IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON KEY TECHNOLOGIES FOR SMART FACTORY OF INDUSTRY 4.0

Received May 10, 2017, accepted May 27, 2017, date of publication June 15, 2017, date of current version July 17, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2715878

Frequency-Tracking Clock Servo for Time Synchronization in Networked Motion Control Systems

XIN CHEN[1], DI LI[1], SHIYONG WANG[1], HAO TANG[1], AND CHENGLIANG LIU[2]
[1]School of Mechanical and Automotive Engineering, South China University of Technology, Guangzhou 510640, China
[2]School of Mechanical Engineering, Shanghai Jiao Tong University, Shanghai 200140, China

Corresponding author: Shiyong Wang (mesywang@scut.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 51605168, the Natural Science Foundation of Guangdong Province under Grant 2015A030308002, the Science and Technology Planning Project of Guangdong Province under Grant 2011B010300002 and Grant 2015B010917001, and the Fundamental Research Funds for the Central Universities under Grant 2017MS016.

ABSTRACT With the application of internet in manufacturing, motion control system is tend to networking. Clock synchronization is a basic requirement to guarantee capability of coordination in networked motion control systems. In this paper, we propose a frequency-tracking clock servo (FTCS) to adjust the local clock to synchronize with the reference clock. The frequency of FTCS can rapidly lock onto the reference frequency and the offset can be fully compensated within a single synchronizing cycle. Simulations and experiments are performed to validate the feasibility and superiority of FTCS. Compared with the proportional-integral clock servo, FTCS can achieve the rapid synchronized speed and better precision with low frequency of sending synchronization messages.

INDEX TERMS Clock synchronization, EtherCAT master, networked motion control.

I. INTRODUCTION

Industry 4.0 is believed to be approaching [1], [2]. It integrates internet of things [3], [4], big data [5], [6] and cloud computing [7], [8] into manufacturing. This brings many challenges to the manufacturing system. As the core of manufacturing, motion control system employs real-time Ethernet to improve its openness, reconfiguration and compatibility to meet the requirements of smart factory. As for networked motion control system (NMCS), clock synchronization makes a key factor affecting its performance as well as the foundation to guarantee its normal operation and multi-axis coordinated motion [9]–[11]. Firstly if the clock is not fully synchronized with the reference, NMCS will suffer packet loss and bad trajectory. Secondly, the clock synchronization precision is the critical factor in trajectory error. Thirdly, because timestamps have the same timeliness as motion control instructions, they compete for real-time resource and bandwidth in resource-constrained system. Consequently, clock synchronization mechanisms with fast synchronized speed, high precision, less timestamps and low computation complexity are preferable in NMCS.

Although there are many protocols for clock synchronization (e.g. IEEE1588, NTP and GPS), the basic principle is the same. It is that the networked nodes need to employ a clock servo to ensure that every local clock keeps in pace with the reference as accurate as possible [12]–[16]. In NMCS, the time of the local clock should be continuous for avoiding packet loss. Frequency adjustment is an effective method to ensure the continuity and stability of clocks. Phase-locked loop (PLL) can achieve perfect synchronization performance, but it needs a specific oscillator [14]. The EtherCAT protocol uses an ordinary oscillator in the slave node to synchronize with the reference according to distributed clock (DC) synchronization mechanism [16], [17]. Nevertheless, master-slave synchronization is not considered in the DC mechanism. A proportional-integral clock servo (PICS) is widely used for synchronization by tracking the offset to correct the local clock frequency [18]–[20]. In [21], a PID controller is employed to achieve better performances on convergence time and precision. However, whether using PI or PID clock servo, the performances on convergence time and precision are highly dependent on their parameters, e.g. k_p, k_i, and synchronizing cycle [22], [23]. PICS with the optimal parameters proposed in [18] and [19] can obtain faster convergence rate but poorer precision. Although there are a number of different filters, such as an averaging method, linear

programming, and Kalman filters [24], [25] for better synchronization accuracy, the filtering algorithm is computation-intensive and the convergence time can't be guaranteed. Balasubramanian *et al.* [26] proposes a frequency compensated method of which the convergence time can be minimized to only one synchronizing cycle, but this method may lead to slight deterioration on precision.

In this paper, we propose a frequency-tracking clock servo (FTCS) which equipped with a P-controller and a frequency-tracking module. It has the faster synchronized speed and better precision with less timestamps. The contributions of this paper can be summarized as follows:

- This paper proposed a frequency-tracking clock servo (FTCS) which can compensate for offset and skew in one synchronizing cycle with better precision and less exchanging of synchronizing messages.
- The better precision and fast synchronized speed can be achieved by regulating the parameter of the P controller dynamically.
- Compared with PICS which uses the optimal parameters and FTCS, FTCS can obtain faster synchronized speed and better precision with less timestamps.

This paper is organized as follows. Section II introduces the basic principle of clock synchronization, the local clock models and the affine model for clocks. Section III illustrates the frequency-tracking clock servo. In section IV, the performances of FTCS and PICS are analyzed and compared. In Section V, FTCS and PICS are respectively employed by the EtherCAT master and their performances are compared.

II. CLOCK SYNCHRONIZATION BASIS

In this section we discuss the basic principle of clock synchronization, the clock model and the affine model for clocks.

A. BASIC PRINCIPLE OF CLOCK SYNCHRONIZATION

Possible misalignments between the local clock and the reference clock occur due to two primary reasons. Firstly, the clocks in the network may not start at exactly the same time, and this results in an initial offset between different clocks. The second reason is the presence of small and unavoidable skew between different clocks, which causes their clocks to diverge over time. Although there are many types of clock synchronization protocols, such as NTP, GPS, PTP or its variants, the basic principle is largely identical with only minor differences between methods. They synchronize with each other by exchanging synchronizing messages (timestamps). However, transport delay and execution delay are introduced in this process.

According to PTP, the offset between the reference clock and the local clock is (as shown in Fig. 1)

$$T_{offset}(n) = \frac{\tau_r^1(n) - \tau_l^2(n) + \tau_l^3(n) - \tau_r^4(n)}{2} + \frac{d_{rl}^n - d_{lr}^n}{2} \tag{1}$$

where d_{rl}^n is the delay from the reference clock to the local clock and d_{lr}^n is the delay from the local clock to the reference clock. From the timestamps, d_{rl}^n and d_{lr}^n can't be calculated

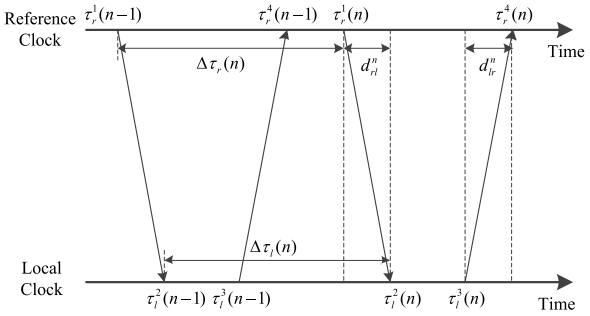


FIGURE 1. Synchronizing messages exchange.

directly [27]–[29]. Usually, assume the forward and backward transport delay is symmetric:

$$d_{rl}^n = d_{lr}^n = \frac{\tau_r^4(n) - \tau_r^1(n) + \tau_l^3(n) - \tau_l^2(n)}{2} \tag{2}$$

Every clock is characterized by a clock skew (i.e. the clock frequency deviation), an offset (i.e. the time differences from the reference clock) and a delay. The relationship between $\tau_r(n)$ and $\tau_l(n)$ is

$$\tau_r(n) + d_{rl} = (1+\alpha)\tau_l(n) + \beta \tag{3}$$

where α is the clock skew and β is the initial offset between the reference and the local clock. The initial offset can be compensated by (1). If $\alpha \neq 0$, the offsets are constantly changing in different times. Therefore, clock skew is the key factor in non-synchronizing between two clocks.

B. LOCAL CLOCK MODELS

The clock usually is made up of a crystal oscillator and a counter. If the clock is equipped with a special oscillator whose frequency can be controlled by voltage or temperature, PLL is employed to adjust its frequency. This method can achieve perfect synchronizing performance. If the clock is equipped with an inexpensive crystal oscillator whose frequency can't be controlled, the frequency compensation clock model can be used. This clock model adjusts its frequency by modifying the addend value. As shown in Fig. 2, $\tau_l\,(k)$ is the local clock time and $u(k)$ is the frequency compensation value. The local model can be described by a z transfer function:

$$G(z) = \frac{\tau_l\,(z)}{u(z)} = \frac{k_c T}{z-1} \tag{4}$$

where T is the synchronizing cycle and k_c is the clock constant, which can be calculated by $k_c = f/u(0)$. It is simple to realize this clock model on FPGA or other embedded microprocessors without the special oscillator. Because of its low cost, this clock model is widely used. This clock model usually employs PICS to adjust the addend value. Meanwhile, frequency-tracing clock servo also can be used in this clock model.

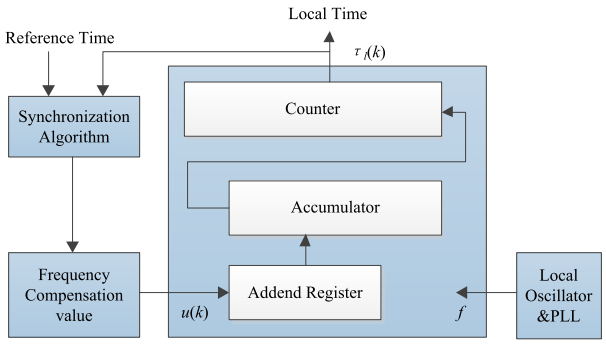


FIGURE 2. Clock model with addend register.

However, in order to ensure the stability, the clock frequency cannot be adjusted directly by users in some systems, such as RTX (the real-time kernel of Windows). The frequency of local clock and their addend register cannot be modified directly in these systems. However, the triggering time of the event can be changed. The PICS or frequency-tracing clock servo also can be used to adjust the triggering time.

C. AFFINE MODEL FOR CLOCKS

Assume that the display of clock i at time t, denoted by $\tau_i(t)$, satisfies

$$\tau_i(t) = \int_0^t a_i(\tau)d\tau + b_i \tag{5}$$

where $a_i(\tau)$ is the frequency of the clock i, b_i is the offset of clock i in the beginning. In the practical application, $a_i(\tau)$ is not invariable because the frequency of the clock oscillator is influenced by the temperature and its age. But it can be seen as a constant in a certain time. When the clock model is equipped with the addend register, the clock time is discrete. Assume that the frequency of clock $i(a_i(\tau))$ is constant in one synchronizing cycle time. Then the display of clock i at the n-th synchronizing cycle T satisfies

$$\tau_i(n) = \sum_{k=0}^{n} a_i(k)T + b_i \tag{6}$$

where $a_i(k)$ represents the speed of the local clock. Assume that the display of clock i in the interval between the n^{th} and $n+1^{\text{th}}$ synchronizing message, denoted by $\tau_i^n(t)$, satisfies

$$\tau_i^n(t) = a_i^n t + b_i^n \tag{7}$$

Where $a_i^n = a_i(n)$ and $b_i^n = \tau_i(n-1) - (n-1)a_i(n)T$. The affine model for clocks [30] can be introduced to describe their relationship intuitively. The time of clock j is translated to the time of clock i and satisfies

$$T_i^j(\tau_j^n(t)) = \frac{a_i^n}{a_j^n}\tau_j^n(t) + b_i^n - \frac{a_i^n}{a_j^n}b_j^n \tag{8}$$

III. FREQUENCY-TRACKING CLOCK SERVO

A. FREQUENCY- TRACKING CLOCK SERVO

The clock frequency deviation is the primary factor for non-synchronizing. If the frequency of the reference is acquired accurately and the local clock set the frequency equal to the reference, the skew is fully compensated immediately. According to clock synchronization protocols, synchronizing messages carrying the current time of the reference clock are broadcast periodically. Though the frequency of the reference and the local clock cannot be captured directly, their proportional relationship can be captured. The time when the reference clock sends its n^{th} synchronizing message is denoted by $\tau_r^{(n)}$ as shown in Fig. 1. The time when the local clock receives the n^{th} synchronization time is denoted by $\tau_l^{(n)}.\tau_r^{(n)}$ and $\tau_l^{(n)}$ is the current display time of the reference and the local time respectively. According to the affine model, the time $\tau_l^{(n)}$ translated to the reference is

$$T_r^l(\tau_l^{(n)}) = \frac{a_r^n}{a_l^n}\tau_l^{(n)} + b_r^n - \frac{a_r^n}{a_l^n}b_l^n \tag{9}$$

Because there is transport and execution delay (as shown in Fig. 1), $T_r^l(\tau_l^{(n)})$ and $\tau_r^{(n)}$ stratifies

$$\tau_r^{(n)} + d_{rl}^n = T_r^l(\tau_l^{(n)}) \tag{10}$$

where d_{rl}^n can be calculated by (2). $\tau_r^{(n)} + d_{rl}^n$ can be denoted by $\tau_{rd}^{(n)}$ Then the proportional relationship of their frequency, denoted by $k(n)$, satisfies

$$\begin{bmatrix} \tau_{rd}^{(n)} \\ \tau_{rd}^{(n-1)} \end{bmatrix} = \begin{bmatrix} \tau_l^{(n)} & 1 \\ \tau_l^{(n-1)} & 1 \end{bmatrix} \begin{bmatrix} k(n) \\ B(n) \end{bmatrix} \tag{11}$$

Where $k(n) = a_r^n/a_l^n$ and $B(n) = b_r^n - k(n)b_j^n$. The vector on the left hand side of (11) above can be denoted by $\boldsymbol{y}$, the matrix on the right hand side of (11) denoted by $\boldsymbol{A}$, and the vector on the right hand side of (11) denoted by $\boldsymbol{x}$, i.e. $\boldsymbol{y} = \boldsymbol{Ax}$. Since $\tau_l^{(n)} > \tau_l^{(n-1)}$, $|\boldsymbol{A}| \neq 0$ and the matrix $\boldsymbol{A}$ has full-rank. The proportional relationship of the frequency between the reference and local clock can be calculated:

$$k(n) = \frac{\tau_{rd}^{(n)} - \tau_{rd}^{(n-1)}}{\tau_l^{(n)} - \tau_l^{(n-1)}} \tag{12}$$

The local clock adjusts its frequency to track the reference frequency according to the proportional relationship $k(n)$. In the next synchronizing cycle, the frequency of the local clock is

$$a_l^{n+1} = k(n)a_l^n \tag{13}$$

There is not only the skew but also the offset in synchronizing process. As shown in Fig. 3, $\tau_r(t)$ represents the time of the reference clock. $\tau_l(t)$ represents the local clock with fully compensation for the clock skew and offset $\tau_l'(t)$ represents the local clock with compensation for the clock skew. $\tau_l''(t)$ represents the free-running local clock. Assume that the ideal convergence time is $T_c(0 < T_c < T_{sync}, T_{sync}$ based on the reference time). In other words, if $t_2 \geq t \geq t_c$,

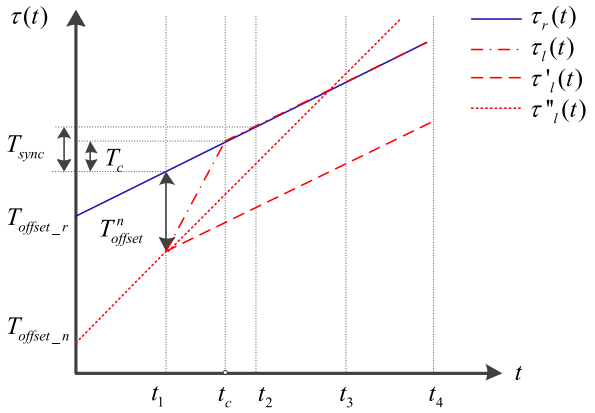


FIGURE 3. Control model of FTCS.

$\tau_l(t) = \tau_r(t)$. The proportional relationship of the frequency before or after T_c, denoted by k_1^{n+1} and k_2^{n+1}, satisfies

$$\begin{bmatrix} \tau_{rd}^{(n)} + T_c \\ \tau_{rd}^{(n)} \\ \tau_{rd}^{(n)} + T_c \\ \tau_{rd}^{(n)} + T_{sync} \end{bmatrix} = \begin{bmatrix} \tau_l(t_c) & 1 & 0 & 0 \\ \tau_l(t_1) & 1 & 0 & 0 \\ 0 & 0 & \tau_l(t_c) & 1 \\ 0 & 0 & \tau_l(t_2) & 1 \end{bmatrix} \begin{bmatrix} k_1^{n+1} \\ B(n) \\ k_2^{n+1} \\ B'(n) \end{bmatrix} \tag{14}$$

where

$$\tau_l(t_c) = T_l^r\left(\tau_{rd}^{(n)} + T_c\right) = \tau_{rd}^{(n)} + T_c \tag{15}$$

$$\tau_l(t_2) = T_l^r\left(\tau_{rd}^{(n)} + T_{sync}\right) = \tau_{rd}^{(n)} + T_{sync} \tag{16}$$

$\tau_l(t_c)$ and $\tau_l(t_2)$ are the time of the local clock when the reference clock are at $\tau_{rd}^{(n)} + T_c$ and $\tau_{rd}^{(n)} + T_{sync}$ respectively. Since $\tau_l(t) = \tau_r(t)$ $(t_c \le t \le t_2)$, $\tau_l(t_c) = \tau_{rd}^{(n)} + T_c$ and $\tau_l(t_2) = \tau_{rd}^{(n)} + T_{sync}$. From (14), $k_1^{n+1} = (T_{offset}^n + T_c)/T_c$ and $k_2^{n+1} = 1$. From (1), the perceived offset is denoted by $T_{offset}^n.a_l^{n+1}(t)(t_1 < t < t_2)$ can be expressed as

$$a_l^{n+1}(t) = \begin{cases} (1 + \frac{T_{offset}^n}{T_c})k(n)a_l^n & (t_1 < t < t_c) \\ k(n)a_l^n & (t_c < t < t_2) \end{cases} \tag{17}$$

Set $T_c = T_{sync}$, then a_l^{n+1} satisfy

$$a_l^{n+1} = (1 + T_{offset}^n/T_{sync})k(n)a_l^n \tag{18}$$

B. ANALYSIS AND IMPROVEMENT

As known to all, the performances are determined by their parameters. The faster synchronized speed and better noise suppression can't be got together. From (18), it likes a P controller with a frequency-tracing module. When the value of the proportional controller is $1/T_{sync}$, the offset can be fully compensated in one synchronizing cycle. The frequency-tracing module can synchronize the local clock speed with the reference quickly. However, its noise suppression isn't better. Add a weight p to improve the precision. (18) is rewritten as

$$a_l^{n+1} = k(n)a_l^n + \frac{pT_{offset}^n}{T_{sync}}k(n)a_l^n \ (0<p \le 1) \tag{19}$$

How the noise influences the performances is analyzed as follows.

The accuracy of the timestamps is the critical factor for synchronization performance. However, there are deviations on the timestamps. The deviations are caused by the drift and jitter of the clock, uncertainties on the transport, execution delay, and truncation errors. The clock drift and the truncation errors lead to frequency deviation but can be compensated in the next synchronizing cycle, because the deviations caused by them are quantifiable and FTCS or PICS is designed for tracing the deviations. Nevertheless, the clock jitter and uncertainties on the transport and execution delay are random and uncertain. Furthermore, the transport and execution delay is the major contributor to the unfaithful timestamps. Since the transport and execution delay far outweigh the clock jitter, we assume there merely exist the uncertainties on the transport and execution delay. The inaccuracy or noise is denoted by $N(n)$ and according to (10), (11) and (19):

$$k(n) = \frac{d\tau_r^{(n)} + V(n)}{d\tau_l^{(n)}} \tag{20}$$

$$\frac{T_{offset}^n}{T_{sync}} = \frac{\tau_{rd}^{(n)} - \tau_l^{(n)}}{T_{sync}} + \frac{N(n)}{T_{sync}} \tag{21}$$

where $d\tau_r^{(n)} = \tau_{rd}^{(n)} - \tau_{rd}^{(n-1)}$, $d\tau_l^{(n)} = \tau_l^{(n)} - \tau_l^{(n-1)}$, $V(n) = N(n) - N(n-1)$. Then, $k(n+1)$ satisfies

$$k(n+1) = \frac{d\tau_r^{(n)}}{d\tau_l^{(n)}} + p\frac{\tau_{rd}^{(n)} - \tau_l^{(n)}}{T_{sync}}\frac{d\tau_r^{(n)}}{d\tau_l^{(n)}} + v_k(n) \tag{22}$$

$$v_k(n) = p\frac{N(n)}{T_{sync}}\frac{d\tau_r^{(n)}}{d\tau_l^{(n)}} + \left(1 + p\frac{\tau_{rd}^{(n)} - \tau_l^{(n)}}{T_{sync}}\right)\frac{V(n)}{d\tau_l^{(n)}} + p\frac{N(n)}{T_{sync}}\frac{V(n)}{d\tau_l^{(n)}} \tag{23}$$

where $v_k(n)$ represents the deviation caused by noise. In the steady state $d\tau_l^{(n)} \approx d\tau_r^{(n)} = T_{sync}$ and $T_{offset}^n \ll T_{sync}$. From (22), when $p = 1$, the convergence time is the shortest (i.e. one synchronizing cycle). When $p = 0$, the offset can't be compensated. When p is less than 1, the local clock can't be locked in one synchronizing cycle. The convergence time is increasing with p decreasing. From (23), the more accurate frequency of the local clock can be achieved with the synchronizing cycle increasing. The deviations T_{off_noise} caused by the noise can be represented by $v_k(n)T_{sync}$ which satisfies

$$T_{off_noise} = V(n) + p\left(N(n) + \frac{N(n)V(n)}{T_{sync}} + \frac{\tau_{rd}^{(n)} - \tau_l^{(n)}}{T_{sync}}V(n)\right) \tag{24}$$

From (24), the error is related to the noise and p. Obviously, the deviation becomes smaller when p is lesser. Because $N(n)v(n) \ll T_{sync}$ and $p \le 1$, T_{off_noise} is lessening with synchronizing cycle increasing when the noise power maintains a constant. However, the synchronizing cycle is not infinite

TABLE 1. Algorithm for Modified FTCS.

Algorithm Modified FTCS
Set p , B , F ;
Calculate $k(n)$ according to (14);
Calculate T^n_{offset} ;
If $\left\|T^n_{offset}\right\| < B$
$\quad a_i(n+1) = k(n)a_i(n) + k(n)a_i(n)pT^n_{offset}$;
Else
$\quad a_i(n+1) = k(n)a_i(n) + k(n)a_i(n)T^n_{offset}$;
End
If $a_i(n+1) - 1 < \left\|F\right\|$
$\quad a_{out} = a_i(n+1)$;
Else
$\quad$ If $a_i(n+1) - 1 < 0$
$\quad\quad a_{out} = 1 - F$;
$\quad$ Else
$\quad\quad a_{out} = 1 + F$;
$\quad$ End
End
$a_i(n) = a_{out}$;

since the clock drift and the truncation errors result in the frequency deviation.

From the above, when $p = 1$, the local clock can be locked in one synchronizing cycle but has the weakest ability for noise suppression. When $p = 0$, the local clock can be fully compensated the skew and has the strongest ability for noise suppression, but the predicted offset T^n_{offset} can't be compensated. When $0 < p < 1$, the local clock can achieve better precision than $p = 1$, but the convergence time isn't the one synchronizing cycle. We set the boundary conditions to combine their advantages. The modified algorithm is shown in Table I. When the absolute value of T^n_{offset} is smaller than B after some time, the local clock can be considered as convergence and the clock speed in the next synchronizing cycle is calculated by (19). On the contrary, the clock speed is calculated by (18) to make the local clock convergence quickly. B is the ideal precision and can be used to monitor the synchronization performance. p is set to (0,1) and also can be adjusting by self-adoption. F is the limitation of the local clock speed. The clock speed is neither infinite nor zero or negative both in PICS or FTCS. The frequency needs have upper and lower limit to ensure continuity and normal operation. The frequency needs a reasonable bound to maintain continuity and normal operation.

IV. SIMULATION AND ANALYSIS

In order to illustrate the feasibility and superiority of FTCS, simulation models of FTCS and PICS are created and their synchronizing performances are compared from the convergence time and the steady state error (SSE) in different conditions in this section. When the perceived offset between the reference and the local clock is less than 500ns at some point and the error is never bigger than 500ns after this point, we consider that the local clock is synchronized with the reference. We use the SSE to describe the accuracy of the synchronization. The noise of the transport and execution delay is introduced for simulating real conditions.

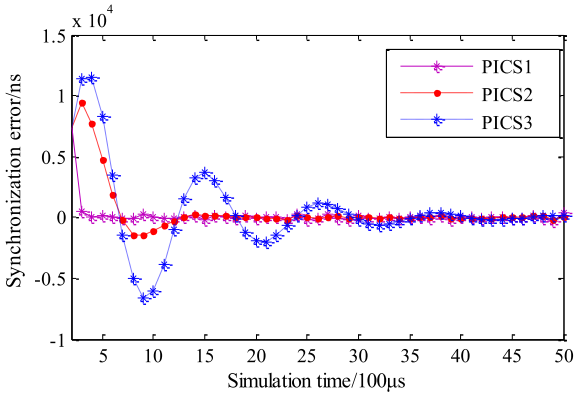


FIGURE 4. Performance of PICS in different parameters.

TABLE 2. Performances different parameters.

	Convergence time/cycle	Parameters/10^{-5}	SSE/ns
FTCS1	1	$p = 1$	124.9173
FTCS2	1	$p = 1/2$	98.5732
FTCS3	1	$p = 1/64$	77.6052
FTCS4	1	$p = 1/128$	78.1814
FTCS5	1	$p = 0$	80.1098
PICS1	2	$k_p = 1, \quad k_i = 1$	125.2856
PICS2	9	$k_p = 0.5, \quad k_i = 0.3$	74.3365
PICS3	40	$k_p = 0.2, \quad k_i = 0.3$	70.5104

A. PERFORMANCE IN DIFFERENT PARAMETERS

It is generally known that the performances of PI clock servo (PICS) are determined by its parameters. From [18], [19], the optimal parameters satisfy $k_p k_c T = 1$, $k_i k_c T = 1$. In this simulation, the performances of FTCS and PICS with different parameters are discussed in the same simulation conditions. The frequency of the local clock is set to $f = 9.26 \times 10^8$Hz, and the synchronizing cycle is set to $T = 100\mu$s. The parameters of FTCS and PICS are set as shown in Table II and the PICS1 uses the optimal parameters.

The performances of FTCS and PICS with different parameters are shown in Fig. 4 and Table II. All of FTCS can be converged in one synchronizing cycle while PICS with the optimal parameters in two synchronizing cycle. FTCS1 has the similar SSE with PICS1. When $p = 10^{-5} = 1/T_{sync}$, the SSE is the biggest. When $p < 1/T_{sync}$, the SSE is lower than FTCS1. FTCS3 has the lowest SSE in FTCSs, but the SSE of PICS2 and PICS3 is better than FTCS3. PICS3 has the lowest convergence time but the best SSE.

TABLE 3. Value of k_p and k_i in different synchronizing cycles.

PICS	Sync. Cycle/μs	k_p	k_i
1	50	2×10^{-5}	2×10^{-5}
2	100	1×10^{-5}	1×10^{-5}
3	500	2×10^{-6}	2×10^{-6}
4	1000	1×10^{-6}	1×10^{-6}

TABLE 4. Performance in different synchronizing cycles.

	Sync. Cycle/μs	Convergence time/cycle	SSE/ns
PICS	100	2	125.2856
	500	3	153.7698
	1000	4	122.1759
FTCS1 $p=1/T_{sync}$	100	1	124.9173
	500	1	124.6888
	1000	1	124.0961
FTCS2 $p=1/64T_{sync}$	100	1	77.6052
	500	1	77.4143
	1000	1	77.4132

From the results, the parameters of PICS and FTCS are the critical factor in performance. In PICS, the convergence time and SSE is determined by its parameters. In FTCS, the convergence time is the shortest and the better SSE can be achieved by decreasing the parameter p. PICS can achieve better SSE but longer convergence time even with the optimal parameters. Compared with PICS, the FTCS can achieve the fast synchronized speed with better SSE.

B. PERFORMANCES IN DIFFERENT SYNCHRONIZING CYCLES

In this simulation, the performances of PICS and FTCS are discussed in different synchronizing cycles. The frequency of the local clock is set to $f = 9.26 \times 10^8$Hz and the synchronizing cycles T_{sync} are respectively set to 100μs, 500μs and 1ms. The values of k_p and k_i are the optimal PI parameters in different synchronizing cycles as shown in Table III. The parameter of FTCS is $p = 1/T_{sync}$ and $B = \inf$. The performances of PICS and FTCS are shown in Fig. 5 and Table IV. The FTCS can be converged in one period and their SSEs are basically similar in different synchronizing cycles. The convergence time of PICS is extended with the synchronizing cycle increasing while their SSE doesn't follow this rule. The SSE of the PICS in 500μs is the worst. The convergence times of FTCS1 and FTCS2 all are one synchronizing cycle. The SSEs of FTCS1 and FTCS2 are stable in different synchronizing cycles.

From the results, the SSEs of FTCS at 100μs, 500μs and 1ms are close to PICS at 100μs. This means that FTCS at 1ms can achieve the same SSE as PICS using optimal

parameters at 100μs. In other words, FTCS can still achieve the same SSE as PICS when sending less synchronizing messages. Therefore, compared with PICS, FTCS can use less timestamps to achieve a better SSE with fast synchronized speed. Since timestamps have the high requirement

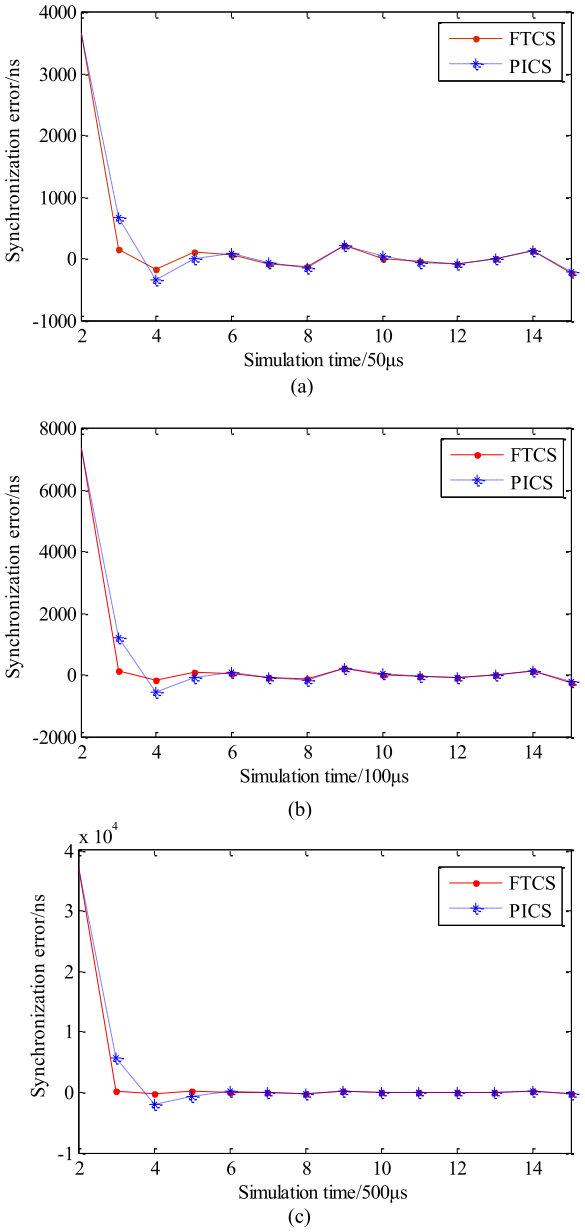


FIGURE 5. Performance of PICS and FTCS in different synchronizing cycles. (a) Synchronizing cycle is 50μs. (b) Synchronizing cycle is 100μs. (c) Synchronizing cycle is 500μs.

on real-time, FTCS can use less real-time resource and bandwidth to get a better performance.

C. PERFORMANCES IN DIFFERENT NOISE POWERS

In this simulation, three types of noise with different powers are chosen. The power of Noise1, Noise2 and Noise3 decreases. The standard deviation of Noise2 is half of Noise1 and double of Noise3. The synchronizing cycle is set to 100μs. The parameters of the PICSs are selected as PICS1 and PICS3 in Table II. The parameters of the FTCSs are selected as PICS1 and PICS3. The simulation results are shown in Table V.

TABLE 5. Performance in different noise power.

	Convergence time/cycle	SSE/ns		
		Noise1	Noise2	Noise3
FTCS1	1	249.7400	124.9973	62.5053
FTCS3	1	180.3025	77.6052	38.9062
PICS1	2	239.4433	125.2856	60.2216
PICS3	40	164.7646	70.5104	45.6006

The SSEs of FTCS1 and PICS1 are similar in different noise power. FTCS3 and PICS3 have the better performance on noise suppression, but PICS2 spends longer time on convergence. In Noise3, the FTCS3 has the best SSE while FTCS1 has the worst. In Noise2, the PICS3 has the best SSE while PICS1 has the worst. In Noise1, PICS3 has the best SSE while the FTCS1 has the worst. From the results, FTCS3 and PICS3 have better performance on noise suppression. The noise is the major factor to influence the SSE both in FTCS and PICS.

D. DISCUSSION

PICS can obtain the faster synchronized speed and SSE by adjusting its parameters in different synchronizing cycles. However, the fastest synchronized speed is two synchronizing cycle. The performance of PICS is determined by several factors, including k_p, k_i, the synchronizing cycle, the oscillator frequency, and the noise power. The fastest synchronized speed and the best SSE cannot be achieved simultaneously. There are many uncertainties on the convergence time and SSE.

FTCS cannot achieve the best SSE, but have stable SSEs with the fastest synchronized speed in difference synchronizing cycle. The synchronized speed of FTCS is fast and only needs a single synchronizing cycle. The SSE of FTCS is mainly influenced by the parameter p and the noise power, and the synchronizing cycle weakly influence the SSE. Additionally, FTCS using lesser timestamps can achieve the same SSE as PICS with the optimal parameters. Since timestamps have the high requirement on real-time, FTCS can use less real-time resource and bandwidth to achieve the fast synchronized speed and better precision.

Though the precision of PICS is not stable, it can achieve a better precision when it gives up the optimal parameters for quick convergence. The precision of PICS can achieve a better precision than FTCS, but it needs decrease the synchronize cycle or give up quick parameters. Therefore, frequency-tracking is the better choice for clock servo in networked motion control system. Furthermore, noise power is the major factor to influence the stability of the clock both in FTCS and PICS. A filter is needed for smoothing to get a better synchronization precision.

V. EXPERIMENTAL RESULTS

A motion control system based on EtherCAT has a high requirement on clock synchronization to avoid packets losses and out-of-order. EtherCAT's DC is used to excellently synchronize slave clocks with the reference but not master-slave clock synchronization. Some embedded masters are equipped with ET1100 as the reference clock which outputs the synchronizing signal to the master controller [31]. This method can acquire better performance but it doesn't fit to PC because it needs a special network card. Therefore, the EtherCAT master should search a clock servo to synchronize with the reference clock quickly and accurately. Our motion controller is set up on a Windows-based PC and communicates with slaves through the EtherCAT. Since Windows does not have a real-time characteristic, an RTX real-time kernel based on Windows is used. As shown in Fig. 6, the platform is composed of a PC as the master, eight EtherCAT drivers (produced by Elmo) as the slaves, and the EtherCAT master developed by our team.

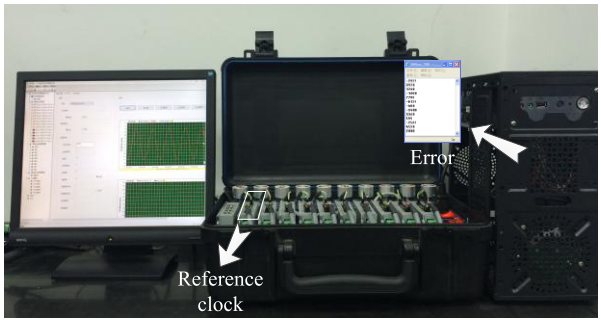


FIGURE 6. Platform of EtherCAT master clock synchronization.

RTX has three clocks which have different accuracies. We use CLOCK2 which has an accuracy of 100ns. However, the lowest resolution of CLOCK2 is $10\mu s$ to ensure normal operation of the whole system. Although RTX has a better real-time performance, we don't known the clock model of the RTX and its internal clock speed or addend register cannot be modified by users. Even so, the triggering time of the event can be changed to track the reference. It means that the event is triggered at the predicted time of the reference clock. Since motion control commands are sent by the EtherCAT master periodically, the master virtual clock synchronizes with the reference clock by adjusting the cycle time. When the master uses FTCS, from Table I, set $p = 0$ and $B = 20\mu s$, the relationship between the current cycle time ($T_m(k)$) and the next cycle time ($T_m(k+1)$) satisfies

$$T_m(k+1) = \begin{cases} (1 + \frac{E(n)}{T_{sync}})k(n)T_m(k)(|E(n)| > B) \\ k(n)T_m(k)(|E(n)| < B) \end{cases} \quad (25)$$

where $E(n)$ is the error of the n$^{\text{th}}$ synchronizing cycle, T_{sync} is the synchronizing cycle time and $k(n) = T_m/T_{ref}$ (T_m is the ideal cycle time of the cycle data and T_{ref} is the actual cycle time of the master mapped to the reference clock). When the master uses PICS, the relationship between $T_m(k)$ and $T_m(k+1)$ satisfies

$$T_m(k+1) = T_m(k) + k_p(E(n) - E(n-1)) + k_iE(n). \quad (26)$$

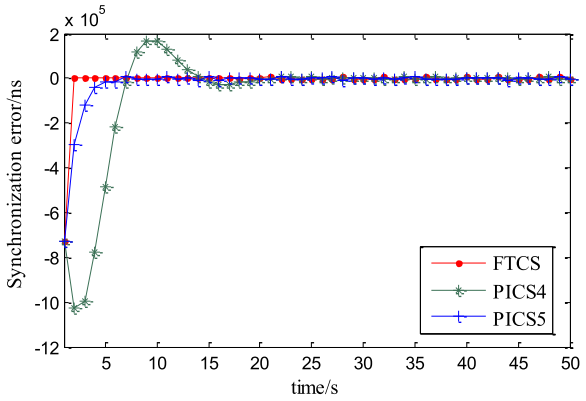


FIGURE 7. Performance of FTCS and PICS in EtherCAT master clock synchronization.

TABLE 6. Performance in EHTERCAT master.

	Sync. Cycle/s	Convergence time/cycle	SSE/ns		
			Min.	Max.	ISE
FTCS	1	1	-11159	10160	4131.5
PICS4	1	4	-12120	13520	5719.4
PICS5	1	20	-8761	11239	4048.0
FTCS	2	1	-10159	9996	4111.7
PICS6	2	5	-15320	15011	6453.8

Because the frequency of the master clock is not known, we search the optimal parameters by adjusting the parameters constantly. Then the parameters of PICS4 are $k_p = 0.02$ and $k_i = 0.01$ and PICS5 are $k_p = 0.005$ and $k_i = 0.003$. Their synchronizing cycle is set to 1s while PICS6 ($k_p = 0.01$ and $k_i = 0.01$) is set to 2s.

The results are shown in Table VI and Fig. 7. Compared with the simulation results, the convergence time of the FTCS is still a single synchronizing cycle, but the SSE is badly deteriorated and the resolution of the master clock is the major factor for this phenomenon. In the simulation, the resolution of the local clock is 1ns while the master is 10μs. Therefore, there is a great difference between their SSE. From the results, the FTCS has the perfect convergence time and the better SSE when synchronizing cycle is 1s and 2s. The PICS4 and PICS6 not only have the best convergence time but SSE. The PICS5 has the best performance on SSE, but its convergence time is 20 synchronizing cycle. From the experiment, PICS can achieve the lower SSE with the suitable parameters but it need more time to converge. FTCS can converge in one synchronizing cycle with the slightly poorer SSE. FTCS has better performances than PICS with low frequency of sending synchronization messages.

VI. CONCLUSION

A new clock servo based on frequency-tracking, called FTCS, is proposed. This clock synchronization method can be convergent in one synchronizing cycle. Its precision can be improved by dynamically adjusting its parameter p. Compared with the PICS, FTCS use less timestamps to achieve better precision. FTCS economizes on the real-time resources and bandwidth. In NMCS, clock synchronization mechanisms with fast synchronized speed, high precision, less timestamps and low computation complexity are preferable. Therefore, FTCS is the better choice. In this paper, the method of adjusting the parameter p of FTCS is simple. Parameter-adaptive method is supposed to be researched. Besides, the noise is a major factor for deteriorating synchronization precision. Therefore, a filter should be added to eliminate noise.

REFERENCES

[1] S. Wang, J. Wan, D. Li, and C. Zhang, "Implementing smart factory of Industrie 4.0: An outlook," *Int. J. Distrib. Sensor Netw.*, vol. 2016, Jan. 2016, Art. no. 3159805, doi: 10.1155/2016/3159805.

[2] S. Wang, J. Wan, D. Zhang, D. Li, and C. Zhang, "Towards the smart factory for industrie 4.0: A self-organized multi-agent system assisted with big data based feedback and coordination," *Comput. Netw.*, vol. 101, pp. 158–168, Jun. 2016.

[3] F. Chen, P. Deng, J. Wan, D. Zhang, A. Vasilakos, and X. Rong, "Data mining for the Internet of Things: Literature review and challenges," *Int. J. Distrib. Sensor Netw.*, vol. 2015, Aug. 2015, Art. no. 431047, doi: 10.1155/2015/431047.

[4] Q. Jing, A. V. Vasilakos, J. Wan, J. Lu, and D. Qiu, "Security of the Internet of Things: Perspectives and challenges," *Wireless Netw.*, vol. 20, no. 8, pp. 2481–2501, Nov. 2014.

[5] J. Wan *et al.*, "A manufacturing big data solution for active preventive maintenance," *IEEE Trans. Ind. Informat.*, to be published, doi: 10.1109/TII.2017.2670505.

[6] W. Yuan, P. Deng, T. Taleb, J. Wan, and C. Bi, "An unlicensed taxi identification model based on big data analysis," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 6, pp. 1703–1713, Jun. 2016.

[7] J. Wan, D. Zhang, Y. Sun, K. Lin, C. Zou, and H. Cai, "VCMIA: A novel architecture for integrating vehicular cyber-physical systems and mobile cloud computing," *Mobile Netw. Appl.*, vol. 19, no. 2, pp. 153–160, 2014.

[8] J. Wan, C. Zou, S. Ullah, C.-F. Lai, M. Zhou, and X. Wang, "Cloud-enabled wireless body area networks for pervasive healthcare," *IEEE Netw.*, vol. 27, no. 5, pp. 56–61, Sep./Oct. 2013.

[9] J. Skaf and S. Boyd, "Analysis and synthesis of state-feedback controllers with timing jitter," *IEEE Trans. Autom. Control*, vol. 54, no. 3, pp. 652–657, Mar. 2009.

[10] D. Orfanus, R. Indergaard, G. Prytz, and T. Wien, "EtherCAT-based platform for distributed control in high-performance industrial applications," in *Proc. IEEE Emerg. Technol. Factory Autom.*, Sep. 2013, pp. 1–8.

[11] D. Orfanus and R. Indergaard, "Recovery of distributed clock in EtherCAT with redundancy for time-drift sensitive applications," in *Proc. IEEE Emerg. Technol. Factory Autom.*, Sep. 2014, pp. 1–4.

[12] *Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, document IEC61588, 2008.

[13] D. Ganz, L. Serafin, and D. D. Hans, "Improving EtherCAT master-slave synchronization precision using PTCP embedded in EtherCAT frames: A proof-of-concept," in *Proc. IEEE World Conf. Factory Commun. Syst.*, May 2015, pp. 1–7.

[14] D. Mills, "Adaptive hybrid clock discipline algorithm for the network time protocol," *IEEE Trans. Netw.*, vol. 6, no. 5, pp. 211–219, Oct. 1998.

[15] C. Gianluca, I. C. Bertolotti, S. Scanzio, A. Valenzano, and C. Zunino, "Evaluation of EtherCAT distributed clock performance," *IEEE Trans. Ind. Inf.*, vol. 8, no. 1, pp. 20–29, Feb. 2012.

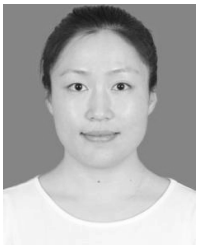
[16] *Additional Fieldbus Profiles for Real-Time Networks Based on ISO/IEC 8802-3*, document IEC61784-2, 2014.

[17] M. Cereia, I. C. Bertolotti, and S. Scanzio, "Performance of a real-time EtherCAT master under Linux," *IEEE Trans. Ind. Inf.*, vol. 7, no. 4, pp. 679–687, Apr. 2011.

[18] X. Xu, Z. Xiong, X. Sheng, J. Wu, and X. Zhu, "A new time synchronization method for reducing quantization error accumulation over real-time networks: Theory and experiments," *IEEE Trans Ind. Informat.*, vol. 9, no. 3, pp. 1659–1669, 3 Aug. 2013.

[19] J. Liu, X. Li, M. Liu, X. Cui, and D. Xu, "A new design of clock synchronization algorithm," *Adv. Mech. Eng.*, vol. 6, pp. 1–9, May 2014.

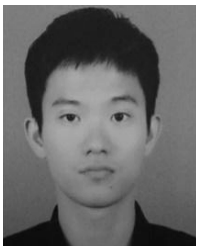
[20] R. Exel and F. Ring, "Improved clock synchronization accuracy through optimized servo parametrization," in *Proc. Int. IEEE Symp. Precision Clock Synchronization Meas. Control Commun. (ISPCS)*, Sep. 2013, pp. 65–70.
[21] Z. Fan, Y. Liu, H. Li, D. Yuan, and H. Hu, "A servo design for slave clock in IEEE 1588 synchronization networks," *Int. J. Commun. Sys.*, vol. 28, no. 4, pp. 615–624, 2015.
[22] D. Macii, D. Fontanelli, and D. Petri, "A master-slave synchronization model for enhanced servo clock design," in *Proc. IEEE Int. Symp. Precision Clock Synchronization Meas., Control Commun. (ISPCS)*, Apr. 2009, pp. 1–6.
[23] J. C. Eidson, *Measurement, Control, and Communication Using IEEE 1588* (Advances in Industrial Control). London, U.K.: Springer, 2006.
[24] G. Giorgi and C. Narduzzi, "Performance analysis of Kalman-filter-based clock synchronization in IEEE 1588 networks," *IEEE Trans. Instrum. Meas.*, vol. 60, no. 8, pp. 2902–2909, Aug. 2011.
[25] G. Giorgi and C. Narduzzi, "A resilient Kalman filter based servo clock," in *Proc. IEEE Int. Symp. Precision Clock Synchronization Meas. Control Commun. (ISPCS)*, Sep. 2013, pp. 59–64.
[26] S. Balasubramanian, K. Harris, and A. Moldovansky, "A frequency compensated clock for precision synchronization using IEEE 1588 protocol and its application to Ethernet," in *Proc. IEEE Workshop*, May 2003, pp. 91–94.
[27] C. Na, R. Scheiterer, D. Obradovic, and J. Nossek, "Clock synchronization based on distributed hidden state estimation," in *Proc. IEEE Int. Symp. Precision Clock Synchronization Meas. Control Commun.*, Oct. 2009, pp. 1–6.
[28] D. Fontanelli and D. Macii, "Accurate time synchronization in PTP based industrial networks with long linear paths," in *Proc. IEEE Int. Symp. Precision Clock Synchronization Meas. Control Commun.*, Jun. 2010, pp. 97–102.
[29] P. Wolfrum, R. Scheiterer, and D. Obradovic, "An optimal control approach to clock synchronization," in *Proc. IEEE Int. Symp. Precision Clock Synchronization Meas., Control Commun.*, Oct. 2010, pp. 122–128.
[30] N. M. Freris, S. R. Graham, and P. R. Kumar, "Fundamental limits on synchronizing clocks over networks," *IEEE Trans. Autom. Control*, vol. 56, no. 6, pp. 1352–1364, Jun. 2010.
[31] I. S. Song, Y. H. Jeon, and J. H. Kim, "Implementation and analysis of the embedded master for EtherCAT," in *Proc. Int. Conf. Control Autom. Syst. (ICCAS)*, 2010, pp. 2418–2422.



XIN CHEN received the M.S. degree from the University of Electronic Science and Technology of China. She is currently pursuing the Ph.D. degree at the South China University of Technology. Her research interests include the implementation and performance evaluation of real time network in actual application, clock synchronization in distributed system, and cyber-physical systems.

DI LI is currently a Professor with the School of Mechanical and Automotive Engineering, South China University of Technology, Guangzhou, China. Her research interests include motion control, industrial Ethernet, computer numerical control, embedded system and computer intelligence, computer vision, and cyber-physical systems.

SHIYONG WANG is currently a Lecturer with the Mechanical and Electrical Engineering Department, South China University of Technology. His research interests include smart manufacturing, motion control, and robotics. He was a recipient of the First Prize for Science & Technology Development of Guangdong Province in 2009.



HAO TANG is currently pursuing the Ph.D. degree at the School of Mechanical and Automotive Engineering, South China University of Technology, Guangzhou, China. His research interests include distributed motion control, robot control and industry 4.0.

CHENGLIANG LIU is currently a Professor with the School of Mechanical Engineering, Shanghai Jiao Tong University, Shanghai, China. He is the Distinguished Professor of the Yangtze scholar program with the Ministry of Education of China. His current interests include mechatronic systems, intelligent maintenance, MEMS design, intelligent robot control, remote monitoring techniques, and condition-based monitoring.