Journal of Intelligent & Robotic Systems (2019) 95:137–147
https://doi.org/10.1007/s10846-018-0781-0

Neural Network Based Adaptive Actuator Fault Detection Algorithm for Robot Manipulators

Chang Nho Cho[1] · **Ji Tae Hong**[1] · **Hong Ju Kim**[1]

Received: 22 May 2017 / Accepted: 19 January 2018 / Published online: 31 January 2018

Abstract
In order to improve the reliability of robotic systems, various fault detection and isolation (FDI) algorithms have been proposed. However, most of these algorithms are model-based and thus, an accurate model of the robot is required although it is hard to obtain and often time-varying. Acceleration estimation is an additional challenge in dynamic model-based algorithms as it is hard to measure accurately in practice. In this study, a neural network based fault detection algorithm that does not require the use of physical robot model and acceleration is proposed. By utilizing neural network, the fault torque can be estimated, which allows effective fault detection and diagnosis. The feasibility of the proposed fault detection algorithm is validated through various simulations and experiments.

Keywords Fault detection · Neural network · Residual observer · Robot safety

1 Introduction

Along with the increase in the use of robots in industry automation, surgery, service and rehabilitation, the reliability of robotic systems is becoming more important. Thus, intensive studies have been done on fault detection and isolation (FDI) algorithms in order to let a robotic system to detect its faults and perform appropriate reactions. Most of these algorithms compare the sensor output with the predicted output from a mathematical model to compute the mismatch. The mismatch is often called the residual, and the residual can be used to diagnose the system; if the residual exceeds the pre-defined threshold, the robot concludes that there is a fault on the system. However, the residual generation often requires accurate robot dynamic model and acceleration; acceleration is noisy and hard to estimate in practice and robot dynamic model is often not available and bounds to change over time.

In order to avoid the use of manipulator acceleration, observers based on torque filtering [1] and generalized momenta [2–4] have been proposed. As for the model uncertainty, several robust observers based on dynamic threshold [5, 6], sliding mode observer [7] or unknown input high gain observer [8] approaches have been developed. On the other hand, as an alternative approach to cope with the model uncertainty of robotic systems, adaptive control-based approaches are also introduced, which perform model identification prior to collision detection to minimize the model uncertainty of the system [1, 9]. However, while adaptive control-based solutions allow the controller to cope with the model uncertainty of the robot manipulators, these algorithms often require heavy computational load or manipulator acceleration. As for the robust algorithms, many of these algorithms are based on the assumption that the model uncertainties are within the user-defined bounds, and the algorithm performances are heavily dependent on these assumptions. Also, for a manipulator with a high number of DOF, the computation of the robot dynamic model involves heavy computational load.

Recently, neural network (NN) has been gaining much attention in robotics research. NN has been successfully

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s10846-018-0781-0) contains supplementary material, which is available to authorized users.

✉ Hong Ju Kim
hjkim0429@keri.re.kr

[1] Precision Control Research Center, Korea Electrotechnology Research Institute, 12, Bulmosan-ro, 10beon-gil, Seongsan-gu, Changwon-si, Korea

implemented for manipulator motion control [10], force control [11], model identification [12] and inverse kinematics problem [13]. NN's nonlinear mapping properties and learning capabilities make NN an attractive solution for robot manipulator problems [14]. Many studies aimed to use NN to detect system faults [15–22]. In [15] and [16], NN based fault detection algorithms are proposed with a fault accommodation scheme. Other studies used a NN to compute the residuals in the form of system position and velocity [17] or acceleration [18] to detect system faults. In [19], multiple NNs and neural sliding mode based observer are used to enable fast, reliable fault detection. Also, a fault detection strategy using a combination of a nonlinear observer and a NN to predict the system output and detect faults of a 3 DOF helicopter is proposed [20]. Several studies further improved the strategy by employing the extended Kalman filter (EKF) to enable faster fault estimation and detection [21, 22].

While these NN based fault detection algorithms are effective, there are several short comings to be addressed. Many of these algorithms are constructed based on the assumption that the model uncertainties are bounded. Thus, the threshold must be set higher than the bound, limiting the sensitivity of the detection. Other algorithms often assume that accurate robot model is available, and their performance heavily depends on the accuracy of the given model. Also, many algorithms require acceleration information or the use of multiple NNs, which makes them difficult to implement in actual practices. Furthermore, most of these algorithms output residuals as position, velocity or acceleration. However, to accurately diagnosis a fault, the estimation of the fault torque is often desired [1, 3]. The estimation of the fault torque is also helpful for collision detection, a type of fault on which a robot collides with its environment or humans. Using the estimated collision torque, one can set up more meaningful threshold based on various safety criterions and perform an appropriate reaction to minimize the damage [2].

In this study, a novel neural network based adaptive actuator fault detection algorithm for robot manipulators is proposed. Like the other NN based fault detection algorithms, the proposed algorithm uses a NN to generate the residual. However, the network is designed to enable effective fault detection and diagnosis. The aims of the proposed collision detection algorithm are: (1) generation of an acceleration-free residual signal, (2) estimation of the fault torque through the residual generation, and (3) replacement of physical robot model with a NN. The acceleration free residual generation enables the proposed algorithm to be easily implemented to robotic systems without any additional sensors or noisy acceleration information. The estimation of the fault torque through residual generation allows system to accurately detect and diagnosis various types of faults. Also, by using a NN, the proposed fault detection algorithm can adapt to changes and uncertainties in robot model. Using the proposed algorithm, a learn-and-detect scheme is established. The performance of the proposed algorithm is validated through simulations and experiments.

2 Model-Based Residual Generation Algorithm

FDI schemes usually aim to detect faults in manipulator sensors and actuators. This study focuses on faults on manipulator actuators and thus, the residual is generated on the actuator torque to diagnose the state of the actuators. The dynamic model of an n-DOF robot manipulator can be described as [23]:

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau} - \boldsymbol{\tau}_{\mathrm{f}} \tag{1}$$

where $\boldsymbol{q} \in \Re^n$ is the manipulator joint position vector, $\boldsymbol{M}(\boldsymbol{q})$ is the manipulator inertia matrix, $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ represents the manipulator Coriolis and centrifugal matrix, $\boldsymbol{g}(\boldsymbol{q})$ is the manipulator gravity vector, $\boldsymbol{\tau}_{\mathbf{f}}$ is the fault torque and $\boldsymbol{\tau}$ is the actuator torque. Note that the expression for the joint torque τ requires the dynamic model of the robot and its acceleration. In order to avoid the use of the manipulator acceleration, an integrator is employed as shown in Eq. 2.

$$\boldsymbol{\tau}_{\boldsymbol{filtered}} = K \int (\boldsymbol{\tau} - \boldsymbol{\tau}_{\boldsymbol{filtered}})dt \tag{2}$$

where $\boldsymbol{\tau}_{\boldsymbol{filtered}}$ is the filtered torque and K is the gain. Equation 2 can be re-written as:

$$\boldsymbol{\tau}_{\boldsymbol{filtered}} = \frac{K}{s+K}\boldsymbol{\tau} \tag{3}$$

where s is the Laplace variable. From Eq. 3, it can be seen that the filtered torque $\boldsymbol{\tau}_{\boldsymbol{filtered}}$ is the low pass filtered version of the joint torque τ and the gain K works as the filter cut-off frequency. The use of an integrator to increase the robustness of the algorithm against sensor noise and avoid using acceleration has been employed in other studies [2, 24].

By substituting Eq. 1 into Eq. 2 and noting Eq. 4 and Eq. 5, one can obtain Eq. 6, the expression for the filtered torque $\boldsymbol{\tau}_{\boldsymbol{filtered}}$.

$$\frac{d}{dt}(\boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{q}}) = \dot{\boldsymbol{M}}(\boldsymbol{q})\dot{\boldsymbol{q}} + \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} \tag{4}$$

$$\dot{\boldsymbol{M}}(\boldsymbol{q}) = \boldsymbol{C}^T(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \tag{5}$$

$$\boldsymbol{\tau}_{\boldsymbol{filtered}} = K\left\{\int \left(\boldsymbol{g}(\boldsymbol{q}) - \boldsymbol{C}^T(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - \boldsymbol{\tau}_{\boldsymbol{filtered}}\right)dt + \boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{q}}\right\} \tag{6}$$

where Eq. 5 is derived from the skew-symmetry of $\dot{\boldsymbol{M}}(\boldsymbol{q}) - 2\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ [23]. Note that the filtered torque $\boldsymbol{\tau_{filtered}}$ expressed in Eq. 6 can be computed without the robot acceleration. In practice, the given robot dynamic model consists of the estimated values. Thus, Eq. 6 becomes:

$$\boldsymbol{\tau_{filtered}} = K\left\{\int\left(\hat{\boldsymbol{g}}(\boldsymbol{q}) - \hat{\boldsymbol{C}}^T(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - \boldsymbol{\tau_{filtered}}\right)dt + \hat{\boldsymbol{M}}(\boldsymbol{q})\dot{\boldsymbol{q}}\right\} \tag{7}$$

where hat is used to denote estimated values. On the other hand, as the robot joint torque can be estimated from the actuator current information or directly measured from joint torque sensors, the actual filtered torque can be computed from Eq. 2. Thus, the residual can be obtained by:

$$\boldsymbol{r} = \boldsymbol{\tau_{filtered}} - \hat{\boldsymbol{\tau}}_{\boldsymbol{filtered}} \tag{8}$$

where $\boldsymbol{r}$ is the residual. As Eqs. 2 and 7 do not require the robot acceleration, the residual can be generated without the robot acceleration. However, it should be noted that Eq. 7 requires the computation of robot dynamic model, which requires heavy computational load. Also, as Eq. 7 consists of the estimated model of the robot, it can be seen that the model uncertainty leads to the error in the residual generation. In this study, instead of directly computing the estimated filtered torque $\hat{\boldsymbol{\tau}}_{\boldsymbol{filtered}}$ using given uncertain robot model, NN will be used to mimic Eq. 6. Through the use of NN, the computation of the robot dynamic model can be avoided, and residual generation without physical robot model can be realized.

3 Neural Network Based Collision Detection Observer

3.1 Neural Network Setup

In this study, 3-layer NN with nonlinear hidden layer and linear output layer is used. NN consists of an input layer, a hidden layer and an output layer. As the goal of NN is to estimate the filtered torque $\boldsymbol{\tau_{filtered}}$, the NN training signal v is defined as:

$$\boldsymbol{v} = \boldsymbol{\tau_{filtered}} - \hat{\boldsymbol{\tau}}_{\boldsymbol{filtered,\ NN}} \tag{9}$$

where $\hat{\boldsymbol{\tau}}_{\boldsymbol{filtered,\ NN}}$ is the filtered torque estimated by NN. As for the NN activation function, the hyperbolic tangent function is used:

$$f(x) = 2\left(\frac{1}{1+e^x}\right) - 1 \tag{10}$$

and

$$f'(x) = \frac{1 - f^2(x)}{2} \tag{11}$$

To mimic Eq. 6, the inputs for NN are chosen as the same as the inputs for Eq. 6: actual joint angles, actual joint velocities and previous value of $\tau_{filtered}$. The output of NN can be described as:

$$\hat{\tau}_{filtered,\ NN,\ k} = \left[\sum_{j=1}^{n_h}\omega_{jk}^2\left(2\left(\frac{1}{1+\exp\left(-\left(\sum_{i=1}^{n_i}\omega_{ij}^1 x_i + b_j^1\right)\right)}\right) - 1\right)\right] + b_k^2 \tag{12}$$

where $\hat{\tau}_{filtered,\ NN,\ k}$, ω_{ij}^1, ω_{jk}^2, b_j^1, b_k^2, x_i, n_i and n_h are the estimated filtered torque of k-th joint, first layer weight, second layer weight, bias of j-th hidden neuron, bias of k-th output neuron, i-th input of input vector $\boldsymbol{X}$, a number of inputs and a number of hidden layer neurons, respectively, where $k = 1 \ldots n$. Thus, the residual can now be defined as:

$$\boldsymbol{r} = \boldsymbol{\tau_{filtered}} - \hat{\boldsymbol{\tau}}_{\boldsymbol{filtered,\ NN}} \tag{13}$$

3.2 Backpropagation Algorithm

The weights and biases of NN network are updated in real time through the use of backpropagation algorithm. The algorithm is listed below [10]:

$$\Delta\omega_{ij}^1(t) = \eta\frac{1}{2}\left(1 - \left(s_j^1\right)^2\right)x_i\left[\sum_{k=1}^{n_o}v_k\omega_{jk}^2\right] + \alpha\Delta\omega_{ij}^1(t-1) \tag{14}$$

$$\Delta\omega_{jk}^2(t) = \eta v_k s_j^1 + \alpha\Delta\omega_{jk}^2(t-1) \tag{15}$$

$$\Delta b_i^1(t) = \eta\frac{1}{2}\left(1 - \left(s_j^1\right)^2\right)\left[\sum_{k=1}^{n_o}v_k\omega_{jk}^2\right] + \alpha\Delta b_j^1(t-1) \tag{16}$$

$$\Delta b_k^2(t) = \eta v_k + \alpha\Delta b_k^2(t-1) \tag{17}$$

$$s_j^1 = 2\left(\frac{1}{1+\exp\left(-\left(\sum_{i=1}^{n_i}\omega_{ij}^1 x_i + b_j^1\right)\right)}\right) - 1 \tag{18}$$

$$\hat{\tau}_{filtered,NN,k} = \sum_{j=1}^{n_h}s_j^1\omega_{jk}^2 + b_k^2 \tag{19}$$

where η and α are the learning rate and the momentum coefficient, respectively. The momentum term is used to avoid local minima while updating the NN weights and biases [14].

3.3 Fault Detection Algorithm with Neural Network

The proposed FDI algorithm uses a NN to estimate the filtered torque $\hat{\boldsymbol{\tau}}_{\boldsymbol{filtered,\ NN}}$. Once trained, NN would yield

a reasonable estimate of the filtered torque. Also, from the sensor data, the actual filtered torque, $\tau_{filtered}$, can be calculated. Thus, the residual, which is the difference between the estimated filtered torque and the actual filtered torque, would be small. However, upon a fault, a fault torque will be added to the actual filtered torque. The estimated filtered torque, on the other hand, would not include the fault torque. Thus, the estimated filtered torque would deviates from the actual filtered torque, which leads to the rise in the residual.

The proposed FDI scheme is illustrated in Fig. 1. As can be seen from the figure, at every control period, the estimated filtered torque and actual filtered torque are obtained from NN network and sensor data, and the residual is computed. If the residual is below the pre-defined threshold, the controller concludes that there is no fault. Then, the NN weights and biases are updated, and the robot manipulator is controlled to carry out its tasks. However, if the residual exceeds the threshold, the manipulator enters the fault reaction mode, on which it performs appropriate reactions to handle the given fault.

However, an issue with using a NN is that a short period of time is required to train the network. This delay is needed upon start up, and whenever the manipulator dynamics change; when the manipulator pick up an object, for example. Thus, a simple window function is proposed. Upon start up or known payload change, the proposed fault detection scheme is enabled when the residual is within the threshold for the pre-defined time t_{window} (Fig. 2).

It should be noted that during the fault reaction state, the update of NN is halted. NN is trained to estimate the normal manipulator behavior; it is thus not desirable for NN to learn from the fault torque. Once the controller decides that the collision is clear, it will resume updating NN.

4 Simulations

4.1 Simulation Setup

The performance of the proposed algorithm is demonstrated through Matlab simulations. A 2-DOF manipulator is considered, which is to follow a rectangle path in the workspace. The DH (Denavit-Hartenberg) notations of the simulated 2-DOF manipulator are listed in Table 1. Each link is assumed to be a uniform rod, with weight of 5 kg and 3 kg for link 1 and 2, respectively. The simulated robot and the desired workspace trajectory are shown in Fig. 3. It is assumed that each joint of the simulate robot is controlled by a proportional-derivative (PD) controller with gravity compensation, and a nonlinear friction model is applied to each joint [25]. To add time-varying dynamics in the simulation, an object of 2 kg is added at the end of link 2 at

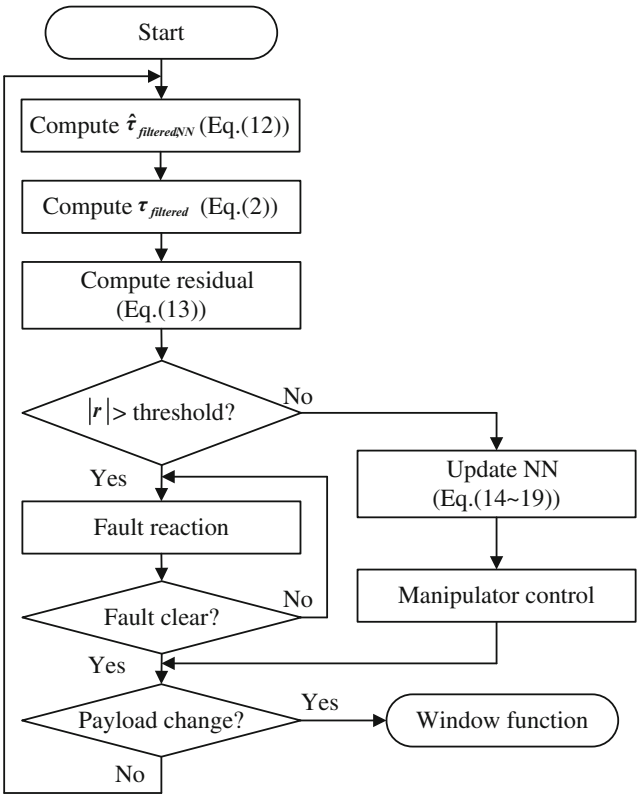

Fig. 1 Block diagram of proposed FDI algorithm

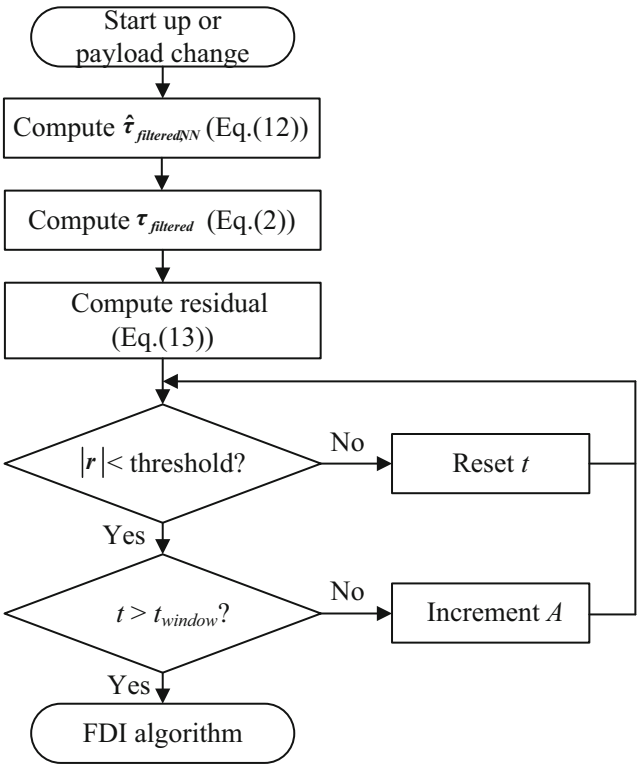

Fig. 2 Block diagram of window function

Table 1 DH notation of simulated 2-DOF manipulator

DH	a_i (m)	α_i	d_i	θ_i
Link 1	1	0	0	θ_1
Link 2	1	0	0	θ_2

point C. In the simulation, it is assume that the manipulator expects a change in the manipulator dynamics upon the object grasping, and the collision detection scheme is halted until the window function is satisfied. The corresponding joint trajectory and velocity tracking error for joint 1 and 2 are depicted in Fig. 4a and b, respectively.

The target rectangle is 0.5 m × 0.8 m (Point A∼D in Fig. 3), and the maximum velocity for joint 1 and 2 are 20°/s and 17°/s, respectively, which corresponds to the maximum workspace velocity of 0.37 m/s. To show that the proposed fault detection algorithm can estimate the fault torque, three types of fault which are widely used for fault detection studies are simulated [1, 3, 9, 21, 22]: step (bias), ramp and total actuator fault. A total actuator fault is a fault that the right hand side of Eq. 1 becomes zero [3]. As for the step and ramp faults, a step torque of 15 Nm and a ramp torque with slope of 900 Nm/s are applied to joint 1 whereas a 35 Nm step torque and a 700 Nm/s ramp torque are given to the second joint.

The simulation is conducted with the sampling period of 1 ms, and the learning rate of NN is set to be 0.005 while 0.9 is used for the momentum coefficient. NN consists of 6 input neurons, 10 hidden layer neurons and 2 output neurons. The inputs of NN are: joint 1 actual angle, joint 1 actual velocity, joint 1 previous $\boldsymbol{\tau}_{\boldsymbol{filtered}}$, joint 2 actual angle, joint 2 actual velocity and joint 2 previous $\tau y_{\boldsymbol{filtered}}$. As for the low pass filter gain, 95 is used. To simulate the noise in joint torque reading, a random noise signal of ±1 Nm

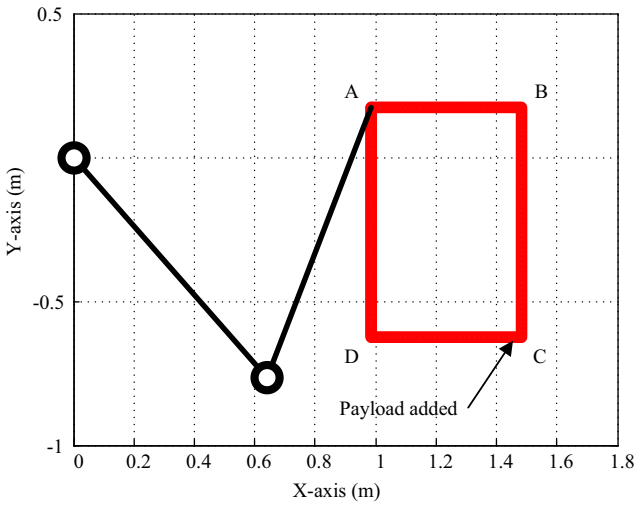

Fig. 3 Simulated 2-DOF manipulator and desired workspace trajectory

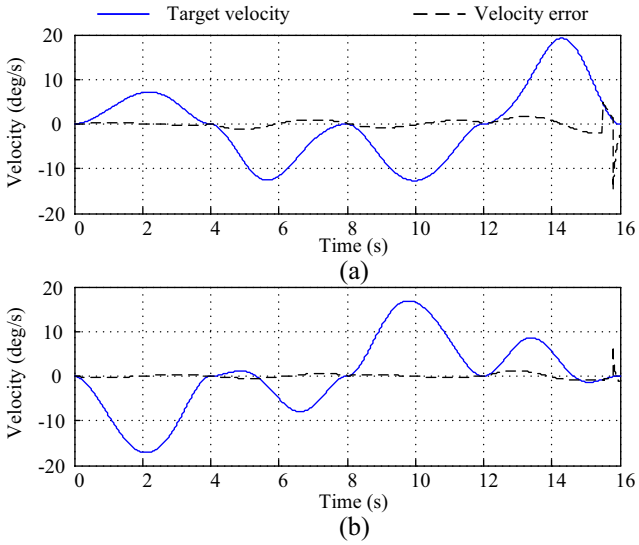

Fig. 4 Desired joint trajectory and corresponding velocity error for 2-DOF robot manipulator **a** joint 1, **b** joint 2

is added to the joint torque. The learning rate, momentum coefficient, number of neurons and low pass filter gains are selected experimentally. The fault detection threshold is set to 1.5 Nm while 0.3 second is used as the window function delay time t_{window}.

4.2 Simulation Results

The simulation results are presented below. The fault toque and the residual for joint 1 are plotted in Fig. 5a whereas the results for joint 2 are illustrated in Fig. 5b.

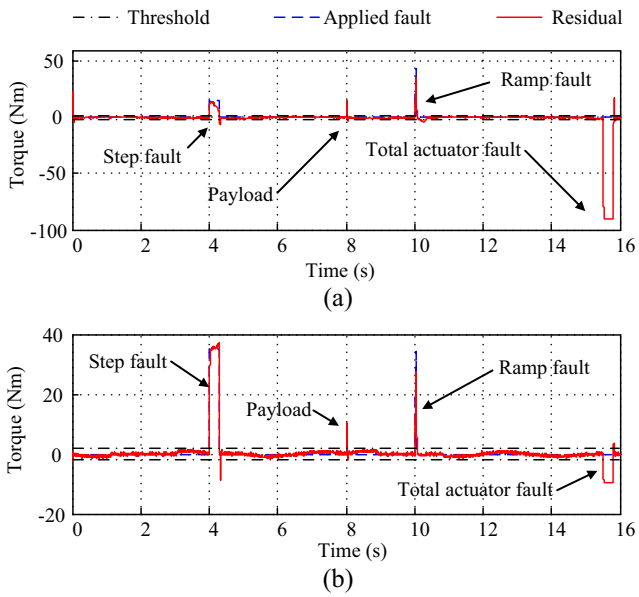

Fig. 5 Fault detection simulation results **a** joint 1, **b** joint 2

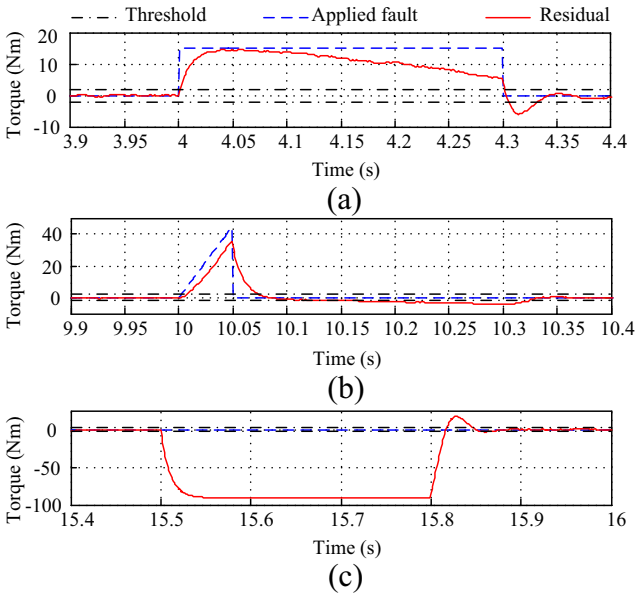

Fig. 6 Fault detection simulation results for joint 1 **a** step fault, **b** ramp fault, **c** total actuator fault

The close up view of the step, ramp and total actuator fault for joint 1 and 2 are depicted in Figs. 6a, b and c and 7a, b and c. As can be seen from the results, the residual stayed below the threshold during normal operations. However, when a fault occurs, the residual increases abruptly, enabling the controller to detect the fault. Figure 8 shows the fault signal, which is 0 during a normal operation and 1 when a fault on either joint is detected. The step, ramp and total actuator faults are successfully detected. The simulated

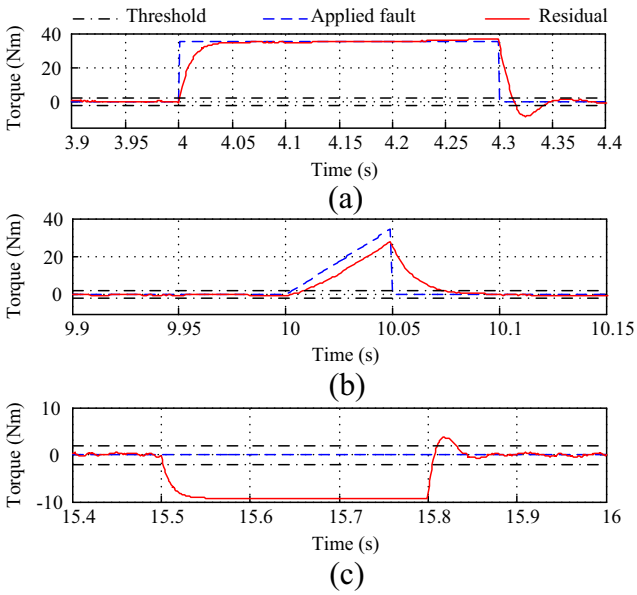

Fig. 7 Fault detection simulation results for joint 2 **a** step fault, **b** ramp fault, **c** total actuator fault

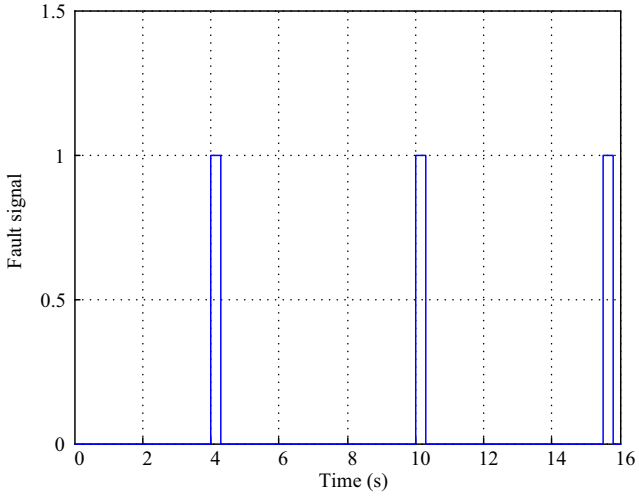

Fig. 8 Fault signal during simulation

faults are given at $t = 4$s, $t = 10$s and t = 15.5s, and the corresponding detection times are $t = 4.001$s, $t = 10.001$s and t = 15.501s; the algorithm detected the faults within 1 sampling period. However, in the actual practice, due to the sensor delay and noise, it is expected that the detection may take a few sampling period.

Also, an increase in the residual is observed for both joints at $t = 8$s. This is due to the addition of the 2 kg payload. However, as can be seen from the results, the residual quickly drops below the threshold as the weights and gains of the NN are updated to cope with the change in the manipulator dynamics.

5 Experiments

5.1 Experimental Setup

In order to evaluate the feasibility of the proposed algorithm, experiments are performed using the system shown in Fig. 9. A 15 cm link is directly attached to a 400 watt servo motor (R88M-K40030T-S2, Omron, Japan) to form a 1-DOF system. A PC running Linux with real time kernel is used as the controller, and EtherCAT communication is established between a servo drive (R88D-KN04H-ECT, Omron, Japan) and the PC to exchange the command and sensor data. The built-in position controller of the servo drive is used to control the link position. The available feedbacks from the driver are motor position and torque, and the velocity is numerically differentiated from the motor position data. A low pass filter is used to suppress the noise in the computed velocity data. The control period of the PC is set to 1 ms while the learning rate, momentum coefficient, number of hidden layer neurons, threshold and window

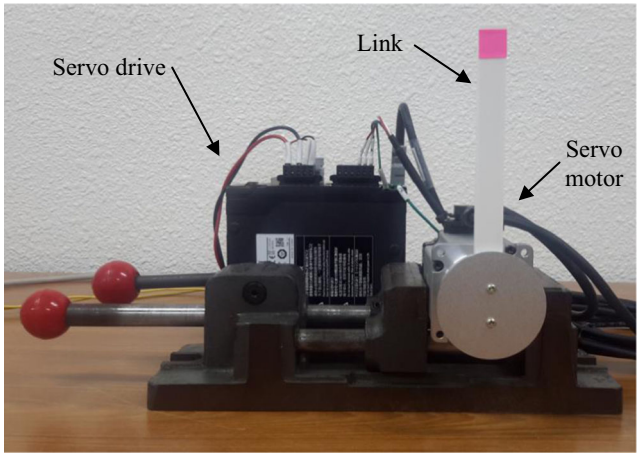

Fig. 9 Set up for collision detection experiment

function delay time are experimentally set to be 0.016, 0.9, 3, 0.0040 Nm and 1.5 seconds, respectively.

The link is controlled to repeatedly follow the given trajectory shown in Fig. 10. The velocity and position tracking errors are also plotted in Fig. 10. During the operation, a fault must be given to the system, so that the proposed FDI algorithm can be experimentally tested. For the experiments, two types of fault are considered: collision and total actuator fault. A collision is one of the common faults on which a robot collides with an object or a human, and it may leads to serious damage or injury. Collisions can be easily simulated for experiments. On the other hand, the servo drive used for the experiments follows CiA402 motion profile [26]. Thus, during a motion, the servo drive is forced from *Operation Enabled* state to *Switch on Disabled* state to cut off the output torque and cause a total actuator fault.

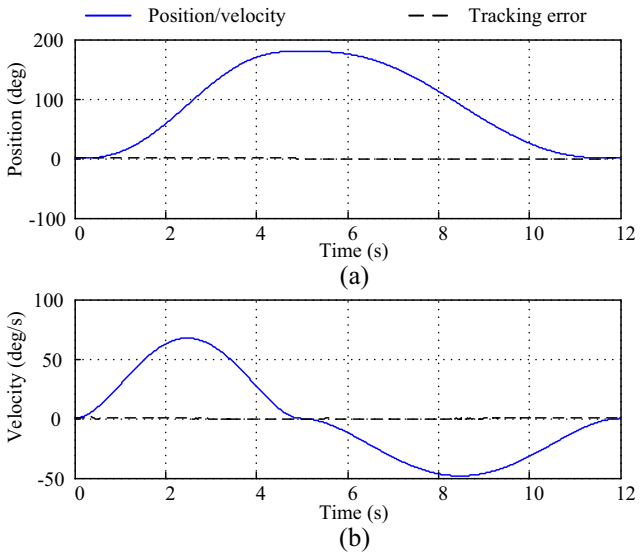

Fig. 10 Target trajectory and tracking performance for 1-DOF system during experiments **a** position, **b** velocity

Once the controller detects a fault, the controller immediately halts the update of NN. Upon the detection of a fault, a proper reaction strategy is desired; assuming the detected fault is a collision, the controller follows a bang-bang trajectory in the opposite to the fault, where the direction of the collision is estimated from the sign of the residual. After that, the servo is to remain stationary for 1 second and move back to the origin, where it resumes the normal operation with NN update. Once the window function is satisfied, collision detection is enabled again. However, for the case of a total actuator fault, the servo cannot retreat from the collision via bang-bang trajectory. Thus, if excessive position error is detected after the bang-bang motion command, the controller assumes that it is a total actuator fault rather than a collision and stops the motion generation and displays an error message.

5.2 Experimental Results

During the collision detection experiment, two collisions are applied to the 1-DOF system, in opposite directions. Figure 11 shows the residual recorded during the experiment and Fig. 12 shows the corresponding fault signal. It can be seen that both collisions are successfully detected. The residual is below the threshold during the normal operation and it increases abruptly upon a collision. A big error on the residual is noted during the collision reaction stage, when no collision is applied to the link but the residual is much bigger than the threshold. This is due to the fact the update of NN has been halted during the collision reaction stage. Without NN update, the controller cannot guarantee that the estimated filtered torque from NN resembles the actual filtered torque. The NN update resumes as the controller enters the next normal operation stage, and the residual

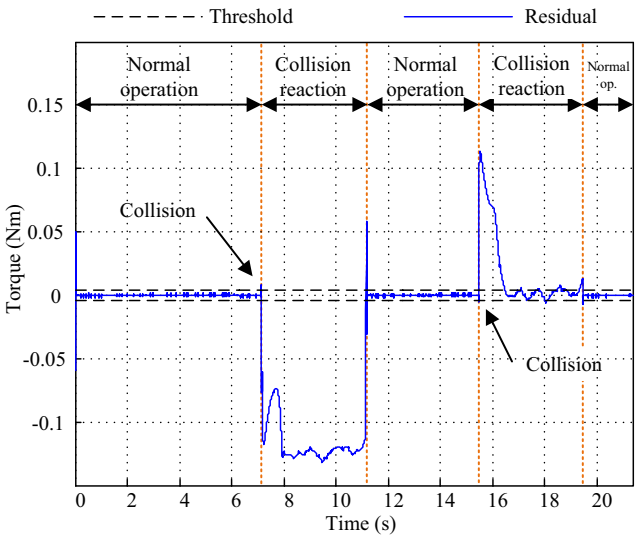

Fig. 11 Residual during collision detection experiments

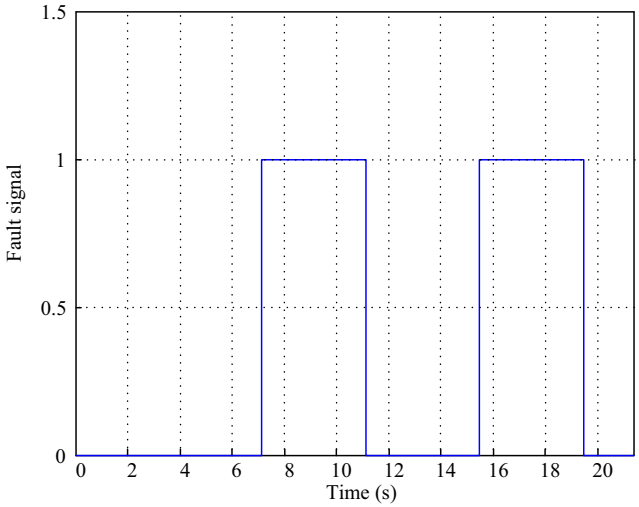

Fig. 12 Fault signal during collision detection experiment

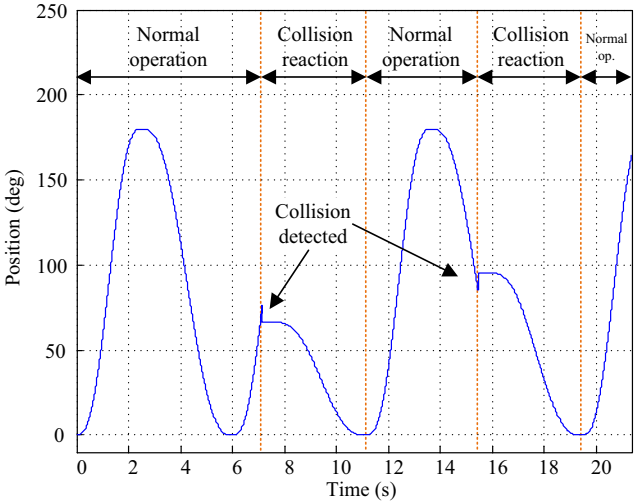

Fig. 14 Position data of 1-DOF system during collision detection experiment

quickly drops below the threshold. The close-up views of the residual at the collisions are depicted in Fig. 13.

Figure 13a shows the actual filtered torque (dashed line) and the estimated filtered torque (solid line) and Fig. 12b shows the corresponding residual. Note from Fig. 13a that upon a collision, which was applied at approximately $t = 14.7$s, only the actual filtered torque quickly rises. The residual also rises abruptly, as can be seen in Fig. 13b. The peak due to the collision in Fig. 13a resembles the residual in Fig. 13b. This indicates that the external torque due to the collision is successfully estimated in the residual. Figure 13c and d shows the similar results for the collision applied at $t = 15.45$s. The second collision is applied in the opposite direction and thus, the residual rises in the opposite direction (Fig. 13b and d).

The resulting motion of the 1-DOF is plotted in Fig. 14. As can be seen from the results, when the collisions were detected, the link quickly retreats, waits and moves back to the origin before resume the original motion.

The resulting residual for the total actuator fault experiment is shown in Fig. 15. The servo drive is forced into the torque off state (*Switch on disabled* state of CiA 402) at t = 3s, and the residual quickly rises accordingly.

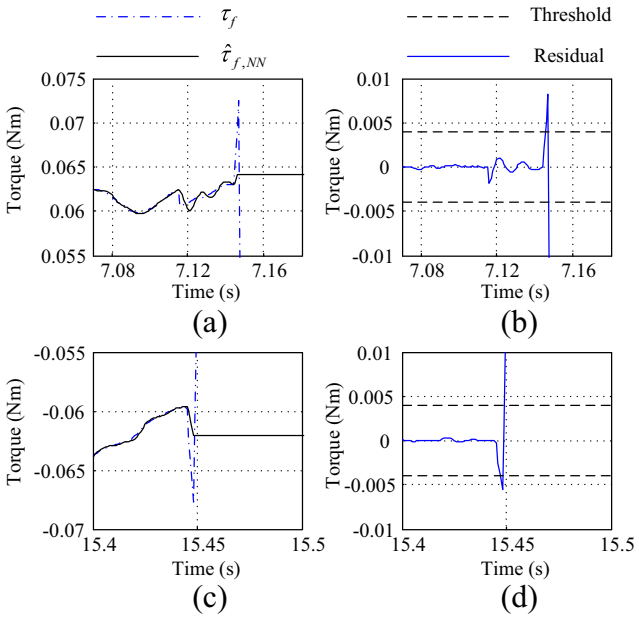

Fig. 13 Close-up view of residual and filtered torques upon collisions **a** filtered torque and estimated filtered torque for first collision, **b** resulting residual for first collision, **c** filtered torque and estimated filtered torque for second collision, **b** resulting residual for second collision

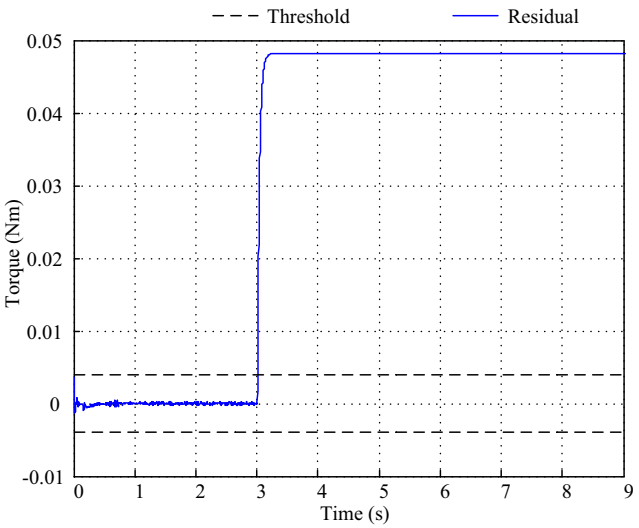

Fig. 15 Residual during total actuator fault experiment

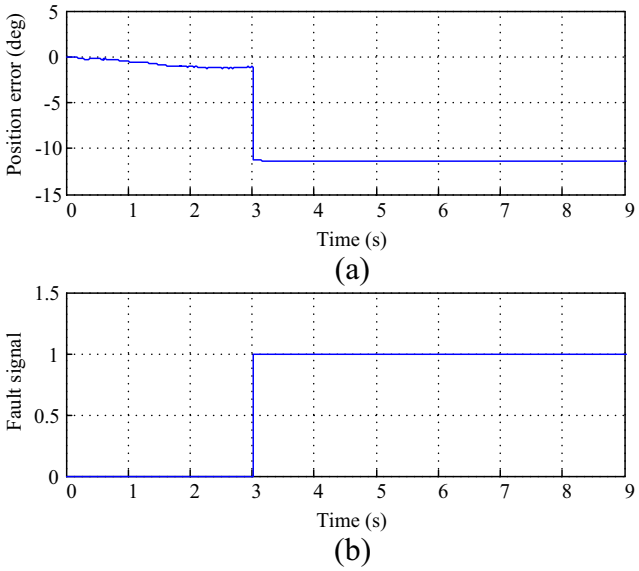

Fig. 16 Results of 1-DOF system during total actuator fault experiment **a** position error, **b** fault signal

The position tracking error and fault signal is shown in Fig. 16a and b, respectively. At t = 3s, upon the total actuator fault, big position error can be noticed, and the controller decides that the detect fault is a total actuator fault. Thus, the fault signal remains high afterwards.

6 Discussion

During the simulation and experiment, the proposed algorithm is able to detect the simulated and applied faults. It is important to notice that using the proposed algorithm, the fault detection can be achieved without knowing the physical model of the system. The simulation and experiment are conducted on the assumption that the controller does not know the physical model of the system at all; instead, a NN is used to replace the physical robot model. This is in contrast to other fault detection algorithms based on adaptive NN, H-infinity or generalized momenta, which requires a model based observer for residual generation or system output prediction. This feature provides the proposed algorithm several advantages. First, the computational load can be reduced considerably, especially for a higher DOF system, as real time computation of robot dynamics is not needed. Also, the proposed algorithm can effectively cope with model uncertainties or parameters which are difficult to estimate and time variant, such as friction coefficient or sensor bias, as NN, which replaces the robot dynamic model, is continuously updated in real time. In addition to that, as the residual is in the form of filtered fault torque, the proposed algorithm is robust against sensor noises as well. Furthermore, the proposed algorithm does not require acceleration information, which is often noisy and difficult to estimate in practice.

On the other hand, learning capability of the algorithm may prevent the algorithm from detecting slow faults, as NN would learn the fault before it can detect it. This is a common drawback for many learning based fault detection algorithms. However, it is reasonable to assume that most faults would induce a sudden change in the robot model, which the proposed algorithm can effectively detect, as shown in the simulation and experimental results. Possible solutions to detect slow faults are (1) the use of offline trained NN, (2) the combination of multiple fault detection algorithms with a statistics-based switching rule, or (3) the combination of an offline trained NN with a real time updating NN. However it is often difficult to set up an appropriate switching or merging rule for the results from different fault detection algorithms. Also, these methods inevitably increase the computational load. The target application of the proposed algorithm is robot manipulators, which consist of multiple joints; to keep the computational load at a reasonable level, the use of large offline-trained NN or multiple detection algorithms are not considered in this study.

It should be noted from the results presented in Figs. 6, 7, 13 and 15 that the proposed algorithm can detect the magnitude, direction and shape of the given faults. This enables systems to effectively identify and react to various types of faults. For example, in case of a collision, as the magnitude and direction of the collision are known, the robot can be controlled to move away from the collision. In this study, a simple bang-bang trajectory was used. However, more sophisticated method can be also applied, if needed. Furthermore, for a multi-DOF system, the location of the collision can be also estimated; it is the last link with the residual greater than the threshold.

The delay time for the window function, t_{window}, is set to 0.3 second and 1.5 seconds for the simulation and experiments, respectively. However, this is a conservative choice and as can be noted from the results, the algorithm did not take much time to sufficiently update its gains and biases. This is also the case when the NN update is resumed after a fault. When the NN update is resumed, the NN experiences a sudden change in the robot state with respect to the current weights and biases, which causes the residual to vibrate with a large error for a short period of time. The window function ensures that no malfunction will occur during such a period.

In this study, the experiments have been done using a single DOF system. However, the same algorithm can be easily expanded to a multi-DOF system, as shown in the simulation. For an actual implementation to a multi-DOF system, to reduce the computational load, it is advisable to

keep the size of NN as small as possible. Thus, starting with a small NN, one should adjust the network size, learning rate and the momentum coefficient which yields stable, small residual below the desired threshold during operations. Setting the learning rate too high is not advisable; it would cause the network to quickly learn the fault before it can be detected.

7 Conclusions

FDI can greatly improve the reliability of robotic systems. The ability of detecting faults, such as actuator malfunctions or collisions, would prevent robot damages and human injuries, and increase the robot productivity and efficiency. In this study, a novel NN based FDI algorithm is proposed, which does not require the complex system dynamic model and inaccurate acceleration. The algorithm is computationally efficient, and can be easily implemented to existing systems as it does not require additional sensors or knowledge of the system physical dynamics. The feasibility of the algorithm is verified through simulations and experiments.

As the future work, the performance of the proposed algorithm will be tested on a multi-DOF robot. Also, the problems of fault isolation and slow fault will be investigated.

Acknowledgements This research was supported by Korea Electrotechnology Research Institute(KERI) Primary research program through the National Research Council of Science & Technology(NST) funded by the Ministry of Science, ICT and Future Planning (MSIP) (No. 17-12-N0101-22)

References

1. Dixon, W.E., Walker, I.D., Dawson, D.M., Hartranft, J.P.: Fault detection for robot manipulators with parametric uncertainty: a prediction error based approach. In: Proceeding of the IEEE International Conference on Robotics and Automation, pp. 3628–3634 (2000)
2. Luca, A.D., Schaffer, A.A., Hadaddin, S., Hirzinger, G.: Collision detection and safe reaction with the DLR-III lightweight manipulator arm. In: Proceeding of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1623–1630 (2006)
3. Luca, A.D., Mattone, R.: Actuator failure detection and isolation using generalized momenta. In: Proceeding of the IEEE International Conference on Robotics and Automation, pp. 634–639 (2003)
4. Lee, S.D., Song, J.B.: Sensorless collision detection based on friction model for a robot manipulator. Int. J. Precise. Eng. Man **17**(1), 11–17 (2016)
5. Caldas, A., Makarov, M., Grossard, M., Ayerbe, P.R., Dumur, D.: Adaptive residual filtering for safe human-robot collision detection under modeling uncertainties. In: Proceeding of the IEEE/ASME International Conference on Advanced Mechatronics, pp. 722–727 (2013)
6. Ferrari, R.M.G., Parisini, T., Polycarpou, M.M.: A robust fault detection and isolation scheme for a class of uncertain input-output discrete-time nonlinear system. In: American Control Conference, pp. 2804–2809 (2008)
7. Capisani, L.M., Ferrara, A., Loza, A.F.D., Fridman, L.M.: Manipulator fault diagnosis via higher order sliding mode observers. IEEE. T. Ind. Electron **59**(10), 3979–3986 (2012)
8. Mondal, S., Chakraborty G., Bhattacharyya, K.: Unknown input high gain observer for fault detection and isolation of uncertain systems. Eng. Lett. **17**(2), 121–127 (2009)
9. Luca, A.D., Mattone, R.: An adapt-and-detect actuator FDI scheme for robot manipulators. In: Proceeding of the IEEE International Conference on Robotics and Automation, pp. 4975–4980 (2004)
10. Jung, S., Hsia, T.C.: Neural network inverse control techniques for PD controlled robot manipulator. Robotica **18**, 305–314 (2000)
11. Jung, S., Hsia, T.C.: Neural network impedance force control of robot manipulator. IEEE. T. Ind. Electron **45**, 451–461 (1998)
12. Ziang, Z.H., Ishida, T., Sunawada, M.: Neural network aided dynamic parameter identification of robot manipulators. In: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, pp. 3298–3303 (2006)
13. Bingul, Z., Ertunc, H.M., Oysu, C.: Comparison of inverse kinematics solutions using neural network for 6R robot manipulator with offset. ICSC Congress Comput. Intell. Methods Appl. https://doi.org/10.1109/CIMA.2005.1662342 (2005)
14. Lewis, F.L., Yesildirak, A., Jagannathan, S.: Neural Network Control of Robot Manipulators and Nonlinear Systems. Taylor & Francis, Inc., Bristol (1998)
15. Vemuri, A.T., Polycarpou, M.M., Diakourtis, S.A.: Neural network based fault detection in robotic manipulators. IEEE T. Robotic. Autom. **14**(2), 342–348 (1998)
16. Vemuri, A.T., Polycarpou, M.M.: A methodology for fault diagnosis in robotic systems using neural networks. Robotica **22**(4), 419–438 (2004)
17. Terra, M.H., Tinos, R.: Fault detection and isolation in robotic manipulators via neural network. J. Field Robot **18**(7), 357–374 (2001)
18. Eski, I., Erkaya, S., Savas, S., Yildirim, S.: Fault detection on robotic manipulators using artificial neural networks. Robot. Cim-Int. Manuf. **27**(1), 115–123 (2011)
19. Van, M., Kang, H.J.: A robust diagnosis and accommodation scheme for robot manipulators. Int. J. Control. Autom. Syst. **11**(2), 377–388 (2013)
20. Chen, M., Shi, P., Lim, C.-C.: Adaptive neural fault-tolerant control of a 3-DOF model helicopter system. IEEE Trans. Syst. Man. Cybern. Syst. **46**(2), 260–270 (2016)
21. Abbaspour, A., Yen, K.K., Noei, S., Sargolzaei, A.: Detection of fault data injection attack on UAB using adaptive neural network. Procedia Comput. Sci. **95**, 193–200 (2016)
22. Abbaspour, A., Aboutalebi, P., Yen, K.K., Sargolzaei, A.: Neural adaptive observer-based sensor and actuator fault detection in nonlinear systems: application in UAV. ISA Trans. **67**, 317–329 (2017)
23. Siciliano, B., Sciavicco, L., Villani, L., Oriolo, G.: Robotics Modelling, Planning and Control. Springer, London (2009)
24. Chan, S.P.: A disturbance observer for robot manipulators with applications to electronic components assembly. IEEE Trans. Ind. Electron **42**(5), 487–493 (1995)
25. Lee, K., Lee, C.-H., Hwang, S., Choi, J., Bang, Y.: Power-assisted wheelchair with gravity and friction compensation. IEEE Trans. Ind. Electron **63**(4), 2203–2211 (2016)
26. EtherCAT technology group: IEC 61800-7 ETG implementation guideline for their cia402 drive profile (2007)

Chang Nho Cho received MS degree in Mechanical Engineering from Korea University, South Korea in 2012. Now he works at Korea Electrotechnology Research Institute. His current research interests include control of robots, machining tools and other dynamic systems.

Ji Tae Hong received PhD degree in Electrical Engineering from Pusan University, South Korea in 2012. Now he works at Korea Electrotechnology Research Institute. His current research interests include high speed synchronous digital communication and control of servo motors.

Hong Ju Kim received PhD degree in Control Engineering from Pohang University, South Korea in 2003. Now he works at Korea Electrotechnology Research Institute. His current research interests include machining tools and smart manufacturing.