Fuzzy Sets and Systems 110 (2000) 313–329

www.elsevier.com/locate/fss

Robot learning schemes that trade motion accuracy for command simplification[1]

Kuu-young Young*, Jyh-Fu Lee, Hui-Jun Jou

Department of Electrical and Control Engineering, National Chiao-Tung University, Hsinchu, Taiwan

Received June 1997; received in revised form November 1997

Abstract

This study was inspired by the human motor control system in its ability to accommodate a wide variety of motions. By contrast, the biologically inspired robot learning controller usually encounters huge learning space problems in many practical applications. A hypothesis for the superiority of the human motor control system is that it may have simplified the motion command at the expense of motion accuracy. This tradeoff provides an insight into how fast and simple control can be achieved when a robot task does not demand high accuracy. Two motion command simplification schemes are proposed in this paper based on the equilibrium-point hypothesis for human motion control. Investigation into the tradeoff between motion accuracy and command simplification reported in this paper was conducted using robot manipulators to generate signatures. Signature generation involves fast handwriting, and handwriting is a human skill acquired via practice. Because humans learn how to sign their names after they learn how to write, in the second learning process, they somehow learn to trade motion accuracy for motion speed and command simplicity, since signatures are simplified forms of original handwriting. Experiments are reported that demonstrate the effectiveness of the proposed schemes. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: Robotics; Learning control; Command simplification; Human motor control; Fuzzy neural network

1. Introduction

Human limbs governed by the human motor control system perform far better in many respects than those of robots, their industrial counterparts, a fact that has stimulated research into human limb motions and control strategies [4,5,9,15,17,20]. One appealing feature demonstrated by the human motor control system is that it can accommodate a wide variety of motions through effective memory management. By contrast, robot learning controllers, which are biologically inspired and intended to emulate the human motor control system, usually encounter huge learning space problems in many practical applications [16,18,21]. For example, using a learning controller, such as a neural controller or a fuzzy system, to govern the general motion of a multi-joint robot manipulator demands quite a number of training patterns, and thus the resulting neural controller consists of a huge number of neurons, or the resulting fuzzy system requires numerous rules to govern motions [23]. This learning space problem severely hinders the application of learning control to robotics.

How the human motor control system resolves the aforementioned learning space problem is of interest. Intuition suggests that the superiority of the

* Corresponding author.

[1] This work was supported in part by the National Science Council, Taiwan, under grant NSC 84-2212-E-009-060.

PII: S0165-0114(98)00062-1

human motor control system can be attributed to its salient control strategies and exceptional ability to make proper decisions using information from abundant and versatile biological sensory feedback sources. On the other hand, it has been observed that human limb motions are not very accurate, leading to the hypothesis that the human motor control system may have simplified its learning space at the expense of motion accuracy [22]. Although there is still doubt over whether the human motor control system actually performs this tradeoff, the hypothesis is not based on weak evidence. Results on the speed-accuracy tradeoff in human movements and others have been reported in several human motor control studies [17]. Inspired by this hypothesis, according to the degree of accuracy given up, motion control can be transitioned from accurate motion tracking toward point-to-point motion regulation via learning. Consequently, the original complex motion commands capable of tracking the motion accurately are simplified. With motion commands in simple forms, learning controllers can then be designed that do not consume excessive memory resources. In addition, simplified motion commands also lead to fast and simple command execution and smooth motion control with fewer oscillations.

Investigation into the tradeoff between motion accuracy and command simplification reported in this paper was conducted using robot manipulators to generate signatures. Signatures are usually generated rapidly and with little demand for motion accuracy, yet handwriting is a skilled human action [14]. The skills of handwriting and signature generation are both acquired via learning; humans learn how to sign their names after they learn how to write. In other words, humans learn to achieve simplicity in writing a signature by giving up certain degree of accuracy after they have learned how to accurately approximate the handwriting. With this appealing feature, signature generation stands as an excellent example that suits our purpose. To implement a similar tradeoff, two command simplification schemes are proposed based on the equilibrium-point hypothesis, discussed in Section 2 [17]. The rest of the paper is organized as follows. In Section 2, biological backgrounds related to the human motor control system and equilibrium-point hypothesis are presented. In Section 3, handwriting generation processes and schemes are described. The proposed command simplification schemes are discussed in Section 4. In Section 5, experimental results and analyses are reported. Finally, conclusions are given in Section 6.

2. Human motor control system and equilibrium-point hypothesis

Fig. 1 shows a simplified block diagram of the human motor control system that governs limb motion. In Fig. 1, we can see that human motion is governed by a hierarchical structure [9,10,17]. In response to various demands, the central nervous system (CNS) makes motion plans. Appropriate motor commands are then generated and sent to the peripheral neuromotor system, which may then modify the motor commands according to sensory feedback. The peripheral neuromotor system behaves as a local controller that adapts to different motions, loads, and environments, in addition to accepting commands from the CNS. Finally, the modified commands are sent to the muscular-skeletal system for motion execution. With this hierarchical structure, the difficulty of performing complex motions can be shared by the CNS at the higher level and the local controller at the lower level.

Among those hypotheses for human motion control, the equilibrium-point hypothesis suggests that the CNS specifies equilibrium points between agonist and antagonist muscle groups that correctly position limbs in relation to the target by indicating new sets of length-tension curves for the muscle groups [4,9,17]. In other words, motions are treated as transitions between postures. The CNS needs only select new levels for the motor commands. The subsequent result, mediated by autogenetic reflexes and the mechanical properties of the muscles, should be a smooth transition from one posture to another. The simple control-signal format makes the equilibrium-point hypothesis very attractive for robot motion control, although there are still debates and controversies in this hypothesis. However, since only one equilibrium point is selected, a control strategy based on this hypothesis would not enable us even to vary the motion speed between two given postures. To exploit the simplicity of the equilibrium-point hypothesis and enable it to deal with different velocities and loads in reaching various positions, motor commands may consist of numbers of equilibrium points. Thus, slow

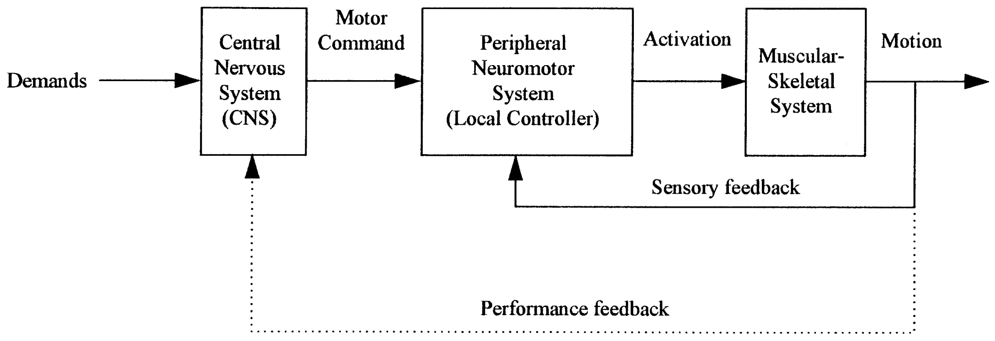


Fig. 1. A simplified block diagram of the human motor control system.

motions can be produced by progressive shifts of equilibrium points. Motions can be speeded up by assigning an initial shift that is larger than necessary, followed by a return to a proper static level [9]. In light of both physiological and engineering considerations, the number of equilibrium points in the motor command should be kept fairly small [5,11,20].

The proposed motion command simplification schemes were developed according to the equilibrium-point hypothesis. By applying the proposed schemes, the original complex motion commands for motion governance can be simplified into series of square pulses of various heights and widths [11,20,22]. With the controlled parameters in the motion command being the heights and widths of the square pulses, the learning space for dealing with variations exhibited by different motions is dramatically reduced. This motion command simplification is, however, achieved by sacrificing motion accuracy, because continuous control signals suitable for accurate tracking are approximated by signals consisting of square pulses. Note that the controlled parameter in the equilibrium-point hypothesis is muscle compliance instead of the equilibrium point used in the proposed schemes. Our purpose is not to propose a new biological hypothesis, but to develop control strategies for robot motion control inspired by the human motor control system.

3. Handwriting generation

Before the proposed motion command simplification schemes can be applied to robot manipulators for signature generation, samples of the handwriting the signature is derived from need to be provided first. In Section 3.1, handwriting generation processes are introduced along with a survey of previous handwriting generation schemes. A handwriting learning scheme (HLS) is then discussed in Section 3.2 to derive motion commands capable of tracking the handwriting for the robot manipulator. With the handwriting and its corresponding continuous, complex motion commands available, the tradeoff between motion accuracy and command simplification can be demonstrated via teaching robot manipulators to generate signatures.

3.1. Previous works

Fig. 2 shows a typical handwriting process, including four stages: cognitive decision, trajectory formation, motor command generation, and motion execution [14]. In Fig. 2, the linguistic information is first transformed into a stream of words. Because the shapes of words are usually complex, they are divided into letters and then strokes during trajectory formation. The trajectory of each stroke is planned according to various considerations, such as size, shape, speed, and location. Various criteria have been proposed for trajectory planning to accommodate different design purposes, such as energy conservation, maintenance of the bell-shaped velocity profile, and trajectory smoothness [3,7]. In the next stage, the CNS generates motor commands to realize the planned trajectories by using proper control parameters. Finally, the muscular system accepts

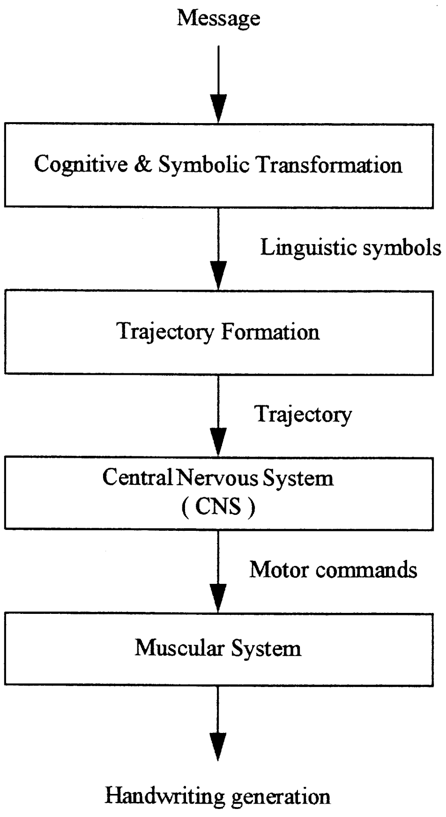


Fig. 2. The handwriting process.

commands from the CNS to execute handwriting motions. Note that the cognitive aspect of handwriting will not be dealt with in this paper, since the study is not intended for how various handwriting to be generated.

Various handwriting generation models have been proposed and can be divided into two major classes: the muscle-oriented model and the space-oriented model [13]. In the muscle-oriented model, trajectory generation is directly related to the configurations of the muscles and their mechanical properties. Most of the models reduce the complexity of the biomechanical system for handwriting by factoring it into two orthogonal functional degrees of freedom. It is generally assumed that horizontal movements are produced by rotation of the hand about the wrist, and vertical movements by oscillations of the thumb, index finger, and middle finger. Mathematical equations describing muscle dynamics are then used to generate handwriting according to different types of input stimuli. Hollerbach proposed an oscillation model for controlling the shape, height, and slant of handwriting [7]. Two orthogonal pairs of springs were used to generate the required oscillations for handwriting. Plamondon and Maarse proposed a more general model to describe and analyze biomechanical handwriting [14]. In their model, a second-order sub-system was used to represent the hand-pen-paper system and a first-order one used as a nerve-muscle interface.

In the space-oriented model, trajectory generation is based on an ability to express and control the trajectory of the hand in space, independent of the actual joint and muscle system. The model is supported by the fact that humans can write in the same way on a sheet of paper and on a blackboard, using the hand or even other parts of the body. Morasso and Ivaldi proposed a trajectory formation model for handwriting [13]. In their model, handwriting was produced by a mechanism of composition of discrete strokes represented by a weighted sum of B-splines. The mechanism was also able to generate smooth trajectories by means of timed overlapping of different strokes. Edelman and Flash proposed a handwriting generation model based on the kinematics from shape principle and on dynamic optimization [3]. Symbolic descriptions of strokes were used in their model.

3.2. Handwriting learning scheme (HLS)

Fig. 3 shows a block diagram of the proposed handwriting learning scheme based on using the two-joint planar robot manipulator shown in Fig. 4(a). Because the cognitive aspect of handwriting is beyond the scope of this paper, the process of transforming messages into chains of strokes is ignored in our scheme. In Fig. 3, human subjects first input samples of their handwriting by writing on a digital tablet. Input handwriting samples are then transformed into Cartesian trajectories $(P_d(t), V_d(t))$ in the robot workspace according to the coordinate system of the human hand (as determined through the digital tablet) via a trajectory mapping process. The Cartesian trajectory $(P_d(t), V_d(t))$ is further mapped into a joint trajectory $(q_d(t), \dot{q}_d(t))$ via an inverse-kinematic transformation. According to the joint trajectory $(q_d(t), \dot{q}_d(t))$, a fuzzy neural network (FNN), that emulates the CNS in

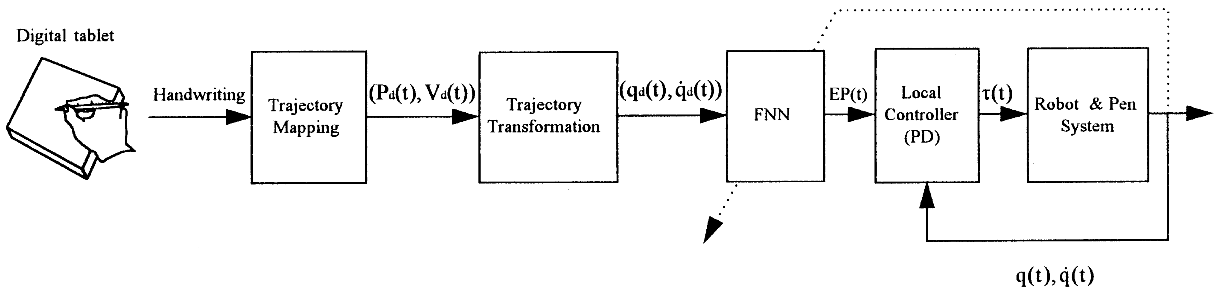


Fig. 3. The handwriting learning scheme (HLS).

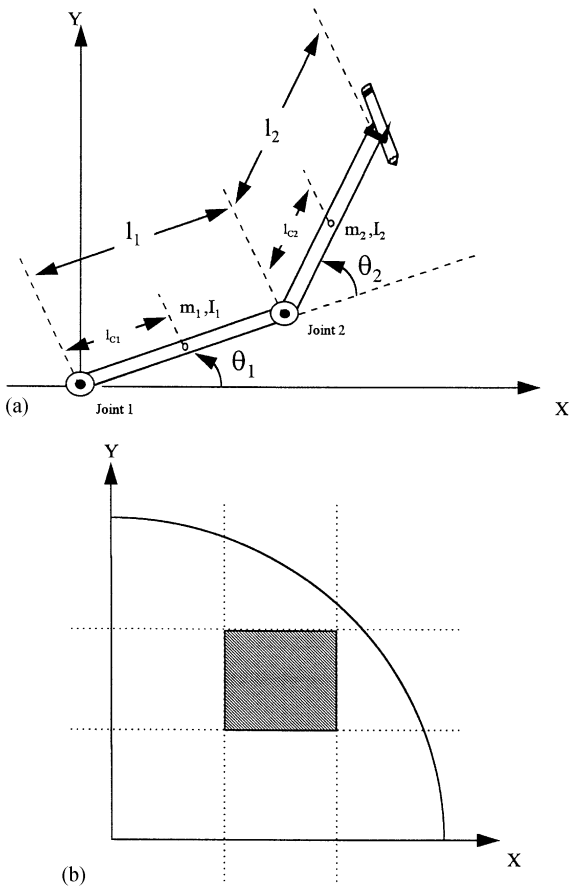


Fig. 4. (a) A two-joint planar robot manipulator. (b) Robot workspace partition.

Fig. 2, generates motion commands $EP(t)$ for trajectory tracking. Note that motion commands generated by the FNN consist of equilibrium points in continuous form. In turn, a local controller, that emulates the peripheral neuromotor system in the muscular system shown in Fig. 2, modulates the motion commands via sensory feedback and uses the resultant torque $\tau(t)$ to move the robot and pen system.

According to some biological evidence, the CNS may provide only the desired equilibrium points for motion control [15]. Therefore, to simplify the design of our scheme, only the desired equilibrium points and no desired velocities are specified in the motion commands. A simple position control law with linear damping is then used for the local controller [19]:

$$\tau = \boldsymbol{K}_p(\boldsymbol{EP} - \boldsymbol{q}) - \boldsymbol{K}_d\dot{\boldsymbol{q}}, \tag{1}$$

where $\boldsymbol{EP}$ stands for the equilibrium point vector, $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$ are the actual position and velocity vectors obtained via sensory feedback, and $\boldsymbol{K}_p$ and $\boldsymbol{K}_d$ are symmetric-positive-definite matrices for stability considerations [2].

In the proposed scheme, trajectory mapping is performed from the human hand coordinate system to that of the two-joint planar robot manipulator, because the human hand and the two-joint robot manipulator are different mechanisms with different kinematic and dynamic features, and thus they choose different optimal locations in their own workspaces and use different configurations to better handwriting generation. Meanwhile, when the handwriting is placed at different locations in the workspace for the robot manipulator to track, different learning results, consequently different equilibrium points, are obtained. To find optimal locations at which to place the handwriting, we performed a series of simulations using the proposed HLS and adopted the minimum-equilibrium-point-change criterion proposed in [6] for performance evaluation. This

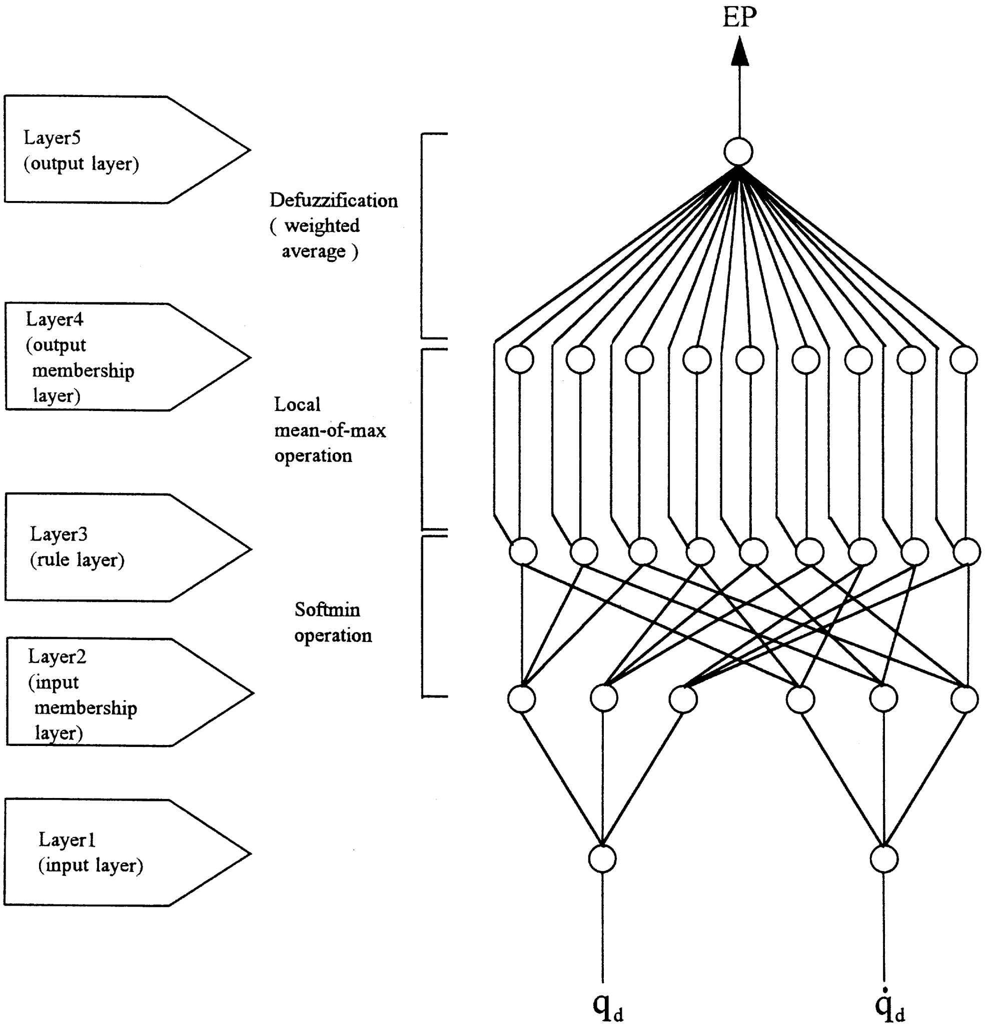


Fig. 5. The structure of the FNN.

criterion aims for a smooth transition between postures. Fig. 4(b) shows the robot workspace in the first quadrant partitioned into nine regions; several locations in each region were chosen for evaluation. Simulation results showed that when the handwriting was placed at locations chosen from the gray region in the middle portion, shown in Fig. 4(b), the derived equilibrium points for trajectory tracking most satisfied the criterion, and this region in the robot workspace was then used as the handwriting location.

The FNN for motion command generation is basically a fuzzy system implemented in the form of a neural network, as shown in Fig. 5 [1,12]. The representation of a fuzzy system using a fuzzy neural network enables us to take advantage of the learning ability of the neural network for automatic tuning of the parameters in the fuzzy system. The fuzzy reasoning parameters are thus expressed in the connection weights or node functions of the neural network. In the proposed HLS, we chose an FNN with a structure similar to that in [1]; of course, other types of FNN can also be used. As Fig. 5 shows, the inputs to the FNN are the joint position and velocity trajectories of the input motions, $(q_d(t), \dot{q}_d(t))$, and the outputs are the equilibrium point trajectories $EP(t)$. We assume that stability and convergence of the FNN in learning to track continuous trajectories are guaranteed, and these issues are well dealt with in previous studies [1,12]. Our previous results have demonstrated that the FNN is capable of governing continuous robot trajectories [23], and the results in this paper also show that the handwriting generated by the HLS approximated the originals quite well. Detailed discussions of the structure and learning process of this FNN can be found in the appendix.

4. Proposed command simplification schemes

Two schemes are proposed to implement the tradeoff between motion accuracy and command simplification. Fig. 6 shows the gradual learning simplification scheme (GLSS) and the command shape simplification scheme (CSSS). These two schemes are applied to simplify motion commands for handwriting generation into those for signature generation. Correspondingly, the continuous equilibrium point trajectories derived by the HLS in Section 3.2 are simplified into trajectories consisting of series of square pulses of various heights and widths. During the simplification process, the schemes gradually give up accuracy in approximating the handwriting trajectory, and the resulting handwriting becomes more and more like actual signatures.

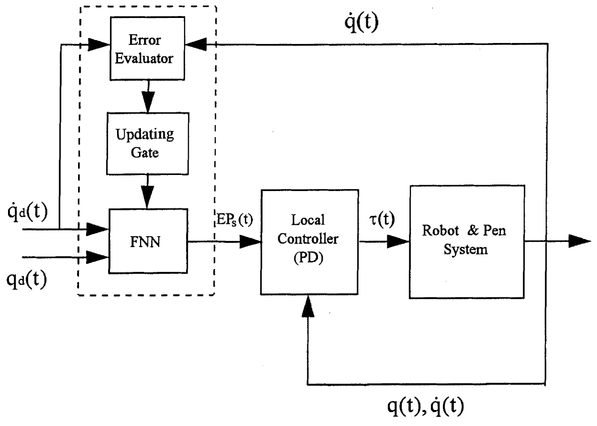
4.1. Gradual learning simplification scheme (GLSS)

Fig. 6(a) shows a block diagram of the gradual learning simplification scheme (GLSS). In the GLSS, tradeoff between motion accuracy and command simplification is achieved via a simplification process involving the error evaluator, the updating gate, and the FNN, as shown in the blocks surrounded by the dotted lines in Fig. 6(a). The FNN used in the GLSS is the same as that used in the HLS in Section 3.2 with the learning process for handwriting tracking in the HLS completed. Therefore, before the simplification process in the GLSS is executed, the input joint trajectory $(q_d(t), \dot{q}_d(t))$ will elicit from the FNN continuous equilibrium point trajectories $EP(t)$ able to track the handwriting accurately. In the GLSS simplification process, an error bound is first set in the error evaluator. This error bound indicates the amount of accuracy to be traded for command simplification for a portion of the $EP(t)$. The design will make the tradeoff be performed in each local portion, leading to a more homogeneous tradeoff over the entire trajectory. When the cumulative error in tracking the handwriting does not exceed the error bound, the updating gate is closed, preventing the FNN from continuing to update motion commands. Thus, the motion commands remain at fixed values during that period. Consequently, the resulting $EP(t)$ will be in the form of series of square pulses. By contrast, a general learning mechanism, in some sense, has the error bound set to zero, and thus updates itself at every sampling time, resulting in continuous motion commands.

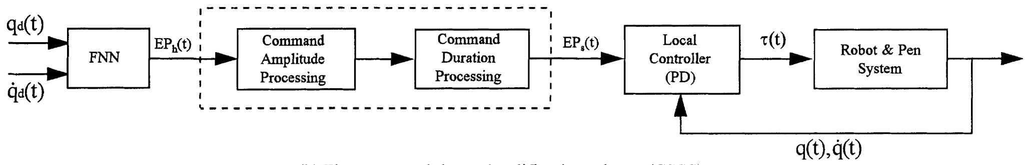
The number of square pulses in $EP(t)$ after command simplification depends on the value of the error bound: when the error bound is large (small), $EP(t)$ will have a small (large) number of square pulses. We use the joint velocity error for the error bound, because variations in joint velocities are more evident than those in joint positions. The error bound is defined as the sum of the joint one and two velocity errors, since the scales of velocity error variations of joints one and two are similar according to our observations. The motion commands for both joints will then be updated simultaneously each time the error bound is exceeded. We set the error bound to a small value at the beginning of the simplification process and increase it gradually. Intuition suggests that the final value of the error bound can be determined according to the preset similarity criterion between the original handwriting and the resulting signature. However, the resemblance between these two is quite subjective and qualitative. In order to quantitatively describe the similarity between the handwriting and the signature, we propose the concept of similarity bounding. The similarity bound E_s is defined as several times the total Cartesian error E_c between the input handwriting after trajectory mapping in the HLS, $T_i(t)$, and handwriting learning by the HLS, $T_h(t)$, as follows:

$$E_s = kE_c, \tag{2}$$

where $k \geqslant 1$ is an empirical value, standing for the degree of similarity and referred to as a similarity



(a) The gradual learning simplification scheme (GLSS).



(b) The command shape simplification scheme (CSSS).

Fig. 6. (a) The gradual learning simplification scheme (GLSS). (b) The command shape simplification scheme (CSSS).

index. A proper selection of k should make the original handwriting still recognizable from the resulting signature. The total Cartesian error E_c can be computed using the following equation:

$$E_c = \sum_{i=1}^{n} [(x_d(i) - x_h(i))^2 + (y_d(i) - y_h(i))^2]^{1/2}, \quad (3)$$

where (x_d, y_d) and (x_h, y_h) are the coordinates of the samples of $T_i(t)$ and $T_h(t)$, respectively, and n is the number of samples.

Based on the discussions above, the command simplification in the GLSS will begin with a small initial error bound. The joint velocity error will be evaluated at each sampling time. The FNN will update motion commands only when the cumulative error exceeds the error bound. After all handwriting command simplification has been completed, the total Cartesian error E between $T_i(t)$ and the resulting trajectory $T_l(t)$ is computed. When E is smaller than the similarity bound E_s, the error bound will be increased and command simplification resumed. The simplification process will proceed until E is greater than E_s. To summarize, the algorithm for the operation in the GLSS is:

Gradual Learning Simplification Algorithm: Simplify continuous equilibrium point trajectories into trajectories consisting of series of square pulses via a simplification process with feedback for evaluation using pre-specified degrees of similarity between originals and derived trajectories.

Step 1: Input $T_i(t)$, $T_h(t)$, the velocity trajectory $V_h(t)$, and the equilibrium point trajectory $EP_h(t)$ corresponding to $T_h(t)$.

Step 2: Compute the total Cartesian error E_c between $T_i(t)$ and $T_h(t)$. Determine the similarity bound E_s by selecting an empirical similarity index k.

Step 3: Initialize the error bound with a small value.

Step 4: Evaluate the joint velocity error between the current joint velocity and the reference joint velocity corresponding to $V_h(t)$ at each sampling time. Allow the FNN update motion commands only when the cumulative joint velocity error exceeds the error bound.

Step 5: Compute the total Cartesian error E between $T_i(t)$ and $T_l(t)$ after all handwriting command simplification.

Step 6: Check whether E is smaller than E_s; if yes, increase the error bound and go to Step 4; otherwise, the simplification process is completed and output the simplified equilibrium point trajectory $EP_s(t)$ as series of square pulses.

4.1.1. Command scaling

The GLSS can also be used to trade motion accuracy for simplified motion commands that generate motions similar to the original motion, but different in movement distance and velocity. By performing motion command simplification and scaling simultaneously, the GLSS is able to achieve motion command scaling without system dynamics recalculation [8]. Possible industrial applications for this feature of the GLSS can be providing simplified motion commands for industrial tasks that involve a number of similar robot motions with different movement distances and velocities.

In the application of signature generation, the equilibrium point trajectory $EP_h(t)$ corresponding to the handwriting $T_h(t)$ is simplified and scaled by the GLSS to generate signatures of different sizes with different velocities. This function of the GLSS can be achieved following the Gradual Learning Simplification algorithm above with modification of the error evaluation process in Steps 4 and 5. In the new evaluation process, $T_h(t)$ and its velocity trajectory $V_h(t)$ are scaled and used for reference. The error evaluator will then compare the trajectories generated by the algorithm with the reference trajectories after scaling, simplifying and scaling $EP_h(t)$ simultaneously. The error bound may also need to be increased (decreased) according to different size and velocity requirements. To generate signatures of different sizes, $T_h(t)$ and $V_h(t)$ can be scaled by varying the sampling time, as follows:

$$r = ct, \tag{4}$$

$$\hat{q}_h(r) = (1 - c) \times q_{h0} + c \times q_h(t), \tag{5}$$

$$\dot{\hat{q}}_h(r) = \dot{q}_h(t), \tag{6}$$

where c is a scaling constant, $(q_h(t), \dot{q}_h(t))$ are the joint position and velocity trajectories corresponding to $T_h(t)$ and $V_h(t)$, respectively, q_{h0} is the initial joint position of $q_h(t)$, and $(\hat{q}_h(r), \dot{\hat{q}}_h(r))$ are the scaled joint position and velocity trajectories, respectively. When $c > 1$, it is amplification, and vice versa. To generate signatures with different velocities, scaling is imposed upon both the sampling time and the velocity, as follows:

$$r = ct, \tag{7}$$

$$\hat{q}_h(r) = q_h(t), \tag{8}$$

$$\dot{\hat{q}}_h(r) = \frac{\dot{q}_h(t)}{c}. \tag{9}$$

When $c < 1$, it is a speed-up, and vice versa.

4.2. Command shape simplification scheme (CSSS)

Fig. 6(b) shows a block diagram of the command shape simplification scheme (CSSS). In the CSSS, the tradeoff between motion accuracy and command simplification is performed according to the characteristics of the command shapes. Local extreme values on the equilibrium point trajectories $EP_h(t)$ are first located and $EP_h(t)$ are then approximated using a series of zero-order polynomials, which replace the curves (or lines) between local extreme points with square pulses. When the difference between two adjacent pulses in the approximated $EP_h(t)$ is smaller than some pre-specified threshold in either amplitude or duration, the pulses will be combined via processing in command amplitude or duration, as shown in the blocks surrounded by the dotted lines in Fig. 6(b). Note that the amplitude and duration thresholds for the two joints of the robot manipulator may be chosen differently according to variations in the command shapes for each joint. The final similarity between the original handwriting and the resulting signature in the CSSS is also specified using the similarity bound, as in the GLSS. The amplitude and duration thresholds will be initialized with small values and be increased gradually when the total Cartesian error E between $T_i(t)$ and the resulting trajectory $T_l(t)$ after command simplification does not exceed the similarity bound. To summarize, the algorithm for the operation in the CSSS is:

Command Shape Simplification Algorithm: Simplify continuous equilibrium point trajectories into trajectories consisting of series of square pulses according to the command shape characteristics using pre-specified degrees of similarity between originals and derived trajectories.

Step 1: Input $T_i(t)$, $T_h(t)$, and the equilibrium point trajectory $EP_h(t)$ corresponding to $T_h(t)$.

Step 2: Compute the total Cartesian error E_c between $T_i(t)$ and $T_h(t)$. Determine the similarity bound E_s by selecting an empirical similarity index k.

Step 3: Initialize the amplitude and duration thresholds with small values.

Step 4: Locate local extreme values on $EP_h(t)$. Use zero-order polynomials to approximate $EP_h(t)$ by replacing the curves (or lines) between local extreme points with square pulses.

Step 5: Perform command amplitude and duration processing to combine motion commands for the approximated $EP_h(t)$.

Step 6: Compute the total Cartesian error E between $T_i(t)$ and $T_l(t)$ after command combination for the entire approximated $EP_h(t)$ is completed.

Step 7: Check whether E is smaller than E_s; if yes, increase the amplitude and duration thresholds and go to Step 5; otherwise, the simplification process is completed and output the simplified equilibrium point trajectory $EP_s(t)$ as series of square pulses.

By applying this algorithm, the final simplified equilibrium point trajectories $EP_s(t)$ will be a series of square pulses. To smooth the $EP_s(t)$, we can include a command smoothing process that approximates the square pulses of the $EP_s(t)$ using second-order polynomials. Obviously, other kinds of functions, e.g., spline functions, can also be used for approximation.

5. Result and analysis

To demonstrate the effectiveness of the two proposed command simplification schemes, the GLSS and the CSSS, they were applied to simplify the equilibrium point trajectories for handwriting generation into those for signature generation for the two-joint planar robot manipulator, shown in Fig. 4(a). Three adult subjects, two male and one female, were asked to provide handwriting samples. They practiced writing on the digital tablet for a while, and their samples were recorded after they were confident about using the digital tablet. The subjects were told to write quickly to generate more natural handwriting, and to select satisfactory samples from what they wrote according to their own standards. The selected samples were then mapped into Cartesian trajectories $T_i(t)$ in the robot workspace using the HLS described in Section 3.2. Via a learning process in the HLS, the equilibrium point trajectories $EP_h(t)$ were derived, which in turn generated trajectories $T_h(t)$ approximating $T_i(t)$.

The two-joint planar robot manipulator was used to simulate the hand and pen system, and its dynamic equations are expressed as follows:

$$\begin{bmatrix}\tau_1\\ \tau_2\end{bmatrix} = \begin{bmatrix}H_{11} & H_{12}\\ H_{21} & H_{22}\end{bmatrix}\begin{bmatrix}\ddot{\theta}_1\\ \ddot{\theta}_2\end{bmatrix} + \begin{bmatrix}-c\dot{\theta}_2^2 - 2c\dot{\theta}_1\dot{\theta}_2\\ c\dot{\theta}_1^2\end{bmatrix}, \tag{10}$$

where

$$H_{11} = m_1 l_{c1}^2 + m_2 l_1^2 + m_2 l_{c2}^2 + 2m_2 l_1 l_{c2}\cos(\theta_2) + I_1 + I_2, \tag{11}$$

$$H_{12} = m_2 l_{c2}^2 + m_2 l_1 l_{c2}\cos(\theta_2) + I_2, \tag{12}$$

$$H_{21} = H_{12}, \tag{13}$$

$$H_{22} = m_2 l_{c2}^2 + I_2, \tag{14}$$

$$c = m_2 l_1 l_{c2}\sin(\theta_2), \tag{15}$$

with τ_1 and τ_2 standing for the torques, θ_1 and θ_2 the joint variables, $m_1 = 2.815$ kg, $m_2 = 1.64$ kg, $l_1 = 0.3$ m, $l_2 = 0.32$ m, $l_{c1} = 0.15$ m, $l_{c2} = 0.16$ m, and $I_1 = I_2 = 0.0234\,\mathrm{kgm}^2$. The effects of load and gravity were ignored in the formulation, and the sampling time in the simulation was 2 ms. For all schemes, the HLS, the GLSS, and the CSSS, each joint of the robot manipulator was equipped with an FNN and a local controller. In each FNN, there were two nodes in Layer 1, ten nodes in Layer 2, 25 nodes in Layers 3 and 4, and one node in Layer 5. The local controller gains were set to $K_p = 15$ and 10 N m/rad and $K_d = 3$ and 1 N m/(rad/s) for joints one and two, respectively.

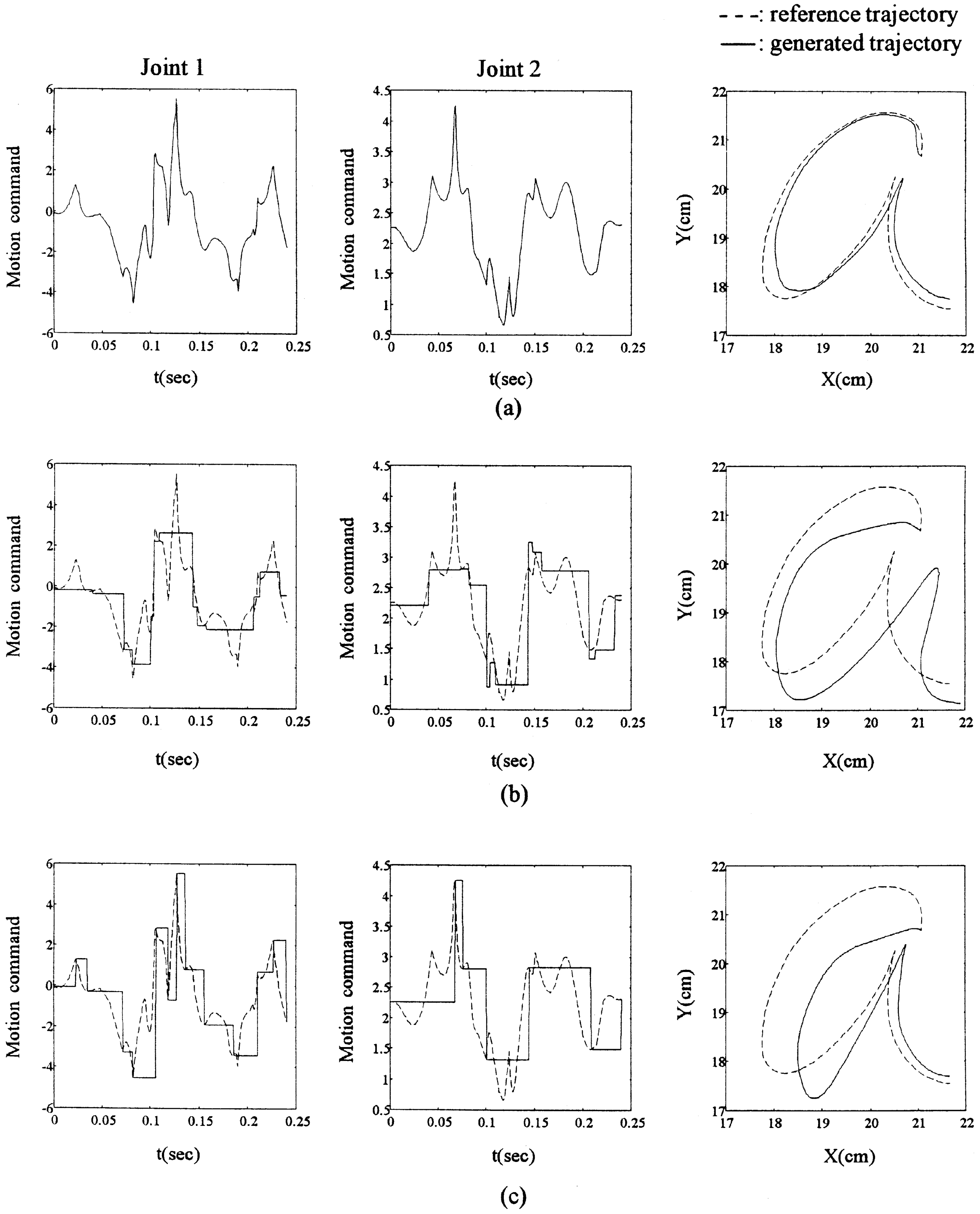


Fig. 7. Motion command simplification for the character '*a*': (a) the HLS, (b) the GLSS, and (c) the CSSS.

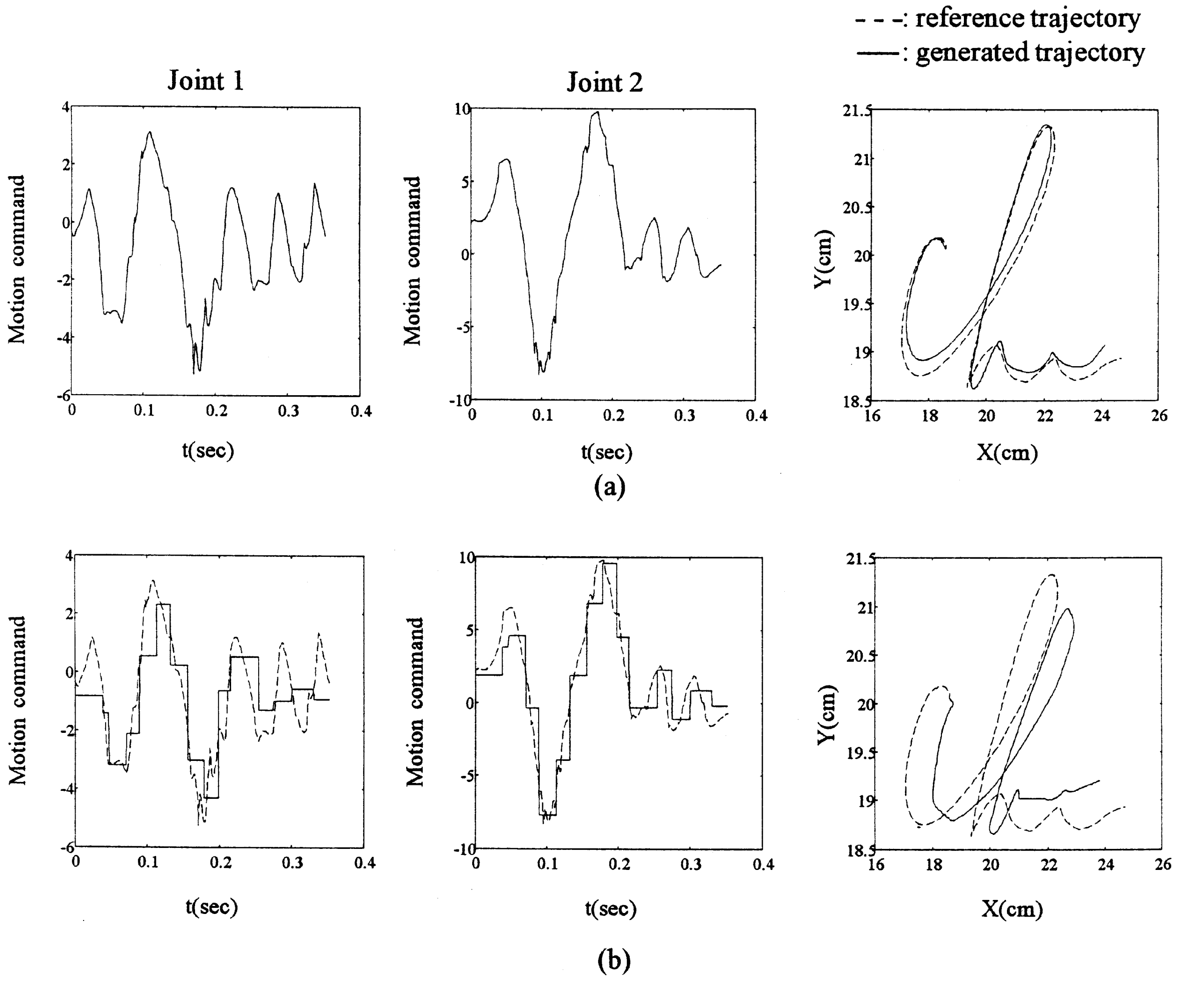
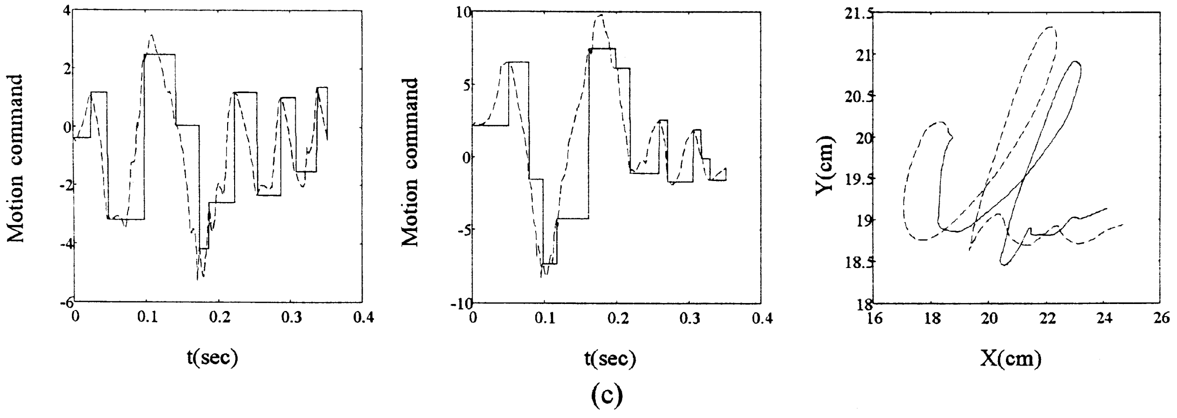


Fig. 8. Motion command simplification for the name ‘Chen’: (a) the HLS, (b) the GLSS, (c) the CSSS, and (d) the CSSS plus smoothing.

Fig. 7 shows the resulting position trajectories and the corresponding equilibrium point trajectories for an input handwritten character ‘a’ from (a) the HLS, (b) the GLSS, and (c) the CSSS. In Fig. 7(a), the total Cartesian error E_c between the input handwritten ‘a’ trajectory used for reference (dotted line) and the trajectory generated by the HLS (solid line) was computed to be about 0.1 m. The equilibrium point trajectories $EP_h(t)$ derived by the HLS were continuous, and were sent to the GLSS and the CSSS for command simplification. Figs. 7(b) and (c) show the $EP_h(t)$ from Fig. 7(a) simplified into series of square-pulse trajectories. For both the GLSS and the CSSS, the similarity index k was set to 5, making the similarity bound equal to 0.5 m. The total Cartesian errors E between the reference and generated trajectories after command simplification were about 0.53 and 0.59 m for the GLSS and the CSSS, respectively. From the results, both the GLSS and the CSSS can generate simplified motion commands that result in pre-specified degrees of similarity between the original and the derived trajectories. In general, the GLSS can generate more accurate trajectories using the same similarity bound, but consumes more computation time, as compared to the CSSS. This is because in command simplification, the GLSS uses a simpli-



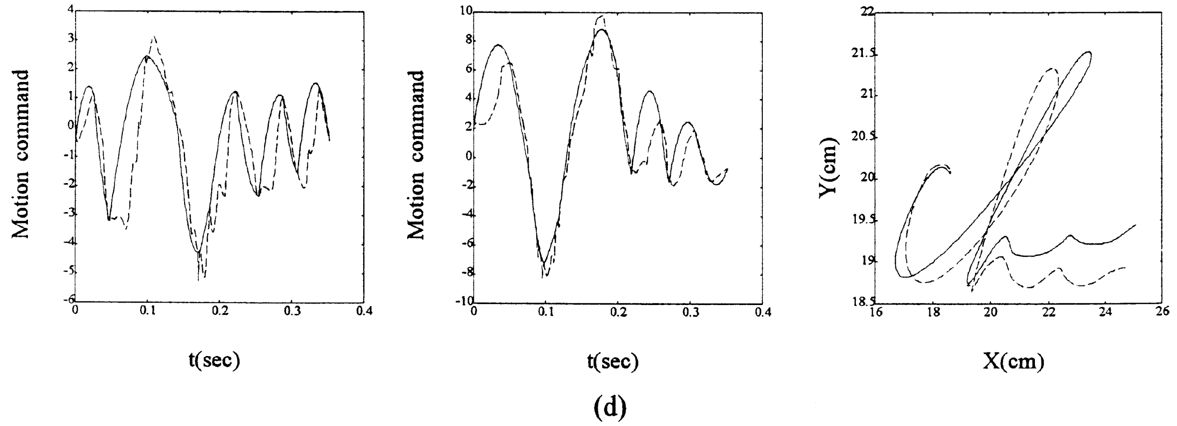


Fig. 8. Continued.

fication process with feedback for evaluation, while the CSSS performs command combination directly on the command shapes.

In the second case study, we used a more complicated sample, the name ‘Chen’, and also evaluated the effect of the CSSS when the command smoothing process described in Section 4.2 was included. Fig. 8 shows the resulting position trajectories and the corresponding equilibrium point trajectories for the input handwritten sample of the name ‘Chen’ from (a) the HLS, (b) the GLSS, (c) the CSSS, and (d) the CSSS plus smoothing. In Fig. 8(a), E_c after learning was computed to be about 0.1 m. For both the GLSS and the CSSS, the similarity index k was set to 5. In Figs. 8(b) and (c), the $EP_h(t)$ in Fig. 8(a) derived by the HLS were simplified into series of square-pulse trajectories. The number of square pulses for the handwritten ‘Chen’ was greater than that for ‘a’ as expected. The total Cartesian errors E after command simplification were about 0.53 and 0.57 m for the GLSS and the CSSS, respectively. Fig. 8(d) shows the result when the command smoothing process was included in the CSSS. Second-order polynomials were used to approximate the motion command trajectories in Fig. 8(c). In Fig. 8(d), the resulting motion command trajectories were smooth and the total Cartesian error E after command smoothing was about 0.55 m, demonstrating the feasibility of the proposed command smoothing technique.

Finally, in the third case study, we evaluated the performance of applying the GLSS with command scaling, as described in Section 4.1.1, to

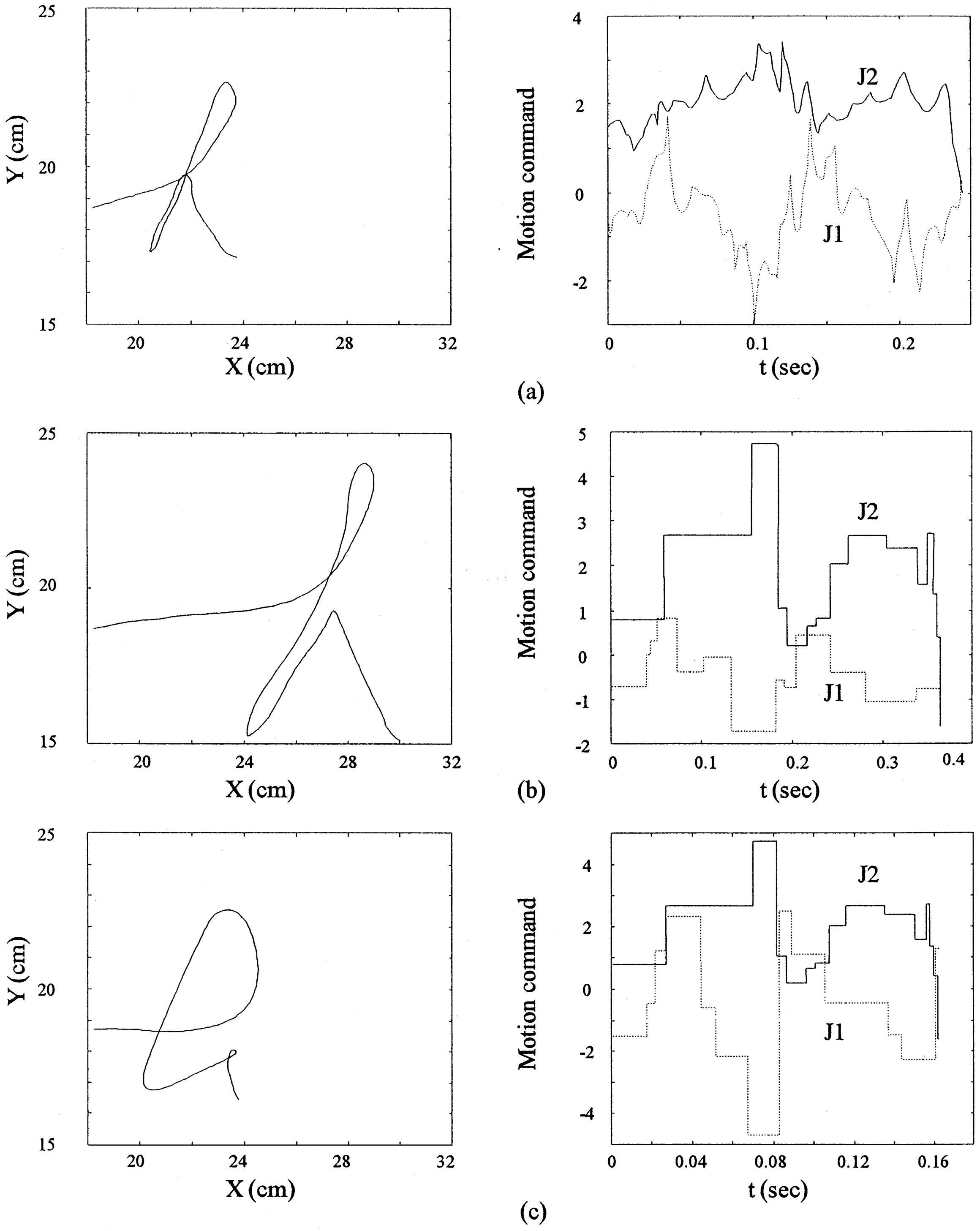


Fig. 9. Generation of the character '*h*' under different size and velocity requirements using the GLSS: (a) the reference '*h*', (b) a larger '*h*', and (c) a faster '*h*'.

generate signatures of different sizes and velocities. Fig. 9(a) shows the character '*h*' used for reference, generated by the HLS with E_c after learning about 0.1 m, and the corresponding continuous equilibrium point trajectories $EP_h(t)$. Command scaling was applied to generate a larger '*h*' and a normal '*h*' written more rapidly. The reference trajectories for error evaluation during command scaling were generated using Eqs. (4)–(9), with the scaling constants $c = 1.5$ and 1.25 for the larger '*h*' and the faster '*h*', respectively. Due to the increases in size and writing velocity, the similarity indices were increased accordingly and set to 26 and 8 for the larger '*h*' and the faster '*h*', respectively. In Figs. 9(b) and (c), the $EP_h(t)$ in Fig. 9(a) were simplified and scaled into series of square-pulse trajectories, which were able to generate larger and faster '*h*'s with errors within the similarity bounds. This demonstrates the feasibility of the proposed command scaling.

6. Conclusion

In this paper, we have developed motion command simplification schemes that can trade motion accuracy for command simplification in robot motion control. The proposed command simplification is taken as a second learning process after accurate motion tracking that demands complicated motion commands has been accomplished. Thus, the proposed schemes provide effective frameworks for achieving fast, simple control when a task does not demand high accuracy, and to transition between motion tracking and regulation according to the degree of motion accuracy given up. The results of applying the proposed schemes to simplify motion commands for handwriting generation into those for signature generation demonstrate the effectiveness of the proposed schemes. In future works, the proposed schemes will be applied to general industrial robot tasks, and to the search for simple, basic motion commands that capture fundamental motion features.

Appendix. Description of the FNN

The structure of the FNN used in the proposed schemes consists of five layers of nodes, all of which are of the same types within the same layer, as shown in Fig. 5. Each of the five layers performs one stage of the fuzzy inference process, as described below:

Layer 1. *The input layer*: It transmits inputs directly to the next layer without performing any computation. As Fig. 5 shows, there are two nodes for two inputs q_d and $\dot{q}_d$ for motions with a single degree-of-freedom.

Layer 2. *The input membership function layer*: It transforms input data into fuzzy data. Each node i in this layer has the node function

$$O_i^2 = \mu(x), \tag{A.1}$$

where $\mu : X \to [0,1]$ a membership function and x is the input to node i. The triangular membership function adopted is described below:

$$\mu(x) = \begin{cases} 1 - \dfrac{(x-b)}{c}, & x \in [b, b+c], \\ 1 + \dfrac{(x-b)}{a}, & x \in [b-a, b], \\ 0 & \text{otherwise.} \end{cases} \tag{A.2}$$

Different membership grades at the same crisp point can be obtained by adjusting the parameter set (a, b, c).

Layer 3. *The rule layer*: It implements fuzzy rules. Each node in this layer corresponds to a rule, defined as a fuzzy conditional statement of the form

Rule: *IF X is A and Y is B THEN Z is C*, (A.3)

where X and Y are fuzzy sets representing the inputs, Z represents the output, and A, B, and C represent linguistic variables, such as small, medium, and large. The number of rules involved in the input–output relationship is pre-specified. In this layer, each node also outputs the firing strength of the rule, O_i^3, by performing the differentiable softmin operation [1]:

$$O_i^3 = \frac{\sum_j O_j^2 \exp(-rO_j^2)}{\sum_j \exp(-rO_j^2)}, \tag{A.4}$$

where O_j^2 is the output of the jth node in Layer 2 connected to the ith node in Layer 3 and r is a constant. When r approaches infinity, the softmin operator becomes a min operator; for finite r, O_i^3 is differentiable, which is required during the learning process.

Layer 4. *The output membership function layer*: Each node i in this layer performs an inversion of

μ_i to locate the X-coordinate of the centroid of the membership function, O_i^4, using the local mean-of-maximum method (LMOM) [1]:

$$O_i^4 = \mu_i^{-1}(O_i^3). \tag{A.5}$$

Layer 5. *The output layer*: It has as many nodes as there are output action variables. Fig. 5 shows only one node is needed for the single motion command *EP*. The defuzzification approach adopted is the weighted averaging method:

$$O^5 = \frac{\sum_i O_i^3 O_i^4}{\sum_i O_i^3}. \tag{A.6}$$

Because the number of rules in Layer 3 is prespecified and weights for the input and output layers (Layers 1 and 5) are fixed, the parameters to be learned in this FNN are the modifiable weights present at the input links to Layers 2 and 4, which correspond to the input and output membership functions. When the FNN learns the input and output membership function parameters required to generate the motion command *EP* corresponding to a sampled motion, an error rate, related to the motion command *EP* and the resultant motion, is initially specified in the last layer (Layer 5). This error rate is then back-propagated to adjust the parameters from layer to layer sequentially. Because a concise form of the inverse dynamic model of the robot manipulator is not available, the error rate cannot be obtained directly by differentiating the error between the desired motion and the actual motion relative to the motion command. Instead, we use the combined feedback error of position (e) and velocity ($\dot{e}$) between the desired and actual motions, denoted as $E = G_p e + G_d \dot{e}$, to derive the error rate $\partial E/\partial EP$ [10]:

$$\frac{\partial E}{\partial EP} = \frac{\partial E}{\partial O^5} = \eta(G_p e + G_d \dot{e}), \tag{A.7}$$

where η is a learning rate and G_p and G_d are gains. The error rate $\partial E/\partial EP$ in Eq. (A.7) is estimated, but not exact, for describing the differential relationship between the motion command *EP* and the resultant motion. Nevertheless, the results in [10] and also ours show that the use of this error rate is appropriate for the learning. Using the error rate $\partial E/\partial EP$ and some straightforward manipulation, we are able to derive updates for the parameters in Layers 2 and 4.

References

[1] H.R. Berenji, P. Khedkar, Learning and tuning fuzzy logic controllers through reinforcements, IEEE Trans. Neural Networks 3(5) (1992) 724–740.

[2] J.J. Craig, Introduction to Robotics, Addison-Wesley, Reading, MA, 1989.

[3] S. Edelman, T. Flash, A model of handwriting, Biol. Cybernet. 57 (1987) 25–36.

[4] T. Flash, The control of hand equilibrium trajectories in multi-joint arm movements, Biol. Cybernet. 57 (1987) 257–274.

[5] G.L. Gottlieb, D.M. Corcos, G.C. Agarwal, Organizing principles for single-joint movements I. A speed-insensitive strategy, J. Neurophysiol. 62(2) (1989) 342–357.

[6] Z. Hasan, Optimized movement trajectories and joint stiffness in unperturbed, initially loaded movements, Biol. Cybernet. 53 (1986) 373–382.

[7] J.M. Hollerbach, An oscillation theory of handwriting, Biol. Cybernet. 39 (1981) 139–156.

[8] J.M. Hollerbach, Dynamic scaling of manipulator trajectories, ASME J. Dyn. Systems Measurement Control 106 (1984) 102–106.

[9] J.C. Houk, W.Z. Rymer, Neural control of muscle length and tension, in: Handbook of Physiology – The Nervous System II, Section 1, Vol. II, Ch. 8, Bethesda, MD, American Physiol. Soc., 1981, pp. 257–323.

[10] M. Kawato, K. Furukawa, R. Suzuki, A hierarchical neural-network model for control and learning of voluntary movement, Biol. Cybernet. 57 (1987) 169–185.

[11] S.L. Lehman, Input identification depends on model complexity, in: Winters and Woo (Eds.), Multiple Muscle Systems, Springer, New York, 1990, pp. 94–100.

[12] C.-T. Lin, C.S.G. Lee, Reinforcement structure/parameter learning for neural-network-based fuzzy logic control systems, IEEE Trans. Fuzzy Systems 2(1) (1994) 46–63.

[13] P. Morasso, M. Ivaldi, Trajectory formation and handwriting: a computational model, Biol. Cybernet. 45 (1982) 131–142.

[14] R. Plamondon, F. Maarse, An evaluation of motor models of handwriting, IEEE Trans. Systems Man Cybernet. 19(5) (1989) 1060–1072.

[15] A. Polit, E. Bizzi, Characteristics of motor programs underlying arm movements in monkeys, J. Neurophysiol. 42(1) (1979) 183–194.

[16] T.D. Sanger, Neural network learning control of robot manipulators using gradually increasing task difficulty, IEEE Trans. Robotics Automat. 10(3) (1994) 323–333.

[17] R.A. Schmidt, Motor control and learning: a behavioral emphasis, 2nd ed., Human Kinetics Publishers, Champaign, IL, 1988.

[18] T. Shibata, T. Fukuda, Hierarchical intelligent control for robotic motion, IEEE Trans. Neural Networks 5(5) (1994) 823–832.

[19] M. Takegaki, S. Arimoto, A new feedback method for dynamic control of manipulators, ASME J. Dyn. Systems Measurement Control 103(2) (1981) 119–125.

[20] C.H. Wu, K.Y. Young, K.S. Hwang, S. Lehman, Voluntary movements for robotic control, IEEE Control Systems Magazine 12(1) (1992) 8–14.

[21] B.-H. Yang, H. Asada, Progressive learning and its application to robot impedance learning, IEEE Trans. Neural Networks 7(4) (1996) 941–952.

[22] K.Y. Young, C.C. Fan, An approach to simplify the learning space for robot learning control, Fuzzy Sets and Systems 95(1) (1998) 23–38.

[23] K.Y. Young, S.J. Shiah, An approach to enlarge learning space coverage for robot learning control, IEEE Trans. Fuzzy Systems 5(4) (1997) 511–522.