Chapter 7

A Real-Time Bilateral Teleoperation Control System over Imperfect Network

Truong Quang Dinh, Jong Il Yoon,
Cheolkeun Ha and James Marco

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/63033

Abstract

Functionality and performance of modern machines are directly affected by the implementation of real-time control systems. Especially in networked teleoperation applications, force feedback control and networked control are two of the most important factors, which determine the performance of the whole system. In force feedback control, generally it is necessary but difficult and expensive to attach sensors (force/torque/pressure sensors) to detect the environment information in order to drive properly the feedback force. In networked control, there always exist inevitable random time-varying delays and packet dropouts, which may degrade the system performance and, even worse, cause the system instability. Therefore in this chapter, a study on a real-time bilateral teleoperation control system (BTCS) over an imperfect network is discussed. First, current technologies for teleoperation as well as BTCSs are briefly reviewed. Second, an advanced concept for designing a bilateral teleoperation networked control (BTNCS) system is proposed, and the working principle is clearly explained. Third, an approach to develop a force-sensorless feedback control (FSFC) is proposed to simplify the sensor requirement in designing the BTNCS, while the correct sense of interaction between the slave and the environment can be ensured. Fourth, a robust-adaptive networked control (RANC)-based master controller is introduced to deal with control of the slave over the network containing both time delays and information loss. Case studies are carried out to evaluate the applicability of the suggested methodology.

Keywords: bilateral teleoperation, real-time, network, control, feedback, classification

1. Introduction

Teleoperation, which allows a human operator to interact with the environment remotely, extends humans' sensing, decision making, and operation beyond direct physical contact. Since its introduction in the late 1940s, teleoperation systems have been deployed worldwide in numerous domains ranging from space exploration, underwater operation, and hazardous assignment to micro-assembly, and minimally invasive surgery.

In common, control schemes for teleoperation systems can be classified as either compliance control or bilateral control. In the compliance control [1, 2], the contact force sensed by the slave device is not reflected back to the operator, but is used for the compliance control of the slave device. On the contrary, in the bilateral control [3–5], the contact force is reflected back to the operator. The operator is able to achieve physical perception of interactions at the remote site similar to as directly working at this site. Consequently, it improves the accuracy and safety in teleoperation. Thus, the bilateral control has drawn a lot of attention.

Figure 1 shows a generic configuration of a bilateral teleoperation system which includes five components: operator, master, communication network, slave, and environment. The master is capable of acquiring information about the desired manipulation actions and assigning the tasks to the slave. It normally consists of an input component, such as a joystick, console, or tactile device, and a force-reflecting mechanism (FRM) to exert the reflected forces on the human operator. The slave is a robotic device that takes the place of the human operator to carry out the required tasks at the remote environment. It is usually equipped with sensors to acquire information about the task process to feed back to the master.

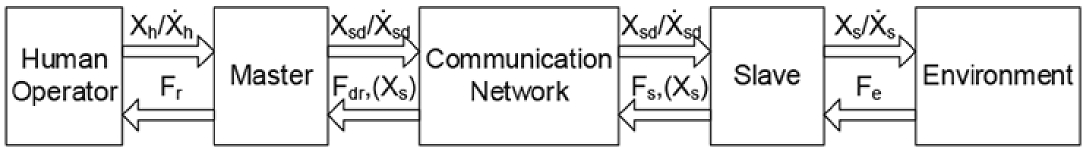

Figure 1. Generic configuration of a bilateral teleoperation system.

There are two common control architectures of bilateral teleoperation systems: position–position and position–force architectures. In the first approach, the master position is passed to the slave device, and the slave position is passed back to the master side. The reflected force applied to the operator is derived from the position difference between the two devices and, therefore, this approach is not desirable in cases of free motion. In contrast, the position–force approach uses directly the force measured at the remote site rather than the position error. In this architecture, the contact force, sensed by a force/torque sensor mounted on the slave device, is scaled by a force-reflecting gain (FRG), and this scaled force is reflected back to the operator via the master device. This method then provides the operator a better perception of tasks execution at the remote site.

In order to derive sufficiently the FRG, two important tasks are required: first, to detect the environment and, second, to determine the contact force at the slave site. Many studies have been carried out to optimize the FRG [4, 5]. Although the reported algorithms showed some remarkable results, there remain some drawbacks such as how to determine the FRG appropriately with unknown environments; and especially, it is difficult and expensive to attach proper sensors (force/torque/pressure sensors) to detect the loading conditions. Additionally, the use of these kinds of sensors is cost-ineffective and difficult to be installed in practice. Especially, it is easy to be damaged when the system operates under hazard conditions. Incorporation of teleoperation and force feedback requires bidirectional information exchange between the master and slave via the network. In contrast to the advantages such as cost saving, power consumption, easy implementation, and maintenance, a networked control system (NCS) leads to two major problems, inevitable random time-varying delays and packet dropouts because of restrictions imposed by the transmission data rate and channel bandwidth. Due to sensitivity of bilateral teleoperation systems to time delays and packet dropouts, even a small time delay can destabilize the system [6].

Disregarding the packet loss problem, numerous methods have been proposed [7–9] to minimize the bad effect of time delay. Herein, the controllers were designed based on the assumptions that time delay was constant [7], bounded [8], or had a probability distribution function [9]. Moreover, these assumptions just considered that the time delay in the closed control loops was less than one sampling period while in practice, it is random and irregular. To compensate large and uncertain delays, other studies suggested adaptive control schemes using variable sampling periods, which were based on neural networks (NNs) or prediction theories [10–12]. Although the performances were improved, there were requirements on acquiring the real delay data for the training processes, which were not appropriately discussed.

To adapt with NCSs compromising not only time delays but also packet dropouts, many important methodologies, such as state feedback control [13], robust control [14–16], predictive and model-based predictive control [17, 18], were proposed. By using these techniques, the NCS performances were remarkably improved over the traditional methods. However in most studies, time delays and packet dropouts were assumed to be priorly known and bounded to design the controllers. The sampling period was always fixed in these studies and, subsequently, limited the control performances. Furthermore, computation delay normally at the master side is also one important factor affecting directly the system performance. Recently, an advanced robust variable sampling period control approach has been developed for nonlinear systems containing both the kinds of delays and packet dropouts [19–21]. The applicability of this method was proved through real-time experiments.

Therefore to fully overcome the above-mentioned problems, a simple and cost-effective real-time bilateral teleoperation networked control system (BTNCS) is introduced in this chapter. The advanced concept for designing a bilateral master-slave teleoperation networked control system is proposed, and the working principle is clearly addressed. Next, an approach to develop a force-sensorless feedback control (FSFC) is proposed to simplify the sensor requirement in designing the BTNCS while the correct sense of interaction between the slave and the environment can be ensured. To deal with the networked control of the slave, a robust-adaptive networked control (RANC)-based master controller is suggested to compensate for the problems of time delays and information loss. Case studies are carried out to evaluate the applicability of the suggested methodology.

2. Bilateral teleoperation networked control system

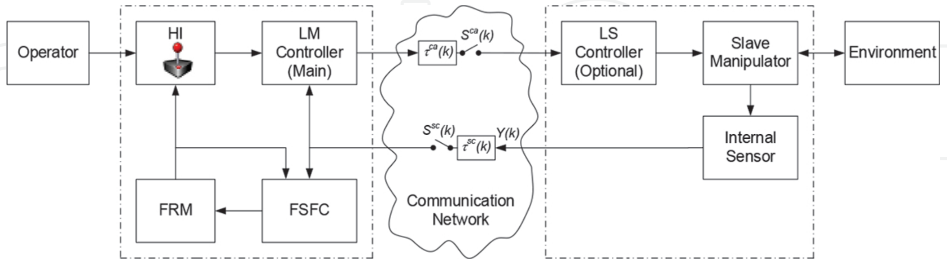

Figure 2. Proposed architecture of the BTNCS.

Without loss of generality, the architecture of the BTNCS is suggested in **Figure 2**. The design for this BTNCS should address the following issues:

- The local master (LM) controller functions as the main control unit of the system to ensure that the slave manipulator could execute robustly and accurately the tasks given by the operator via a human interface (HI), disregarding the impacts of networked communication problems and environments. The local slave (LS) controller is, therefore, an optional design. It can be just a buffer or a zero-order holder (ZOH) to receive the control input derived by the LM controller and distribute sequentially to the manipulator.
- There is no force/pressure/torque sensor to be attached at the manipulator end to minimize the cost and risks. Only an internal sensor, as displacement sensor, is used to monitor the manipulator trajectory and send back to the LM controller to form a closed control loop.
- The interaction between the manipulator and environment is, therefore, estimated by the FSFC module at the master side. Here, the FSFC should consist of two parts: first, an environmental interaction (EI) estimator to detect the environment characteristics as well as to derive properly a reflecting force required to be applied to a physical device in the HI module (as a joystick); second, an FRM controller to drive the FRM to generate the desired reflecting force.
- The FRM is suggested to be constructed using pneumatic rotary actuators. This use brings some advantages over the traditional design with DC electric motors. Compared with an electric motor, a pneumatic actuator provides a higher ratio of force-mass, and can produce larger reflected force without using any reduction mechanism, such as gearbox. In addition, with the pneumatic solution, the FRM is able to work under safe conditions without damage from the operator.
- Due to having different control modules and other functions at the master side, computation delays τ^{com} need to be taken into account when designing the LM controller.

- The communication network has unavoidable delays, τ^{ca} and τ^{sc}, and packet dropouts, represented by "virtual" switches, S^{ca} and S^{sc}, in the forward and backward channels, respectively. S^{ca} (S^{sc}) is opened (or 1) when a packet loss event exits.

The following are attempted to address the two important issues in designing the BTNCS which are the FSFC and the LM controller.

3. Force-sensorless feedback control

3.1. Design of FSFC

As stated in Section 2, the FSFC compromises the two main modules: EI estimator and FRM controller. The FRM controller can be selected as a typical feedback controller in which the reference input is the desired reflecting force sent from the environment classifier and the feedback signal can be the force/torque/pressure at the physical device of the HI module.

The EI estimator is suggested as in **Figure 3**. This estimator with two inputs and one output contains four blocks: Learning Vector Quantitative Neural Network (LVQNN) classifier, slave dynamics, environment dynamics, and fuzzy-based FRG tuner. Without loss of generative, the interactive environment can be represented by two factors: damping c_e and stiffness k_e. Thus, The LVQNN classifier is firstly designed with two inputs and two outputs to classify the environment. The two inputs are the command and response from the slave while the two outputs are predicted values of the environment damping and stiffness, $\hat{c}_e$ and $\hat{k}_e$, respectively. Next, these predicted values are fed into both the environment dynamics and fuzzy-based FRG tuner. Synchronously, the slave command is also input to the slave dynamics. The interaction between the slave dynamics and environment dynamics—loading force—is, therefore, estimated and denoted as $\hat{F}_e$. Meanwhile, the fuzzy-based FRG tuner is designed with two inputs, $\hat{c}_e$ and $\hat{k}_e$, and one output which is value of the FRG. Finally, the desired reflected force, F_{dr}, is derived from the estimated loading force and the FRG.

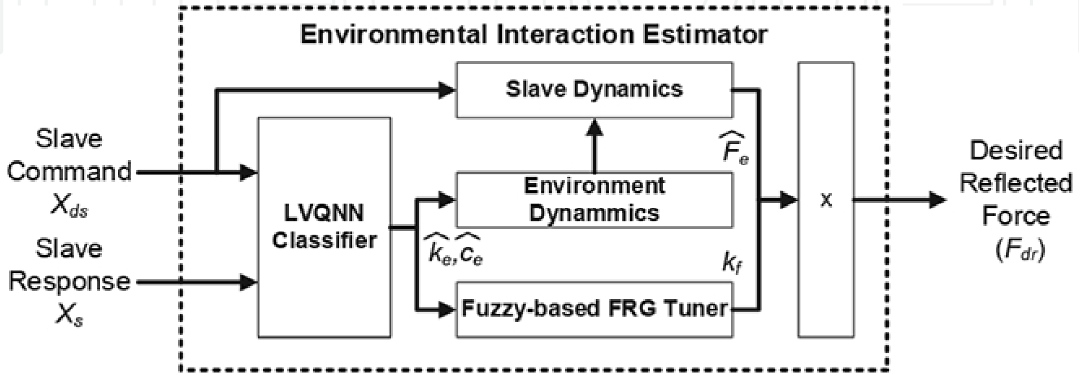

Figure 3. Configuration of the proposed environmental interaction estimator.

3.2. LVQNN classifier

3.2.1. Learning vector quantitative Neural Network

NN is one of the powerful artificial intelligent techniques that emulates the activity of biological NNs in the human brain. LVQNN is a hybrid network which uses advantages of competitive learning and bases on Kohonen self-organizing map or Kohonen feature map to form the classification [22].

Figure 4 shows a generic structure of an LVQNN with four layers: one input layer with m nodes, first hidden layer named competitive layer with S_1 nodes, second hidden layer named linear layer with S_2 nodes, and one output layer with n nodes (in this case, $S_2 \equiv n$).

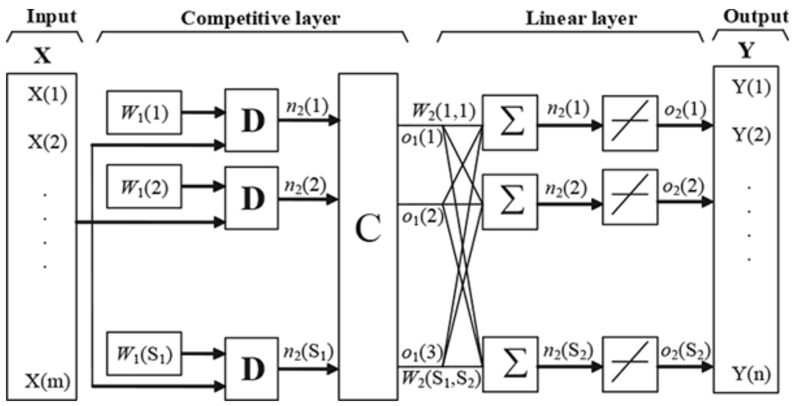

Figure 4. Structure of the LVQNN.

The core of the LVQNN is based on the nearest-neighbor method by calculating the Euclidean distance weight function, D, for each node, n_j, in the competitive layer as in the following:

$$n_j = D\left(X, W_1(j)\right) = \sqrt{\sum_{i=1}^{m}\left(X(i) - W_1(j,i)\right)^2}, \; j = 1,..,S_1 \tag{1}$$

where X is the input vector; $W_1(j,i)$ is the weight of node j^{th} in the competitive layer corresponding to element i^{th} of the input vector.

Next, the Euclidean distances are fed into function C which is a competitive transfer function. This function returns an output vector o_1, with 1, where each net input vector has its maximum value, and 0 elsewhere. This vector is then input to the linear layer to derive an output vector o_2, where each element corresponded to each node of the output layer and computed as

$$Y(k) = o_2(k) = k_W(k) n_2(k) = k_W(k) \sum_{j=1}^{S_1} W_2(k,j) o_1(j), \; k = 1,..,n, (n \equiv S_2) \tag{2}$$

where $W_2(k,j)$ is the weight of node kth in the linear layer corresponding to element jth of the competitive output vector; $k_W(k)$ is linearized gain of node kth in the linear layer.

In the learning process, the weights of LVQNN are updated by the well-known Kohonen rule, which is shown in the following equation:

$$\begin{cases} W_1^{t+1}(j) = W_1^t(j) + \mu\left(X - W_1^t(j)\right) \text{ IF: } X \text{ is classified correctly} \\ W_1^{t+1}(j) = W_1^t(j) - \mu\left(X - W_1^t(j)\right) \text{ IF: } X \text{ is classified incorrectly} \end{cases}, j = 1,..,S_1 \tag{3}$$

where μ is the learning ratio with positive and decreasing with respect to the number of training iterations ($n_{iteration}$), $\mu = n_{iteration}^{-1}$.

3.2.2. Design of LVQNN classifier

Here, the LVQNN classifier is designed and implemented into the FSFC to distinguish different working environments at the slave side in an online manner. To enhance the given task with the limited number of input information, the input vector of the classifier is constructed as a vector of current and several historical values of four signals: the slave driving command, $\{X_{ds}^{(0)}, X_{ds}^{(-1)}, \dots, X_{ds}^{(-g)}\}$, slave response, $\{X_s^{(0)}, X_s^{(-1)}, \dots, X_s^{(-p)}\}$, and their derivatives, $\{dX_{ds}^{(0)}, dX_{ds}^{(-1)}, \dots, dX_{ds}^{(-h)}\}$and $\{dX_s^{(0)}, dX_s^{(-1)}, \dots, dX_s^{(-q)}\}$, respectively, while the final outputs are the environment damping and stiffness, $\hat{c}_e$ and $\hat{k}_e$.

For applications to classify the environment which is varied with both the damping and stiffness values, the output from the classifier should be the mix of classes with different ratios. In order to enhance this task, a so-called smooth switching algorithm is proposed here. The environment class is, therefore, determined by the smooth combination of the current class detected by the LVQNN (Y) and the previous class using a forgetting factor, λ

$$class(t) = \lambda \times class(t-1) + (1-\lambda) \times Y(t) \tag{4}$$

Similarly, the estimated values of the damping coefficient and stiffness are produced by

$$\begin{cases} \hat{c}_e(t) = \lambda \times \hat{c}_e\big|_{class(t-1)} + (1-\lambda) \times \hat{c}_e\big|_{Y(t)} \\ \hat{k}_e(t) = \lambda \times \hat{k}_e\big|_{class(t-1)} + (1-\lambda) \times \hat{k}_e\big|_{Y(t)} \end{cases} \tag{5}$$

Additionally, in order to avoid influences of noises on the classification performance, the forgetting factor is online tuned with respect to the changing speed of the classifier outputs:

Step 1: Set the initial value for the forgetting factor, λ=0.5; define a small positive threshold, $0<\gamma_1<\gamma_2$, for the classifier output changing speed, v_Y, which is defined by the number of sampling periods when Y continuously changes.

Step 2: For each step, check v_Y and update λ by comparing v_Y with γ using the following rule:

+ If $v_Y=0$, Then $\lambda(t+1)=\lambda(t)$;

+ Else If $(v_Y>\gamma_2)$, Then $\lambda(t+1)=\lambda(t+1)/2$ and reset $v_Y=0$;

+ Else If $(v_Y\ge\gamma_1)\&(v_Y(t)\le\gamma_2)$, Then $\lambda(t+1)=\lambda(t+1)\times 2$ and reset $v_Y=0$;

+ Otherwise, $\lambda(t+1)=\lambda(t)$.

3.2.3. *An illustrative example*

3.2.3.1. *Test rig setup*

To evaluate the effectiveness of the LVQNN classifier, an experimental system has been designed as in **Figure 5**. One joystick with one Degree of freedom (DOF) is used to generate driving commands for the slave. An FRM which employs a valve-driven mini pneumatic rotary actuator and two bias springs is attached to the opposite side of the joystick handle. The slave employs an asymmetrically pneumatic cylinder as its manipulator. The displacement of the cylinder rod is sensed by a linear variable displacement transducer (LVDT). The interaction between the master and slave is enhanced by the communication module which can perform either wire-by-wire or wireless protocol. To generate environmental conditions for the slave manipulator, compression springs with different stiffness values are installed in serial with the cylinder rod. An air compressor is used to supply the pressurized air for both the FRM and the slave. A compatible PC and a multifunction data acquisition device are employed to perform the communication between the PC and master. The system specifications are listed in **Table 1**.

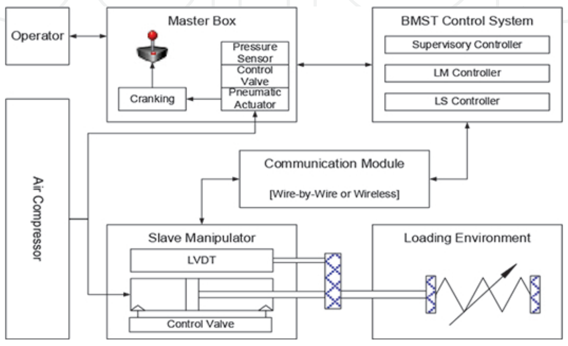

Figure 5. Design layout for the experimental BMST system.

Parts	Type	Component characteristics
Rotary actuator	CRB1BW15 90-D	Max. torque: 0.9 Nm
Pressure sensors	SDE1-D10-G2-W18	Pressure range: 0–10 bar
Pneumatic cylinder	CDC-20	Stroke: 100 mm, bore: 20 mm, rod: 8 mm
Servo valves	MPYE-5-1/4-010B	Control voltage range: 0–10 VDC
LVDT	Novotechnik TR100	Measurement range: 0–100 mm
Springs	Case 1, Case 2, Case 3	Randomly selected

Table 1. Specifications of the system components.

Input number	Number of nodes in the hidden layer				
	20	25	30	35	40
20	80.35	80.50	81.48	82.17	81.29
24	81.14	81.19	82.26	81.48	82.78
28	80.61	80.03	81.66	81.12	81.69
32	75.34	80.64	85.64	81.00	81.80
36	81.03	81.58	81.93	81.85	80.49
40	79.30	80.98	80.90	80.13	80.22
	Goodness of fit [%]				

Table 2. Learning success rate of the LVQNN classifier [%].

3.2.3.2. LVQNN classifier training and verification

In order to train the network, the prior task is acquiring the target data. Real-time teleoperation experiments without force feedback concept were performed on the test rig. Both the three springs mentioned in Table 1 were used to generate the environmental conditions. For each condition, a trajectory for the slave manipulator was randomly given by the operator. The slave information, including the valve driving command and cylinder rod displacement, with respect to each spring was acquired with a sampling period of 0.02 s.

Next, each of the acquired slave data sets was used to perform the input vector, while the correspondingly selected spring was used as the target output class (1, 2, or 3). To investigate performance of the LVQNN classifier with respect to different structures, several trainings were performed with the selected data set by varying the number of inputs from 20 to 40, and the number of hidden neurons was changed from 20 to 40. After the training process, the results (goodness of fit [%]) of the LVQNN are analyzed in **Table 2**. It shows that the most suitable LVQNN structure was realized with 32 nodes in the input vector and 30 nodes in the competitive layer. The learning success rate in this case was highest with 85.64 [%].

Real-time teleoperation experiments were performed in order to investigate the ability of the LVQNN classifier in practice. Other three experiments with the three loading conditions were carried out with the same cylinder trajectories given from the master using an open-loop control. The classification results were then achieved as displayed in **Figure 6–8**. The results imply that the proposed classifier could online detect well the loading conditions and, therefore, is capable of producing precisely decisions on the environment characteristics.

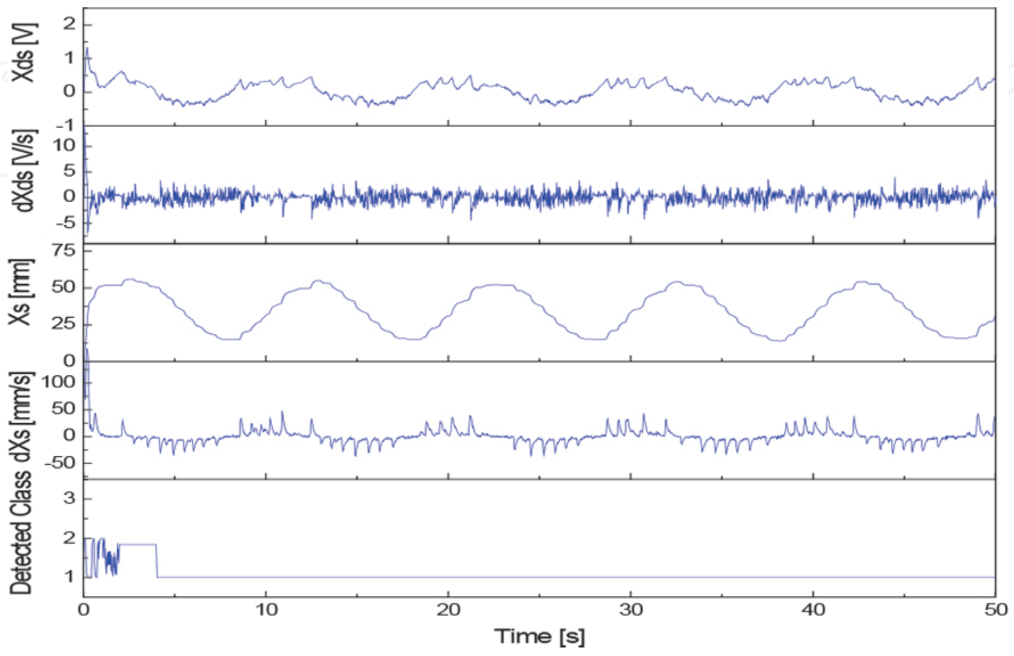

Figure 6. Real-time classification of the optimized LVQNN with respect to spring case 1.

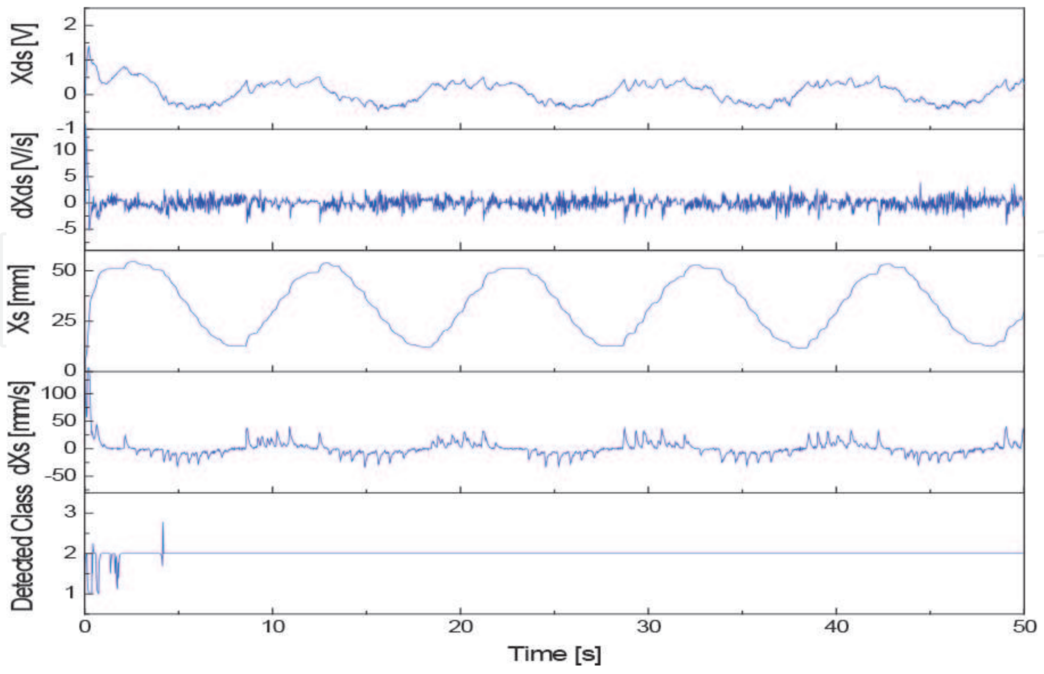

Figure 7. Real-time classification of the optimized LVQNN with respect to spring case 2.

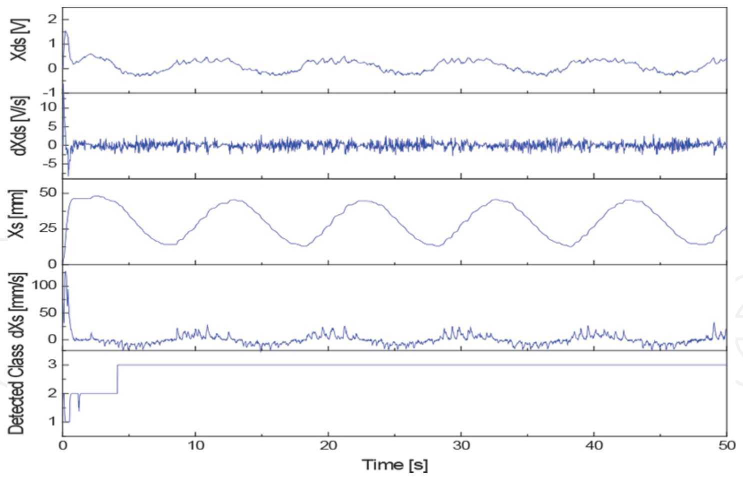

Figure 8. Classification performance of the optimized LVQNN with respect to spring case 3.

4. Robust-adaptive networked control

4.1. Problem and RANC design concept

In this section, the RANC approach for a generic system is introduced to support the design of the LM controller. Consider a discrete-time plant [20, 21]:

$$\begin{cases} x_{k+1} = Ax_k + Bu_k + Ed_k \\ y_k = Cx_k \end{cases} \tag{6}$$

where $x_k \in R^n$ is the state vector, $u_k \in R^m$ is the control input, $y_k \in R^p$ is the controlled output, $d_k \in R^d$ is the environment impact, and A, B, C and E are matrices with appropriate dimensions.

The RANC-based LM controller in **Figure 2** is then designed based on the following issues [20, 21]:

- Both the actuators and controller are event-driven.
- The sensors are time-driven with variable sampling period T. For step $(k+1)^{\text{th}}$, this sampling period T_{k+1} is online optimized depending on the total system time delay τ_{k+1} to make sure:

$$\begin{cases} \tau_{k+1} = \tau_{k+1}^{com} + \tau_{k+1}^{ca} + \tau_{k+1}^{sc} \\ T_{k+1} \geq \tau_{k+1} > 0 \end{cases} \tag{7}$$

- Sets of continuous packet dropouts $\{p_d\}$ are online detected and bounded by $\bar{p}_{k+1}$:

$$\bar{p}_{k+1} = \sup(p_d), 1 \le d < k+1 \tag{8}$$

Using the results in references [20, 21], the configuration of an NSC using the proposed RANC controller is clarified in **Figure 9**. Herein, the RANC mainly consists of five modules: Time Delay and Packet Detector (TDPD), Time Delay Predictor (TDP), Variable Sampling Period Adjuster (VSPA), Quantitative Feedback Theory (QFT), and Robust State Feedback Controller (RSFC). The TDPD module is firstly used to detect the network problems at the current state. This information is then sent to the TDP to perform one-step-ahead prediction of system delays which are the inputs of the VSPA to adjust effectively the sampling period. The two modules, QFT and RSFC, are employed to construct the so-called hybrid controller to compensate for the influences of delays and packet dropouts. A smart switch (SSW) is employed to switch the hybrid controller to QFT or RSFC based on the outputs of the TDPD detector and the TDP predictor with rule:

- The QFT is selected (SSW = 0) once there is no packet loss and all delay components are less than their pre-defined threshold values: ($\lceil * \rceil$is ceiling function to return the nearest integer)

$$\bar{\tau}_{QFT}^{com} = \lceil \tau_k^{com} \rceil, \bar{\tau}_{QFT}^{ca} = \lceil \tau_k^{ca} \rceil, \bar{\tau}_{QFT}^{sc} = \lceil \tau_k^{sc} \rceil, \bar{\tau}_{QFT} = \lceil \tau_k \rceil, \forall k \tag{9}$$

- Otherwise, the RSFC is chosen (SSW = 1).

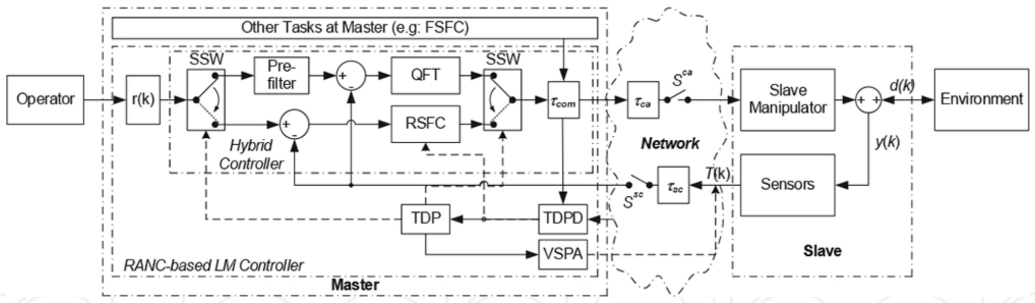

Figure 9. Configuration of networked LM controller using the RANC approach.

4.2. Design of RANC components

4.2.1. Design of VSPA

By using the TDPD and TDP, the sampling period for the coming step ($k + 1$) is defined as

$$\begin{aligned} &T_{k+1} = \lceil T_{new} / T_0 \rceil T_0 = k_T T_0,\ k_T = \lceil T_{new} / T_0 \rceil = 1, 2, \ldots \\ &T_{new} = \max(T_0, \hat{\tau}_{k+1}^{com} + \hat{\tau}_{k+1}^{ca} + \hat{\tau}_{k+1}^{sa}) = \max(T_0, \hat{\tau}_{k+1}), T_0 >= 0.1 T_p,\ (T_0 \text{is initial sampling period}) \end{aligned} \tag{10}$$

where $\hat{\tau}_{k+1}^{com}$, $\hat{\tau}_{k+1}^{ca}$ and $\hat{\tau}_{k+1}^{sa}$are the delays estimated by the TDP.

4.2.2. Design of TDPD

To measure accurately time delays and packet dropouts, the TDPD employs a micro-control unit (MCU) with proper logic. Here, PIC18F4620 MCU from microchip equipped with a 4 MHz oscillator is suggested to be used [19]. The real-time measurement accuracy of 50μs[19] which is suitable for this application. For each step k^{th}, the TDPD enhances the following tasks:

- Derive the real working time using the MCU, t_k.
- For a command u_k sent from the controller to the plant, a time stamp is encapsulated into this packet to detect the forward delay, τ_k^{ca}, and packet dropouts if having, p_k^{ca}.
- For the signals sent from the sensors to the controller, they are combined with time stamps to detect the delay, τ_k^{sc}, and packet dropouts if having, p_k^{sc}.
- The total number of continuous packet dropouts is then determined as:

$$p_k = S_k^{ca} p_k^{ca} + S_k^{sc} p_k^{sc} \tag{11}$$

- Depend on the time stamps to detect the computation delay at the controller side, τ_k^{com}.

4.2.3. Design of TDP

The TDP based on a so-called adaptive gray model with single-variable first-order, AGM (1,1) to estimate the system delays in the coming step, $\hat{\tau}_{k+1}^{com}$, $\hat{\tau}_{k+1}^{ca}$, and $\hat{\tau}_{k+1}^{sa}$. The AGM (1,1) prediction procedure is as follows:

<u>Step 1:</u> For an object with a data sequence
$\{y_{Object}(t_{O1}),\ y_{Object}(t_{O2}),\ \ldots,\ y_{Object}(t_{Om})\}$ ($m \geq 4$), the raw input gray sequence representing for the object data is derived as

$$y_{raw}^{(0)} = \left\{ y_{raw}^{(0)}(t_1), y_{raw}^{(0)}(t_2), \ldots, y_{raw}^{(0)}(t_n) \right\};\ \Delta t_k = t_k - t_{k-1};\ k = 2,..,n \geq 4; \tag{12}$$

Note: Eq. (12) is obtained from Eq. (14) if condition (13) satisfies; otherwise, it is obtained from Eq. (15).

$$\begin{aligned} &\Delta t_{Oi} / T_{\text{TDP}} < 2;\ \Delta t_{Oi} = t_{Oi} - t_{O(i-1)};\ \forall i \in [2,\ldots,m]; T_{\text{TDP}} \text{ is the prediction sampling period} \\ &\Delta t_{O1} = t_{O1} - t_{Olast}; t_{Olast} : \text{time of previous value of } y_{Object}(t_{O1}) \end{aligned} \tag{13}$$

$$y_{raw}^{(0)}(t_k) = y_{Object}(t_i);\ t_k = t_{Oi};\ k = 1,\dots,n;\ n = m \tag{14}$$

$$\begin{aligned} & y_{raw}^{(0)}(t_1) = y_{Object}(t_1);\ t_k = t_{k-1} + T_{\mathrm{TDP}} \\ & y_{raw}^{(0)}(t_k) = Sa_i + Sb_i(t_k - t_{Oi})^1 + Sc_i(t_k - t_{Oi})^2 + Sd_i(t_k - t_{Oi})^3 \\ & t_{Oi} \le t_k \le t_{O(i+1)};k = 2,\dots,n;\ n = \lfloor (t_{Om} - t_1)/T_{TDP} \rfloor;i = 1,\dots,m-1 \end{aligned} \tag{15}$$

where $\{Sa_i, Sb_i, Sc_i, Sd_i\}$ (i=1, …, m−1)is a [4 × m – 1] coefficient matrix of the spline function going through the object data set $\{(t_1, y_{Object}(t_1)), \dots, (t_m, y_{Object}(t_m))\}$.

Theorem 1: There always exist two non-negative additive factors, c_1 and c_2, to convert any raw sequence (12) to a gray sequence which satisfies both the gray checking conditions.

Proof: The proof of this theorem can be found in reference [19].

Thus from (12), the gray sequence is derived using *Theorem* 1:

$$y^{(0)} = \left\{ y^{(0)}(t_1), y^{(0)}(t_2),\dots, y^{(0)}(t_n) \right\} > 0;\ y^{(0)}(t_k) = y_{raw}^{(0)}(t_k) + c_1 + c_2;\ k = 1,\dots,n \tag{16}$$

Step 2: Use the 1-AGO to obtain a new series $y^{(1)}$ from $y^{(0)}$:

$$y^{(1)}(t_k) = \sum_{i=1}^{k} y^{(0)}(t_i) \times \Delta t_k = \sum_{i=1}^{k} \left(y_{raw}(t_i) + c_1 + c_2 \right) \times \Delta t_k \tag{17}$$

Step 3: Create the background series $z^{(1)}$ from $y^{(1)}$ by the following general algorithm:

$$z^{(1)}(t_k) = \alpha(t_k) y^{(1)}(t_k) + (1 - \alpha(t_k)) y^{(1)}(t_{k-1});k = 2,\dots,n \tag{18}$$

$$\alpha(t_k) = \beta\alpha_{aver} + (1-\beta)\alpha_{adapt}(t_k) \tag{19}$$

$$\alpha_{adapt}(t_k) = \begin{cases} 1,\ \mathrm{IF}: s(t_k) \ge \sum_{i=1}^{k+1} \Delta t_i / \sum_{i=1}^{n} \Delta t_i \\ s,\ \mathrm{IF}: \sum_{i=1}^{k-1} \Delta t_i / \sum_{i=1}^{n} \Delta t_i < s(t_k) < \sum_{i=1}^{k+1} \Delta t_i / \sum_{i=1}^{n} \Delta t_i \\ 0,\ \mathrm{IF}: s(t_k) \le \sum_{i=1}^{k-1} \Delta t_i / \sum_{i=1}^{n} \Delta t_i \end{cases} \tag{20}$$

$$s(t_k) = \log\left(y^{(0)}(t_k) / y^{(0)}(t_1) \right) / \log\left(y^{(0)}(t_n) / y^{(0)}(t_1) \right) \tag{21}$$

where α_{aver} is the average weight and set as 0.5 [23, 24]; β is a momentum rate within range [0, 1] and tuned by a Lyaponov-based SISO fuzzy mechanism (LFM) in order to guarantee a robust prediction performance (see the proof of this theory in reference [20])

Step 4: Establish the gray differential equation:

$$y^{(0)}(t_k) + az^{(1)}(t_k) = b \tag{22}$$

$$\hat{\beta}_{ab} = \begin{bmatrix} \hat{a} & \hat{b} \end{bmatrix}^T = \left(B^T B\right)^{-1} B^T Y; B = \begin{bmatrix} -z^{(1)}(t_2) & 1 \\ -z^{(1)}(t_3) & 1 \\ \vdots & \vdots \\ -z^{(1)}(t_n) & 1 \end{bmatrix}, Y = \begin{bmatrix} y^{(0)}(t_2) \\ y^{(0)}(t_3) \\ \vdots \\ y^{(0)}(t_n) \end{bmatrix} \tag{23}$$

Step 5: Setup the AGM (1,1) prediction as follows:

$$\hat{y}^{(0)}(t_k) = \left(\hat{b} - \hat{a}y^{(1)}(t_{k-1})\right) \Big/ \left(1 + \alpha(t_k)\hat{a}\Delta t_k\right) = \frac{\hat{b} - \hat{a}y^{(0)}(t_1)\Delta t_1}{1 + \alpha(t_2)\hat{a}\Delta t_2} \prod_{i=3}^{k} \frac{1 + \left(\alpha(t_{i-1}) - 1\right)\hat{a}\Delta t_{i-1}}{1 + \alpha(t_i)\hat{a}\Delta t_i} \tag{24}$$

Step 6: Perform the predicted value of y at step $(n + p)$th:

$$\hat{y}_{raw}^{(0)}(t_{n+p}) = \frac{\hat{b} - \hat{a}y^{(0)}(t_1)\Delta t_1}{1 + \alpha(t_2)\hat{a}\Delta t_2} \prod_{i=3}^{n+p} \frac{1 + \left(\alpha(t_{i-1}) - 1\right)\hat{a}\Delta t_{i-1}}{1 + \alpha(t_i)\hat{a}\Delta t_i} - c_1 - c_2 \tag{25}$$

where p is the step size of the grey predictor. In this case, $p = 1$.

4.2.4. Design of QFT controller

Denotes the transfer functions of the plant and the controller in the NCS as $G_p(s)$ and $G_c(s)$, respectively. The closed-loop transfer function from input $R(s)$ to output $Y(s)$ including delays can be expressed as

$$T^{\text{NCS}}(s) = \frac{Y(s)}{R(s)} = \frac{G_c(s)G_p(s)e^{-(\tau_{com} + \tau_{ca})s}}{1 + G_c(s)G_p(s)e^{-\tau s}} \tag{26}$$

And the open-loop transfer function is then derived as

$$L^{\text{NCS}}(s) = G_c(s)G_p(s)e^{-\tau s} \tag{27}$$

Next, the procedure to design this robust controller can be expressed as follows [25, 26]:

Step 1: The QFT controller should be designed on how the tracking signal meets the acceptable variation range with respect to a reference (for each value of ?i of interest)

$$T_l^{\text{NCS}}(j\omega_i) \le \left|T^{\text{NCS}}(j\omega_i)\right| \le T_u^{\text{NCS}}(j\omega_i) \tag{28}$$

Step 2: Establish bounds for robust stability of the closed-loop system

$$\left|T^{\text{NCS}}(j\omega)\right| \le M = 1.4,\ \omega \ge 0 \tag{29}$$

Step 3: A sensitivity function is defined as

$$S(j\omega) = \left(1 + L^{\text{NCS}}(j\omega)\right)^{-1},\ \omega \ge 0 \tag{30}$$

Establish bounds for noise and disturbance rejection of the closed-loop system

$$\left|1 + L(j\omega)\right|_{\max}^{-1} \le M_D(\omega),\ M_D > 1\ (M_{\text{dB}} > 0\,\text{dB}), \omega \ge 0 \tag{31}$$

Step 4: Define a working frequency range of the NCS.

Step 5: Bases on the nominal plant to design the controller, $G_{QFT}(s)$, with initial poles and zeros.

Step 6: Use the loop-shaping method to refine the controller designed in Step 5. The principle to guarantee a robust control performance is the loop gain value is always on or above the boundaries defined by constraints (28)–(31) and, is to the right or on the robustness forbidden region for any critical frequency within the range defined in Step 4.

Step 7: The controller resulted from Step 6 could ensure that the variation in magnitude of $T^{\text{NCS}}(s)$ (26) is satisfied at the desired constraints. However, it does not guarantee that the magnitude of $T^{NCS}(s)$ actually lies between the given bounds $T_l^{\text{NCS}}(j\omega_i)$ and $T_u^{\text{NCS}}(j\omega_i)$for the whole frequency range. Therefore, a pre-filter $F_{QFT}(s)$ is necessary to be designed and placed in front of the controller to compensate for this problem.

4.2.5. Design of RSFC

The RSFC is designed to deal with the NCS in cases that the delays are greater than the threshold values defined in Eq. (9), and/or there exists packet dropouts. To simplify the analysis, it can be assumed that $r_k = 0$ (**Figure 9**) during the system modeling and RSFC design. The system discrete form for step $(k + 1)^{\text{th}}$ can be expressed through the following three cases:

Case 1: There is no packet loss in network transmission during both current period and previous period:

$$x_{k+1} = A_k^0 x_k + B_k^0 u_k + B_{k-1}^1 u_{k-1} + Ed_k \tag{32}$$

where $A_k^0 = e^{AT_k}$, $B_k^0 = \int_0^{T_k - \tau_k} e^{At} B \mathrm{d}t$, $B_k^1 = \int_{T_k - \tau_k}^{T_k} e^{At} B \mathrm{d}t$.

Case 2: There exists *i* packet dropouts up to step *k*th:

$$x_{k+1} = A_k^0 x_k + B_{k-i}^2 u_{k-i} + Ed_k \tag{33}$$

where $B_k^2 = \int_0^{T_k} e^{At} B \mathrm{d}t$.

Case 3: There were p_d continuous packet dropouts just passed up to step *(k* − 1)th:

$$x_{k+1} = A_k^0 x_k + B_k^0 u_k + B_{k-p_d-1}^1 u_{k-p_d-1} + Ed_k \tag{34}$$

Since, the general discrete form of the system can be established as

$$x_{k+1} = A_k^0 x_k + B_k^0 \delta_k^0 u_k + \sum_{j=1}^{2} \sum_{i=1}^{\bar{p}_{k+1}} B_{k-i}^j \delta_{k-i}^j u_{k-i} + Ed_k \tag{35}$$

where δ^l is an activation function:

$$\delta^l = \begin{cases} 1, & \text{If subsystem } l \text{ is activated}, l = \{0,1,2\} \\ 0, & \text{Otherwise.} \end{cases} \tag{36}$$

In case of no delays, the control rule is designed as

$$u_k = Kx_k \tag{37}$$

Then, Eq. (35) can be re-written in the following form:

$$x_{k+1} = \left(A_k^0 + B_k^0\delta_k^0 K\right)x_k + \sum_{i=1}^{\bar{p}_{k+1}}\left(\sum_{j=1}^{2} B_{k-i}^j \delta_{k-i}^j\right)Kx_{k-i} + Ed_k \tag{38}$$

or

$$X_{k+1} = \Phi_k X_k + \Phi_E d_k \tag{39}$$

where $X_k = \begin{bmatrix} x_k^T & x_{k-1}^T & \cdots & x_{k-\bar{p}_{k+1}}^T \end{bmatrix}^T$, $\Phi_k = \begin{bmatrix} \begin{bmatrix} A_k^* & B_{1k}^* & \cdots & B_{(\bar{p}_{k+1}-1)k}^* \end{bmatrix} & B_{\bar{p}_{k+1}k}^* \\ I_{\bar{p}_{k+1}\times\bar{p}_{k+1}} & 0 \end{bmatrix}$,

$$\Phi_E = \begin{bmatrix} E & 0 & \cdots & 0 \end{bmatrix}^T, A_k^* = A_k^0 + B_k^0\delta_k^0 K, B_{ik}^* = K\sum_{j=1}^{2} B_{k-i}^j \delta_{k-i}^j.$$

Theorem 2: For given constants $\xi_i^j > 0$, if there exist positive definite matrices $P_i^* > 0,\ K_1 > 0,\ K_2 > 0$, $i \in [0,\ 1,\ \ldots,\ \bar{p}_{k+1}],\ j = \{0,\ 1,\ 2\}$ such that following inequalities:

$$\begin{bmatrix} \xi_i^{-2} P_i^* & K_1^* \Phi_A^T + K_2^{*T} \Phi_B^T \\ \Phi_A K_1^{*T} + \Phi_B K_2^* & K_1^* + K_1^{*T} - P_i^* \end{bmatrix} > 0 \tag{40}$$

hold, then the NCS (39) is exponentially stable with a positive decay rate by using the RSFC control gain derived by

$$K_{\text{RSFC}} = K_2 K_1^{-T} \tag{41}$$

with $K_1^* = diag[\{K_1\}],\ K_2^* = diag[\{K_2\}],\ K_{\text{RSFC}}^* = diag[\{K_{\text{RSFC}}\}]$

$$\Phi_A = \begin{bmatrix} \begin{bmatrix} A_k^0 & \cdots & 0 \end{bmatrix} & 0 \\ I & 0 \end{bmatrix}, \Phi_B = \begin{bmatrix} \begin{bmatrix} B_k^0\delta_k^0 & \cdots & \sum_{j=1}^{2} B_{k-\bar{p}_{k+1}+1}^j \delta_{k-\bar{p}_{k+1}+1}^j \end{bmatrix} & \sum_{j=1}^{2} B_{k-\bar{p}_{k+1}}^j \delta_{k-\bar{p}_{k+1}}^j \\ 0 & 0 \end{bmatrix}.$$

Proof: The proof of this theorem can be found in reference [20].

4.3. An illustrative example

4.3.1. Networked control system setup

To validate the RANC applicability, a simple networked control system was set up based on the configuration in **Figure 9**. Here, the main controller was built in an experimental PC equipped with the MCU (presented in Section 4.2.2) The network module includes one coordinator and one router employed the ZigBee protocol [19]. The multifunction card was used to perform the communications with the TDPD module and the network. The control objective is speed tracking control of a servomotor system via the setup network. Here, the servomotor system was *RE*-max series manufactured by Maxon Corporation. The transfer function from the input to output of this system can be expressed as [20]

$$G_p(s) = \frac{32300}{0.001s^2 + 20.93s + 493.4} \tag{42}$$

To evaluate the capability of the proposed RANC, a comparative study of the RANC with three existing approaches, a QFT-based robust controller (QFTRC), a static state feedback controller (SSFC), and a hybrid QFT–SSFC as shown in **Table 3**, has been investigated. To make control challenges, disturbance loads and computation delays were randomly generated. Here, the loads were created by putting the system into a varied magnetic field which affected on the motor shaft. Meanwhile, an arbitrary matrix-based complex calculating procedure representing a complex application was added to the controller at the PC side.

Control	**Main control modules**					**Functions**	
method	**QFT**	**SFC**		**SSW-based**	**VSPA**	**Delay**	**Packet loss compensation**
		Fixed gain	**Dynamic gain**	**TDPD–TDP**	**based TDP**	**compensation**	
QFTRC	√					√	
SSFC		√				√	√
QFT–SSFC	√	√		√		√	√
RANC	√		√	√	√	√	√

Table 3. Specifications of the compared controllers.

Based on Section 4.2.4, QFT delay bounds were defined as
$\bar{\tau}^{com}_{QFT} = 0.02s$, $\bar{\tau}^{ca}_{QFT} = 0.03$, $\bar{\tau}^{sc}_{QFT} = 0.03$and, the QFT controller was designed as

$$G_{QFT}(s)=\frac{30.5387s^2+56684.6s+3328880}{10^{-8}s^3+0.002852s^2+180.2s+1260000}$$
$$F_{QFT}(s)=426.65/(s+400) \tag{43}$$

Based on Section 4.2.5 and by setting the initial bounds for the total delay and packet dropouts as: $\bar{\tau}_0=0.1s>\bar{\tau}_{QFT}$, $\bar{p}_0=2$; the initial RSFC gain was computed as $K^0_{RSFC}=[0.3796 \quad 0.2642]$.

The initial sampling period of the RANC was selected as 10 ms, while that of the other controllers must be fixed to 0.15 s to cover all the delays.

4.3.2. Real-time experiments

In this section, real-time speed tracking of a servomotor system over the setup network has been investigated in which the reference speed was selected as 10 rad/s. The experiments using the comparative controllers were carried out and, consequently, the results were displaced in **Figure 10**. An analysis on the control performances using four evaluation criteria, *PO* (percent overshoot), *ST* (settling time), *SSE* (steady state error) and *MSE* (mean square error), is then performed as demonstrated in **Table 4**.

Controller	Step responses			
	PO [%]	ST [s]	SSE [%]	MSE [rad/s]^2
QFTRC	2.6	1.0	13.9	1.23
SSFC	8.67	2.1	6.18	1.87
QFT–SSFC	2.07	1.35	5.88	1.35
RANC	0.2	0.8	1.58	0.63

Table 4. Comparison of NCS performances using different controllers.

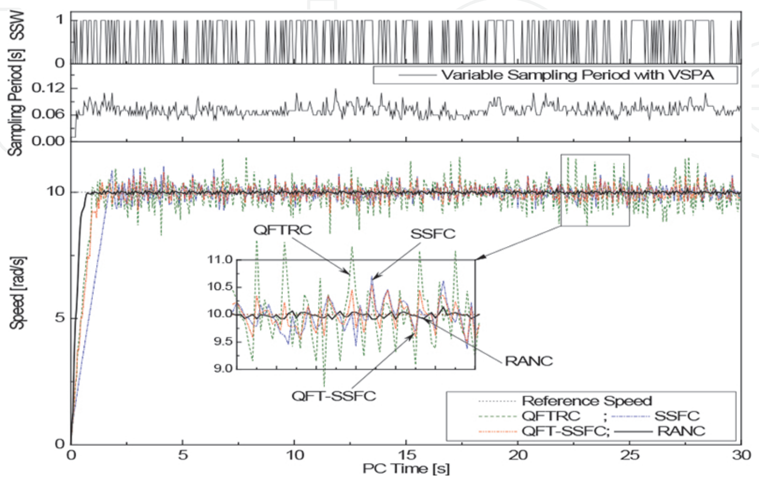

Figure 10. Step speed performances using different controllers.

From **Figure 10** and **Table 4**, it is clear that the worst-case performance was with the QFTRC. It is because the QFTRC was designed for an NCS in which the delays are bounded by constraint (9) and there is no packet dropout. Hence when facing with the networked servo-motor including both the large delays and packet dropouts, the controller could not guarantee the robust tracking (*SSE* was 13.9%). In case of using the SSFC for which the control gain was designed for the worst network conditions (large delays and highest number of continuous packet dropouts), *SSE* was remarkably reduced to 6.18%. However due to this fixed control gain use, the system response was much slower than that of the QFTRC. Additionally, due to lack of disturbance rejection capability, the SSFC could not ensure a smooth tracking (PO = 8.67% and ST = 2.1s). The QFT-–SSFC, by taking advantages of both the QFT and SSFC, could improve the tracking result. Nonetheless, the QFT–SSFC with fixed control gains and fixed sampling period restricted the system adaptability to the sharp variations of network problems and disturbances (*SSE* was slightly reduced to 5.88%).

The best tracking result was achieved by using the RANC (see **Figure 10** and **Table 4**). It comes as no surprise because the RANC possesses all the advantages of the QFT and state feedback designs and, the high adaptability to the system changes by using the adaptive control gains and adaptive sampling period. **Figure 11** demonstrates the delay and packet dropout detection and prediction results of the TDPD and TDP modules, respectively. During the operation, these were supported by the VSPA and hybrid QFT–RSFC modules to regulate properly the sampling period and control gains.

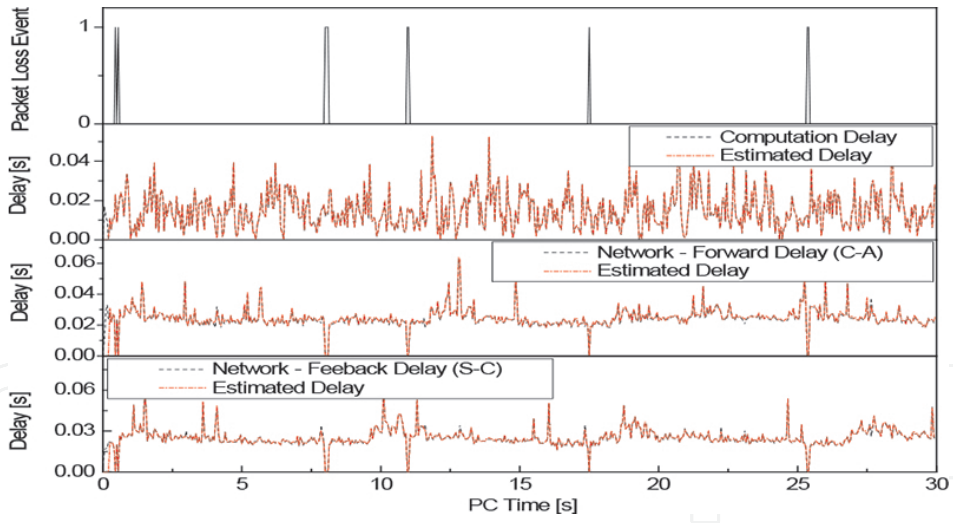

Figure 11. Step speed case: performances of TDPD and TDP.

5. Conclusions

In summary, an advanced bilateral teleoperation NCS has been introduced. In addition, the two important issues in designing the BTNCS system, FSFC and RANC, have been clearly addressed. The LVQNN-based FSFC shows the safe and cost-effective solution in controlling the FRM while the RANC-based LM controller is a feasible choice to drive the slave manipulator over the network with delay and packet loss problems.

The effectiveness of the LVQNN classifier as well as the RANC controller has been confirmed through the two case studies and analysis. One of our future research topics would be the full design of the proposed BTNCS to investigate the applicability of this method in a practical application such as remote construction equipment.

Author details

Truong Quang Dinh[1*], Jong Il Yoon[2], Cheolkeun Ha[3] and James Marco[1]

*Address all correspondence to: q.dinh@warwick.ac.uk

1 WMG, University of Warwick, United Kingdom

2 Korea Construction Equipment Technology Institute, Korea

3 School of Mechanical Engineering, University of Ulsan, Korea

References

[1] Park W.I, Kwon S.C, Lee H.D, Kim, J (2012) Real-time thumb-tip force predictions from noninvasive biosignals and biomechanical models. Int. J. Prec. Eng. Man. 13(9): 1679–1688.

[2] Jung K.M, Chu B.S, Park S.S, Hong D.H (2013) An implementation of a teleoperation system for robotic beam assembly in construction. Int. J. Prec. Eng. Man. 14(3): 351–358.

[3] Shafiqul I, Peter X.L, Abdulmotaleb E.S (2014) New stability and tracking criteria for a class of bilateral teleoperation systems. Inf. Sci. 278: 868–882.

[4] Kuchenbecker J.K, Niemeyer G (2006) Induced master motion in force-reflecting teleoperation. ASME J. Dyn. Sys. Meas. Control 128(4): pp. 800–810.

[5] Polushin I.G, Liu P.X, Lung C.H (2012) Stability of bilateral teleoperators with generalized projection-based force reflection algorithms. Automatica 48: 1005-1016.

[6] Sun D, Naghdy F, Du H (2014) Application of wave-variable control to bilateral teleoperation systems: A survey. Annu. Rev. Control 38: 12–31

[7] Zhang W, Branicky M.S, Phillips S.M (2001) Stability of networked control systems. IEEE Control Syst. Mag. 2: 84–99.

[8] Hua C.C, Liu X.P (2013) A new coordinated slave torque feedback control algorithm for network-based teleoperation systems. IEEE/ASME Trans. Mechatronics 18(2): 764–774.

[9] Yang F, Wang Z, Hung Y.S, Mahbub G (2006) H∞ control for networked control systems with random communication delays. IEEE Trans. Autom. Control 51(3): 511–518.

[10] Yi J.Q, Wang Q. D, Zhao B, Wen J.T (2007) BP neural network prediction-based variable-period sampling approach for networked control systems. Appl. Math. Comput. 185: 976–988.

[11] Chien L.L, Pau L.H (2010) Design the remote control system with the time-delay estimator and the adaptive Smith predictor. IEEE Trans. Ind. Informat. 6(1): 73–80.

[12] Daniel L.E, Mario E.M (2012) Variable sampling approach to mitigate instability in networked control systems with delays. IEEE Trans. Neural Netw. Learn. Syst. 23(1): 119–126.

[13] Li H, Sun Z, Chow M.-Y, Sun F (2011) Gain-scheduling-based state feedback integral control for networked control systems. IEEE Trans. Ind. Electron. 58(6): 2465–2472.

[14] Shi Y, Huang J, Yu B (2013) Robust tracking control of networked control systems: Application to a networked DC motor. IEEE Trans. Ind. Electron. 60(12): 5864–5874.

[15] Gao H, Chen T, Lam J (2008) A new delay system approach to network-based control. Automatica 44(1): 39–52.

[16] Gao H, Chen T (2008) Network-based H∞ output tracking control. IEEE Trans. Autom. Control 53(3): 655–667.

[17] Yang R, Liu G.-P, Shi P, Thomas C, Basin M.V (2014) Predictive output feedback control for networked control systems. IEEE Trans. Ind. Electron. 61(1): 512–520.

[18] Pang Z.-H, Liu G.-P, Zhou D, Chen M (2014) Output tracking control for networked systems: A model-based prediction approach. IEEE Trans. Ind. Electron. 61(9): 4867–4877.

[19] Truong D.Q, Ahn K.K, Trung N.T (2013) Design of an advanced time delay measurement and a smart adaptive unequal interval grey predictor for real-time nonlinear control systems. IEEE Trans. Ind. Electron. 60(10): 4574–4589.

[20] Truong D.Q, Ahn K.K (2015) Robust variable sampling period control for networked control systems. IEEE Trans. Ind. Electron. 62(9): 5630–5643.

[21] Truong D.Q, Lee S, Liem D.T, Ahn K.K (2015) Robust tracking control of networked control systems included uncertainties. Proceedings of 2015 IEEE International Conference on Advanced Intelligent Mechatronics (AIM): 1049–1054.

[22] Schneider P, Biehl M, Hammer B (2009) Adaptive relevance matrices in learning vector quantization. Neural Comput. 21(12): 3532–3561.

[23] Deng J.L (1982) Control problem of grey system. Syst. Control Lett. 1(5): 288-294.

[24] Truong D.Q, Ahn K.K (2009) Force control for hydraulic load simulator using self-tuning grey predictor—fuzzy PID. Mechatronics 19(2): 233-246.

[25] Truong D.Q, Ahn K.K (2009) Self tuning of quantitative feedback theory for force control of an electro-hydraulic test machine. Control Eng. Pract. 17(11): 1291–1306.

[26] Ahn K.K, Truong D.Q, Nam D.N.C, Yoon J.I, Yokota S (2010) Position control of polymer metal composite actuator using quantitative feedback theory. Sens. Actuat. A Phys. 159(2): 204–212.