Fig. 14. The dug weight response during working time.

Figure 14 illustrates the dug weight response of the system during working time. In the first step, the excavator works with the given trajectory and the dug weight is 0.4. The information about the trajectory and the weight will be given to the controller to calculate the reference trajectory for the next step. After 5 steps, the dug weight accomplishes the desired value and remains at this state during the working time despite the change of the pile.

V. CONCLUSION

A simple but efficient algorithm has been proposed to design the reference trajectory for an excavator arm in a dynamic environment. The model of the tracking controller, the arm, and the pile is approximated, at first by an RNN. A second RNN combined with the MRAC algorithm is used to calculate the reference trajectory of the system. By this combination, the calculated trajectory can be adjusted after each duty cycle to adapt with the change of the pile so that the dug weight remains around the nominal value despite the reduction of the pile. The effectiveness of the overall system was verified through simulations. The results show that the proposed scheme gives a good performance, i.e. the dug weight always tracks the nominal value as the pile changes its shape during working time.

REFERENCES

[1] H. Feng *et al.*, "Robotic excavator trajectory control using an improved GA based PID controller," *Mechanical Systems and Signal Processing*, vol. 105, pp. 153–168, May 2018, https://doi.org/10.1016/j.ymssp.2017.12.014.

[2] R. Ding, B. Xu, J. Zhang, and M. Cheng, "Self-tuning pressure-feedback control by pole placement for vibration reduction of excavator with independent metering fluid power system," *Mechanical Systems and Signal Processing*, vol. 92, pp. 86–106, Aug. 2017, https://doi.org/10.1016/j.ymssp.2017.01.012.

[3] H. Shao, H. Yamamoto, Y. Sakaida, T. Yamaguchi, Y. Yanagisawa, and A. Nozue, "Automatic Excavation Planning of Hydraulic Excavator," in *Intelligent Robotics and Applications*, Berlin, Heidelberg, 2008, pp. 1201–1211, https://doi.org/10.1007/978-3-540-88518-4_128.

[4] A. Stentz, J. Bares, S. Singh, and P. Rowe, "A robotic excavator for autonomous truck loading," in *Proceedings. 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications*, Victoria, BC, Canada, Oct. 1998, vol. 3, pp. 1885–1893, https://doi.org/10.1109/IROS.1998.724871.

[5] J. Seo, S. Lee, J. Kim, and S.-K. Kim, "Task planner design for an automated excavation system," *Automation in Construction*, vol. 20, no. 7, pp. 954–966, Nov. 2011, https://doi.org/10.1016/j.autcon.2011.03.013.

[6] Y. H. Zweiri, L. D. Seneviratne, and K. Althoefer, "Model-based automation for heavy duty mobile excavator," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Lausanne, Switzerland, Sep. 2002, vol. 3, pp. 2967–2972, https://doi.org/10.1109/IRDS.2002.1041723.

[7] S. Lee, D. Hong, H. Park, and J. Bae, "Optimal path generation for excavator with neural networks based soil models," in *2008 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, Seoul, Korea, Aug. 2008, pp. 632–637, https://doi.org/10.1109/MFI.2008.4648015.

[8] N. T.-T. Vu, N. P. Tran, and N. H. Nguyen, "Adaptive Neuro-Fuzzy Inference System Based Path Planning for Excavator Arm," *Journal of Robotics*, vol. 2018, Dec. 2018, Art. no. e2571243, https://doi.org/10.1155/2018/2571243.

[9] Z. Li, X. Li, S. Liu, and L. Jin, "A study on trajectory planning of hydraulic robotic excavator based on movement stability," in *2016 13th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, Xi'an, China, Aug. 2016, pp. 582–586, https://doi.org/10.1109/URAI.2016.7625784.

[10] R. Tiwari, J. Knowles, and G. Danko, "Bucket trajectory classification of mining excavators," *Automation in Construction*, vol. 31, pp. 128–139, May 2013, https://doi.org/10.1016/j.autcon.2012.11.006.

[11] Y. B. Kim, J. Ha, H. Kang, P. Y. Kim, J. Park, and F. C. Park, "Dynamically optimal trajectories for earthmoving excavators," *Automation in Construction*, vol. 35, pp. 568–578, Nov. 2013, https://doi.org/10.1016/j.autcon.2013.01.007.

[12] S. X. Yang and M. Meng, "Neural network approaches to dynamic collision-free trajectory generation," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 31, no. 3, pp. 302–318, Jun. 2001, https://doi.org/10.1109/3477.931512.

[13] Howard Li, S. X. Yang, and Y. Biletskiy, "Neural network based path planning for a multi-robot system with moving obstacles," in *2008 IEEE International Conference on Automation Science and Engineering*, Arlington, VA, USA, Aug. 2008, pp. 163–168, https://doi.org/10.1109/COASE.2008.4626446.

[14] A. Pashkevich, M. Kazheunikau, and A. E. Ruano, "Neural network approach to collision free path-planning for robotic manipulators," *International Journal of Systems Science*, vol. 37, no. 8, pp. 555–564, Jun. 2006, https://doi.org/10.1080/00207720600783884.

[15] D. Simon, "The application of neural networks to optimal robot trajectory planning," *Robotics and Autonomous Systems*, vol. 11, no. 1, pp. 23–34, May 1993, https://doi.org/10.1016/0921-8890(93)90005-W.

[16] J. Giri, P. Giri, and R. Chadge, "Neural Network Based Modelling of Time Optimal Interpolation of Non-linear Trajectory," *Materials Today: Proceedings*, vol. 5, no. 2, Part 2, pp. 7981–7990, Jan. 2018, https://doi.org/10.1016/j.matpr.2017.11.482.

[17] G. Atmeh and K. Subbarao, "A Dynamic Neural Network with Feedback for Trajectory Generation," *IFAC-PapersOnLine*, vol. 49, no. 1, pp. 367–372, Jan. 2016, https://doi.org/10.1016/j.ifacol.2016.03.081.

[18] Y. Li, R. Cui, Z. Li, and D. Xu, "Neural Network Approximation Based Near-Optimal Motion Planning With Kinodynamic Constraints Using RRT," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 11, pp. 8718–8729, Nov. 2018, https://doi.org/10.1109/TIE.2018.2816000.

[19] P. Zhang, C. Xiong, W. Li, X. Du, and C. Zhao, "Path planning for mobile robot based on modified rapidly exploring random tree method and neural network," *International Journal of Advanced Robotic Systems*, vol. 15, no. 3, May 2018, Art. no. 1729881418784221, https://doi.org/10.1177/1729881418784221.

[20] A. S. Kote and D. V. Wadkar, "Modeling of Chlorine and Coagulant Dose in a Water Treatment Plant by Artificial Neural Networks," *Engineering, Technology & Applied Science Research*, vol. 9, no. 3, pp. 4176–4181, Jun. 2019, https://doi.org/10.48084/etasr.2725.

[21] L. B. Salah and F. Fourati, "Systems Modeling Using Deep Elman Neural Network," *Engineering, Technology & Applied Science Research*, vol. 9, no. 2, pp. 3881–3886, Apr. 2019, https://doi.org/10.48084/etasr.2455.

[22] N. H. Nguyen and M. Hagan, "Stability analysis of layered digital dynamic networks using dissipativity theory," in *The 2011 International Joint Conference on Neural Networks*, San Jose, CA, USA, Jul. 2011, pp. 1692–1699, https://doi.org/10.1109/IJCNN.2011.6033428.

[23] N. E. Barabanov and D. V. Prokhorov, "Stability analysis of discrete-time recurrent neural networks," *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 292–303, Mar. 2002, https://doi.org/10.1109/72.991416.

[24] N. E. Barabanov and D. V. Prokhorov, "A new method for stability analysis of nonlinear discrete-time systems," *IEEE Transactions on Automatic Control*, vol. 48, no. 12, pp. 2250–2255, Dec. 2003, https://doi.org/10.1109/TAC.2003.820158.

[25] M. Liu, "Delayed Standard Neural Network Models for Control Systems," *IEEE Transactions on Neural Networks*, vol. 18, no. 5, pp. 1376–1391, Sep. 2007, https://doi.org/10.1109/TNN.2007.894084.