Available online at www.sciencedirect.com

SCIENCE DIRECT®

Microprocessors and Microsystems 30 (2006) 259–267

MICROPROCESSORS AND MICROSYSTEMS

www.elsevier.com/locate/micpro

Implementation of ILC batch update using a robotic experimental setup

Yongqiang Ye, Danwei Wang *

School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798

Received 16 June 2004; received in revised form 6 April 2005; accepted 18 November 2005
Available online 6 January 2006

Abstract

Batch update is one important feature of iterative learning control (ILC). This feature is fully exploited to simplify the development of a software platform in a robotic experimental setup for ILC. The software platform has a hierarchical structure that has two levels associated with the two phases of ILC. The low level software is rooted in a digital-signal-processor card and the high level software is hosted in the PC. This structure makes the experimental setup match the unique requirements of ILC. Moreover, a graphical-user-interface is built for real-time monitoring the results of ILC and for easy human-interaction with the software platform. The experimental setup is verified with a view of learning control in real-time and has served as a general test-bed for ILC laws.
© 2006 Elsevier B.V. All rights reserved.

Keywords: Iterative learning control; Batch update; Manipulators

1. ILC and batch update

Iterative learning control (ILC) [1] can be applied to a control system that has to execute, repeatedly, the same trajectory, the same motion or the same operation. In the presence of few stochastic disturbances, a comparable error will occur in the response of the system when the system executes a trajectory. This error can be filtered and buffered into a memory to offer a modified input signal for the next repetition. The error in the system response should be smaller after modification of the input signal until an acceptable level.

Given a desired output trajectory $y_d(t)$ for a fixed operation period $\Gamma=[0,T]$, a general form of ILC can be represented by,

$$u_j(t) = u_{j-1}(t) + kL(\cdot, e_i(\tau)) \tag{1}$$

where $i \le j$, $\tau \in \Gamma$, $e_i(\tau)=y_d(\tau)-y_i(\tau)$, $L(\cdot)$ is a function chosen by the designer and k is a scalar learning gain to adjust the learning speed. Except the current-cycle ILC [8,11], which is actually a feedback scheme, we have $i \le j-1$ in (1) and input update relies on error of past cycles which are already available. Hence, ILC is featured by batch update. A new repetition can be executed after the input update is totally completed. In other words, ILC can be divided into two phases running alternatively and continuously. One phase is the execution of a command which is usually an implementation of feedback control, and the other is the command update (1) (Fig. 1).

Any feedback control algorithm should run concurrently with the output of command and thus is termed *on-line*. Implementation of ILC is quite different and the ILC algorithm needs not be on-line with respect to the execution of command. Instead, the algorithm can run between two successive repetitions. Real-time experimental implementation of ILC only requires the ILC algorithms to be *semi-on-line* [12].

In this project, ILC batch update is implemented using a robotic experimental setup. A two-level software hierarchy is adopted and the two levels are functionally associated with the two phases of ILC. The low level, rooted in the (digital signal processing) DSP-card, makes the robot execute a given command. The high level, hosted in the PC, is the main program which contains the ILC algorithms (command update) and communicates with the low level. The PC does most jobs. The two-level arrangement greatly relieves the burden on the DSP processor and utilizes the powerful computational/memorial capacity of the PC.

The main programming language is Matlab/Simulink and C only serves as complement: the low-level software is constructed in Simulink and C; the high level is programmed in Matlab or Simulink. Using Matlab/Simulink speeds the project significantly and complex ILC algorithms can be realized straightforward. A user-friendly GUI is easily constructed via a commercial software dSPACE ControlDesk [18] for real-time observation/modification of the data/parameters in the ILC

* Corresponding author. Tel.: +65 67905376; fax: +65 67920415.
E-mail address: edwwang@ntu.edu.sg (D. Wang).

0141-9331/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.micpro.2005.11.003

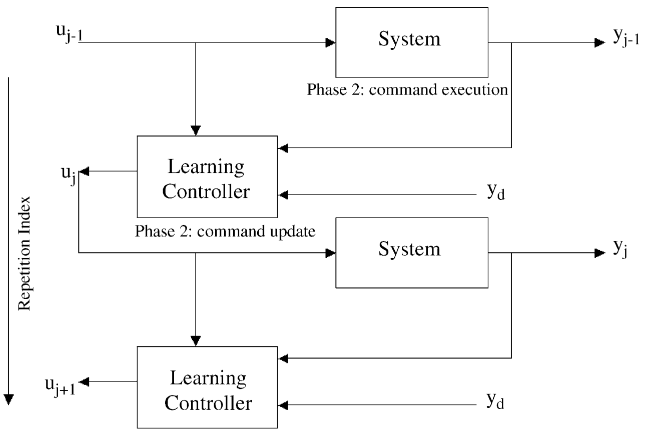

Fig. 1. Two phases of ILC.

system. Moreover, certain tricks make the GUI a unified approach to access the data in the DSP card and the PC, eliminating the need for two GUIs. The experimental setup has been shown a quick and effective tester for ILC laws [14–17].

2. Hardware system

The SEIKO D-TRAN 3000 Series robot used in this work is a four-axis, closed-loop DC servo Selective Compliant Assembly Robot Arm (SCARA) with high gear ratios [24]. Each of the four axes provides a different motion and contributes to one degree of freedom of the robot (Fig. 2).

Figure 3 shows an overview of the hardware connections. A DS1102 DSP controller board [19] sits in the 16-bit ISA slot on the PC motherboard. The Connector Panel CP1102 provides connection between the DS1102 DSP controller board and the interface board. The interface board contains signals like control signals for the various axes of the robot arm, the encoder count signal, the signals of overrun limit switches and power supply. Separate PWM DC servomotor amplifiers are used to control the respective joint movements of the SCARA. An external power supply unit supplies 30 V DC needed by the PWM DC servomotor amplifiers and 5 V DC needed for the interface board. More details about the hardware system can be found in [6].

3. Software platform tailored for ILC

3.1. Revisiting the uniqueness of ILC

First, the execution of command in ILC is finite time. This is quite different from feedback control where the execution is typically infinite. One can regard the phase of command update in ILC as an interval between repetitions. There is no apparent time constraint for this interval. Therefore, ILC algorithm is allowed to be complex and the computational burden can be heavy. In the interval, calculations other than ILC algorithms can also be implemented. For example, in [5], system identification is carried out occasionally between repetitions to update the model which is used in the ILC algorithm.

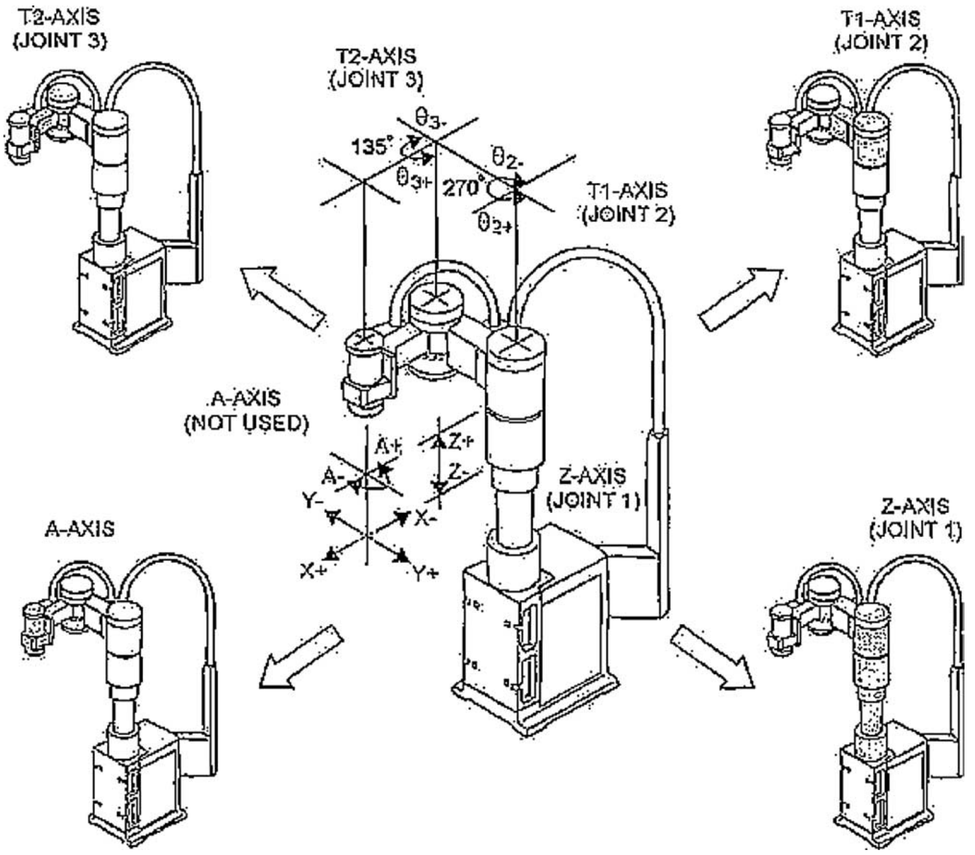

Fig. 2. Experimental robot arm.