

A REPETITIVE SEGMENTED TRAINING NEURAL NETWORK CONTROLLER WITH APPLICATIONS TO ROBOT VISUAL SERVOING

P. Jiang* and Y. Chen**

Abstract

The authors design a neural network controller for a nonlinear system with uncertainties that are invariant or repetitive over repeatedly executed tasks. The training of the neural networks is carried out iteratively as the task repeats. The desired trajectory is segmented, and for each segment a local neural network is constructed to keep the training errors within a permitted region. Meanwhile, the training is segment-wise progressively from the starting segment to the ending one. The accurate tracking of the whole desired trajectory is thus accomplished in a step-by-step or segment-by-segment manner, which means that the training of the second segment starts after the first segment tracking has reached a desired accuracy level. To guarantee the uniform boundedness of the point-wise training, a projection-type learning update law and deadzone technique are proposed. As an application example, a robot visual servoing control problem with an uncalibrated camera is considered. The effectiveness of the proposed neural network controller with repetitive segmented training is demonstrated by simulations of a typical robot visual servoing task, robot movement learning from demonstration.

Key Words

Neural networks, iterative learning control, nonlinear control, visual servoing


1. Introduction


Ordinarily, there are two different methods that can be used to cope with the control problem of an uncertain or unknown nonlinear system: linear adaptive control [1, 2] and robust control [3, 4]. Generally speaking, robust control uses a sufficient feedback to restrain the influence of uncertainties and linear adaptive control uses an adaptive feedforward mechanism to compensate uncertainties.

Compared with robust control, linear adaptive control relies more on prior model knowledge, for example, linearly parameterized model. In order to control a nonlinear system with an unknown model structure, neural networks (NN) have been introduced to learn and reconstruct the unknown nonlinearities (e.g., GRBF NN [5], multilayer NN [6], CMAC NN [7], and Fuzzy NN [8]). These approaches often worked by combining the linear adaptive control technique with the robust control technique. Whereas NNs are used to approximate uncertainties with only unknown linear weights, the linear adaptive technique is adopted to update the weights and the residual modelling errors are controlled by a robust control scheme.

In the approaches just mentioned, the desired operation was supposed to be continual and the asymptotic tracking was achieved through persistent training along time horizon. For a *nonperiodic trajectory with a finite time interval*, which is widely applied in practical systems such as cutting, painting, robot pick-and-place operation, and multi-times resetting, tracking becomes a choice to keep the training persistently. However, multi-times resetting violates the assumption of continual operation along the time horizon of the linear adaptive control. Iterative Learning Control (ILC) [9] is a more specialized technology for this kind of application and can be considered an efficient alternative.

In this article, we present an NN iterative training scheme for tracking a *nonperiodic trajectory with a finite time interval* based on ILC. The NNs are used as approximators of unknown nonlinearities instead of the common structure assumptions in ILC, such as meeting a global Lipschitz condition [10–13], with known linearly parameterized nonlinearities [14, 15]. Unlike the adaptive NN controls, we give every point (every servo period in application) along the desired trajectory an independent local NN, and the independent networks are trained according to the previous tracking history. This makes segmented training possible because the training of any segment of a trajectory does not interfere the other segments. In common with neural network-based controls, during the early

* Department of Cybernetics and Virtual Systems, University of Bradford, Bradford, BD7 1DP, UK; e-mail: 

** Department of Electrical and Computer Engineering, UMC 4160, College of Engineering, 4160 Old Main Hill, Utah State University, Logan, UT 84322-4160, USA; e-mail: 
(paper no. 201-1344)

stages of training the system may deviate from an admissible region Ω for NN approximation. A big deviation from the nominal trajectory must be avoided because it means invalidity of an NN control [3] and may cause collision or even damage in some applications. The segmented training of the ILC NNs can help contain the tracking error always within the admissible region during training. It measures the maximum tracking error along a trajectory during the training. If the error at any trajectory point exceeds the predefined region, then the trajectory is divided into two segments from this point on. After that, only the NNs related to the first segment are trained so that the whole tracking of this segment reaches a desired precision. Then the trained segment can be extended to the remaining trajectory. It works in a step-by-step or segment-by-segment manner to make the whole trajectory trained. Consequently, no tracking may happen outside the predefined region for the NN approximation. In fact, the step-by-step manner is also the learning manner of human beings. People tend to resolve a complex project into several segments and then learn all simpler segments repetitively. Until some basic goals are reached, they will not move to the next step. In addition, bounded training implies that the modelling errors can always be restricted within any given admissible region even with a low-order approximation model. Therefore, from another viewpoint, the problem of modelling error with an upper bound [16, 17] in NN control is solved.

The application background of this research is robot movement learning from demonstration with the help of an uncalibrated camera (or cameras). It includes two phases, teaching phase and training phase. During the teaching phase, a teacher grasps a tool or simply an object to do a demonstration and a static camera records the trajectories of some selected features of the object on the image plane, which represent a desired movement. During the training phase, a robot grasps the same object to imitate the movement repetitively. With the aid of the proposed NN iterative training scheme, an accurate replay of the demonstrated movement can be achieved gradually. This is a convenient means for robot trajectory programming.

In the next section, we describe the structure of the iterative learning NNs for tracking a nonperiodic trajectory with a finite time interval. In Section 3 we present a learning scheme for NN weights in the form of a positive-definite discrete matrix kernel and prove that every point along a segment will converge to a prespecified deadzone after tracking the segment repetitively. In Section 4 the scheme for segment extension is presented. It ensures uniformly bounded training for the whole trajectory. The presented controller is further applied to a robot motion imitation in Section 5, and conclusions are given in Section 6.

2. Iterative Learning Neural Networks

For a nonperiodic desired trajectory $x_d(t)$ with a finite time interval $0 \leq t \leq t_f$, iterative learning control can be considered as an alternative of the aforementioned adaptive control. The objective is to force the state vector $x(t, i)$ of

the subsequent affine nonlinear system to follow the desired trajectory after repetitive training:

$$\dot{x}(t, i) = f(t, x) + G(t, x)u(t, x, i) \quad 0 \leq t \leq t_f \quad (1)$$

where i indicates the i^{th} iterative training or tracking, $f(t, x) \in R^n$ and $G(t, x) \in R^{n \times m}$ are unknown nonlinear continuous functions, and $x(t, i) \in R^n$, $u(t, x, i) \in R^m$ are the state and the control input of the i^{th} training at time instant t , respectively.

The system (1) is described in terms of both continuous time t and discrete iterative index i . Although the time t is within a finite interval, the ILC can ensure that the entire trajectory converges to the desired one by repetitive tracking. Note that system (1) is a more general class of nonlinear systems than the one discussed in the adaptive NN control, where the system has to be autonomous with $f(x)$ and $G(x)$ but, in system (1), $f(t, x)$ and $G(t, x)$ can be time-varying provided they are repetitive over iterations. Therefore, the control $u(t, x, i)$ can integrate both time domain feedback to handle disturbance and experience-based learning for the repeated uncertainties.

At first, we employ a linear parametric network to approximate the nonlinear functions $f(t, x)$ and $G(t, x)$ as follows:

$$\begin{aligned} f(t, x) &= W_f(t)\varphi(t, x) + \varepsilon_1(t, x) \\ G(t, x) &= G^*(t, x) + \varepsilon_2 = \begin{bmatrix} \varphi(t, x)^T W_1^*(t) \\ \vdots \\ \varphi(t, x)^T W_n^*(t) \end{bmatrix} + \varepsilon_2(t, x) \end{aligned} \quad (2)$$

where $W_f(t) \in R^{n \times L}$ and $W_l(t) \in R^{L \times m}$, $l = 1 \dots n$, are the corresponding unknown optimal weights of the neural networks at a specific instant t . They are invariant over iterations. $\varepsilon_1(t, x)$ and $\varepsilon_2(t, x)$ denote modelling errors of the approximation, and their norms are supposed to be bounded on a compact region Ω . $\varphi(t, x) \in R^L$ is a vector composed of basis functions that depend on what kind of neural networks we intend to use, for example:

1. $\varphi_i(t, x) = e^{-\|x - p_i\|^2 / \sigma_i^2}$ for a Gaussian RBF neuron, where p_i is the centre of the i^{th} RBF neuron and the σ_i is the width of the neuron [5, 17];
2. $\varphi_i(t, x) = \prod_{j \in I_i} x_j^{d_j(i)}$ for a high-order neuron, where $d_j(i)$ is a nonnegative integer [18];
3. $\varphi_i(t, x) = \frac{\prod_{j=1}^n \mu_{A_j}^i(x_j)}{\sum_{i=1}^L \left(\prod_{j=1}^n \mu_{A_j}^i(x_j) \right)}$ for a fuzzy neuron, where $\mu_{A_j}^i$ is the membership function of a fuzzy set [8].

The approximation model can even be any mathematical approximation model with linear parameters such as a Taylor series, spline functions, and so on

Note that the structure of the above neural networks is different from the adaptive neural networks [5–8] that distribute neurons over a region Ω containing the whole desired trajectory, as shown in Fig. 1. It is assumed that the optimal weights are unknown constants on the

approximation region Ω , and then the weight training is carried out along the time axis t . In equation (2), the optimal weights $W_f(t)$ and $W_l(t)$ can be time-varying and the networks are used to approximate time-varying but repeatable uncertainties. This means that a series of local networks in workspace are further distributed along the time axis around the desired trajectory as shown in Fig. 2, and then the weight training is carried out along the iteration axis i by repetitive tracking. This implies that we give every specific time t (every sample period in application) or every point along the desired trajectory a local NN, and then the local NN at the instant t is trained by previous tracking experience at the same instant. In fact, each local NN tries to learn an approximation model in a neighbourhood around a specific point of the desired trajectory.

Figure 1. Adaptive.

Figure 2 Iterative learning NN.

Therefore, we can expect that, at first, local NNs along the desired trajectory can be trained persistently through repetitive tracking. Second, due to a series of time-varying local NNs, the presented scheme can be used for a time-varying uncertain system if the uncertainties are invariant over iterations but the adaptive NN controller only works for an autonomous system. Last, an efficient ‘‘segmented training’’ can be accomplished by the ILC. As each point along the desired trajectory has an independent local NN, the training of a segment $x_d(t), 0 \leq t \leq T_k$, only adjusts the weights of the corresponding local networks but has no interference with the other segments $x_d(t), T_k < t \leq t_f$. This means that we can divide a trajectory into several segments to realize segmented training, where a segment of the trajectory is trained repetitively until some requirements have been met and further training of the other segments does not cause any interference to this well-trained segment.

Because a continuous process is considered in this research, our method cannot track a segment $x_d(t), T_k \leq t \leq T_m$ without tracking the previous segment $x_d(t), 0 \leq t \leq T_k$, in Fig. 2. All of the segments in this work are defined to be started from the start point $x_d(0)$, such as segment $x_d(t), 0 \leq t \leq T_k$ or segment $x_d(t), 0 \leq t \leq T_m$, and the presented segmented training is a process of segment extension. At the beginning, because of lack of knowledge about the system, a relatively short segment only, for example $x_d(t), 0 \leq t \leq T_k$, is repetitively trained to avoid a large deviation from the desired trajectory. After the networks of this segment have been well trained, the training can be extended to a longer segment $x_d(t), 0 \leq t \leq T_m$ and eventually to the whole trajectory $x_d(t), 0 \leq t \leq t_f$.

3. Iterative Learning Control for a Segment of Trajectory

Suppose the k^{th} segment as $x_d(t), 0 \leq t \leq T_k$, and substitute equation (2) into (1); the repeated tracking of the trajectory segment can be expressed as:

$$\begin{aligned} \dot{x}(t, i) &= W_f(t)\varphi(t, x) + \varepsilon_1(t, x) \\ &+ (G^*(t, x) + \varepsilon_2(t, x))u(t, x, i) \quad 0 \leq t \leq T_k \end{aligned} \quad (3)$$

At first, let the i^{th} tracking error be

$e(t, i) = [e_1(t, i), \dots, e_n(t, i)]^T = x_d(t) - x(t, i)$. In order to deal with the modelling error and the initial resetting error, we define a modified error vector with deadzone [5] as:

$$\begin{aligned} e_\Delta(t, i) &= e(t, i) - \phi(t, i) \\ \phi(t, i) &= [\varepsilon_{f1} \text{sat}(e_1(t, i)/\varepsilon_{f1}), \dots, \varepsilon_{fn} \text{sat}(e_n(t, i)/\varepsilon_{fn})]^T \end{aligned} \quad (4)$$

where $\varepsilon_f = [\varepsilon_{f1}, \dots, \varepsilon_{fn}]^T$ is an n -dimensional width of the deadzone.

Then the usual initial resetting condition $x(0, i) = x_d(0), \forall i$, used in ILC is weakened to $|e_j(0, i)| < \varepsilon_{fj}, j = 1 \dots n, \text{ for } \forall i, [19]$. This means that, for every tracking, the initial states should be controlled into this predefined deadzone. It is a practical assumption about initial states.

Furthermore, we utilize the following neural networks as the approximators:

$$\hat{f}(t, x) = \hat{W}_f(t, i)\varphi(t, x) \text{ and } \hat{G}(t, x, i) = \begin{bmatrix} \varphi(t, x)^T \hat{W}_1(t, i) \\ \vdots \\ \varphi(t, x)^T \hat{W}_n(t, i) \end{bmatrix} \quad (5)$$

where $\hat{W}_f(t, i)$ and $\hat{W}_l(t, i), l = 1 \dots n$, are the i^{th} identifications of the optimal weights in equation (2).

We make the following assumptions concerning the modelling errors.

Assumption 1: On the compact region Ω , the L_∞ norm of the modelling error $\varepsilon_1(t, x)$ is bounded with a known constant $\|\varepsilon_1\|_\infty$.

Assumption 2: On the compact region Ω , the estimate of the input matrix $G(t, x)$ can ensure that the equation $\hat{G}(t, x, i)x = y$ has a solution, the norm of the relative estimate error $E = \varepsilon_2(t, x)\hat{G}(t, x, i)^+$ is upper bounded by a known constant $\|E\|_M$, and $\|E\|_M < |\varepsilon_f|_m / |\varepsilon_f|_M \leq 1$, where $|\varepsilon_f|_m$ and $|\varepsilon_f|_M$ are the minimum width and the maximum width of the deadzone in (4), respectively.

Assumption 2 is an assumption to avoid control singularity specifically for systems with unknown input matrix. Its satisfaction requires some prior knowledge about the unknown input matrix, that is, the NNs should be able to approximate the input matrix with a relative error less than the ratio between $|\varepsilon_f|_m$ and $|\varepsilon_f|_M$. When all deadzone widths in (4) are equal, the allowed maximum modelling

error for the input matrix can reach 100%. Assumption 2 is similar to the condition required by the Frobenius-Perron Theorem that is used for multi-input system sliding mode control [20]. It is the simplest way to cope with an uncertain input matrix if some prior knowledge about the matrix is available.

Taking networks (5) as an estimation of the uncertainties and applying feedback linearization, a control with least norm can be proposed as:

$$u(t, x, i) = \hat{G}(t, x, i)^+ \left(\hat{x}_d(t, i) - \hat{f}(t, x, i) + K_1 e(t, i) + K_2 e(t, i) \right) \quad (6)$$

where $\hat{G}(t, x, i)^+$ denotes the pseudo-inverse matrix of $\hat{G}(t, x, i)$.

In this control law, we combine NN compensations with robust control (linear feedback), where the NNs are used to approximate and estimate uncertainties of f and G and the residual modelling errors $\varepsilon_1(t, x)$ and $\varepsilon_2(t, x)$ are controlled by the linear feedback of $K_1 e(t, i)$, $K_2 e(t, i)$. The unknown weights of NNs can be updated using ILC technique. In addition, $\hat{x}_d(t, i)$ is an estimated velocity of the desired trajectory. If $\dot{x}_d(t)$ is available to use, as assumed in the adaptive NN control, we could directly assign $\dot{x}_d(t)$ to $\hat{x}_d(t, i)$. In some cases, when the desired velocity has to be obtained by differentiating a given trajectory, for example, the derivative may be sensitive to the image noise for the feature-based visual servoing, we can treat the desired velocity as a repeated uncertainty and estimate it through iterations as well.

Substituting (6) into equation (3) gives:

$$\begin{aligned} \dot{x}(t, i) &= W_f(t) \varphi(t, x) + \varepsilon_1(t, x) \\ &+ \left(\hat{G}(t, x, i) + \varepsilon_2(t, x) \right) u(t, x, i) \\ &\stackrel{\text{③}}{=} G^*(t, x) - \hat{G}(t, x, i) \Big) u(t, x, i) \\ &= \hat{x}_d(t, i) + \left(W_f(t) - \hat{W}_f(t, i) \right) \varphi(t, x) \quad (7) \\ &+ \left(G^*(t, x) - \hat{G}(t, x, i) \right) u(t, x, i) \\ &\quad + \left(\varepsilon_1(t, x) + K_1 e(t, i) \right) \\ &\quad + \left(\varepsilon_2(t, x) u(t, x, i) + K_2 e(t, i) \right) \end{aligned}$$

where we have used the relation of:

$$\hat{G}(t, x, i)x = y \text{ has a solution } \Leftrightarrow \hat{G}(t, x, i)\hat{G}(t, x, i)^+x = x$$

The next lemma provides a criterion for designing an iterative learning controller [14, 19]:

Lemma 1: If the width of the deadzone in equation (7) is kept to be unchanged for a whole segment and a weak initial resetting condition $|e_j(0, i)| < \varepsilon_{fj}$, $j = 1 \dots n$, for $\forall i$, is satisfied, then the upper bounded condition

$\lim_{N \rightarrow \infty} \sum_{i=0}^N e_{\Delta}^T(t, i) \dot{e}(t, i) \leq \gamma_0^2$, $\forall 0 \leq t \leq T_k$, ensures that the tracking error of every point along the segment $x_d(t)$, $0 \leq t \leq T_k$, will converge to the deadzone, that is, $\forall 0 \leq t \leq T_k$, when the system tracks the desired trajectory repetitively.

Proof: See Appendix.

In order to satisfy the upper bounded condition in Lemma 1, the following property of a *positive definite discrete matrix kernel* [1] can provide a more general learning form except simple P type or D type ILC.

Lemma 2: For any vector $y(i)$ and any unknown constant vector a , a *positive-definite discrete matrix kernel* $F(i)$, whose z-transform is a *positive real discrete transfer matrix* and with a pole at $z = 1$ can ensure that the following accumulation is always upper bounded:

$$\sum_{i=0}^N y^T(i) \left(a - \sum_{j=0}^i F(i-j)y(j) \right) \leq \gamma_0^2, \forall N$$

In this work, we use γ_0^2 for describing any bounded positive scalar.

Now, we are going to examine the upper bounded condition in Lemma 1 for the system (7):

$$\begin{aligned} \sum_{i=0}^N e_{\Delta}^T(t, i) \dot{e}(t, i) &= \sum_{i=0}^N e_{\Delta}^T(t, i) (\dot{x}_d(t) - \dot{x}(t, i)) \\ &= \sum_{i=0}^N e_{\Delta}^T(t, i) \left(\dot{x}_d(t) - \hat{x}_d(t, i) \right) \\ &\quad + \sum_{i=0}^N e_{\Delta}^T(t, i) \left(\hat{W}_f(t, i) - W_f(t) \right) \varphi(t, x) \quad (8) \\ &\quad + \sum_{i=0}^N e_{\Delta}^T(t, i) \left(\hat{G}(t, x, i) - G^*(t, x) \right) u(t, x, i) \\ &\quad - \sum_{i=0}^N e_{\Delta}^T(t, i) (\varepsilon_1(t, x) + K_1 e(t, i)) \\ &\quad - \sum_{i=0}^N e_{\Delta}^T(t, i) (\varepsilon_2(t, x) u(t, x, i) + K_2 e(t, i)) \end{aligned}$$

For the first term in equation (8), although the desired trajectory $\dot{x}_d(t)$ is a function of time t , it remains constant over iterations, that is, it can be repeated. From Lemma 2, if we select the learning law of the desired velocity through iterations as:

$$\hat{x}_d(t, i) = \sum_{j=0}^i F(i-j) e_{\Delta}(t, j) \quad (9)$$

where $F(i-j)$ can be any positive-definite discrete matrix kernel, then we get:

$$\begin{aligned} \sum_{i=0}^N e_{\Delta}^T(t, i) \left(\dot{x}_d(t) - \hat{x}_d(t, i) \right) \\ = \sum_{i=0}^N e_{\Delta}^T(t, i) \left(\dot{x}_d(t) - \sum_{j=0}^i F(i-j) e_{\Delta}(t, j) \right) \leq \gamma_0^2, \quad (10) \\ 0 \leq t \leq T_k \end{aligned}$$

The second term can be expressed as:

$$\begin{aligned} \sum_{i=0}^N e_{\Delta}^T(t, i) \left(\hat{W}_f(t, i) - W_f(t) \right) \varphi(t, x) \\ = \sum_{i=0}^N \text{tr} \left(\varphi(t, x) e_{\Delta}^T(t, i) \left(\hat{W}_f(t, i) - W_f(t) \right) \right) \end{aligned}$$

For a similar reason, $W_f(t)$ can be repeated. If we adopt the following updating law for the weight matrix:

$$\hat{W}_f(t, i) = - \sum_{j=0}^i F_f(i-j) e_{\Delta}^T(t, j) \varphi(t, x)^T \quad (11)$$

and apply Lemma 2 again to every column vector of the matrix $e_{\Delta}^T(t, j) \varphi(t, x)^T$, then:

$$\sum_{i=0}^N e_{\Delta}^T(t, i) \left(\hat{W}_f(t, i) - W_f(t) \right) \varphi(t, x) \leq \gamma_0^2, \quad 0 \leq t \leq T_k \quad (12)$$

where $F_f(i-j)$ is a positive-definite discrete matrix kernel.

From equations (2) and (5), the next term can be written as:

$$\begin{aligned} & \sum_{i=0}^N e_{\Delta}^T(t, i) \left(\hat{G}(t, x, i) - G^*(t, x) \right) u(t, i) \\ &= \sum_{i=0}^N e_{\Delta}^T(t, i) \begin{bmatrix} \varphi(t, x)^T \left(\hat{W}_1(t, i) - W_1(t) \right) \\ \vdots \\ \varphi(t, x)^T \left(\hat{W}_n(t, i) - W_n(t) \right) \end{bmatrix} u(t, x, i) \\ &= \sum_{i=0}^N \sum_{l=1}^n e_{\Delta l}(t, i) \varphi(t, x)^T \left(\hat{W}_l(t, i) - W_l(t) \right) u(t, x, i) \\ &= \sum_{l=1}^n \sum_{i=0}^N \text{tr} \left(e_{\Delta l}(t, i) u(t, x, i) \varphi(t, x)^T \left(\hat{W}_l(t, i) - W_l(t) \right) \right) \end{aligned}$$

For the same reason, if the network weight matrix is trained by:

$$\begin{aligned} \hat{W}_l(t, i) &= - \sum_{j=0}^i F_l(i-j) e_{\Delta l}(t, j) \varphi(t, x) u^T(t, x, j), \\ & \quad l = 1 \dots n \end{aligned} \quad (13)$$

then:

$$\begin{aligned} & \sum_{i=0}^N e_{\Delta}^T(t, i) \left(\hat{G}(t, x, i) - G^*(t, x) \right) u(t, x, i) \leq \gamma_0^2, \\ & \quad 0 \leq t \leq T_k \end{aligned} \quad (14)$$

where $F_l(i-j)$ is a positive-definite discrete matrix kernel as well.

Note that the training laws (9), (11), and (13) are updated through accumulations of the tracking histories at a specific time t (a sample period in application). They are different from the training law used by adaptive NNs where the weights are updated through integration of the state along the time axis t .

The last two terms in (8) are the robust control terms. The first one for the modelling error $\varepsilon_1(t, x)$ can be written as:

$$\begin{aligned} & - \sum_{i=0}^N e_{\Delta}^T(t, i) (\varepsilon_1(t, x) + K_1 e(t, i)) = \\ & - \sum_{i=0}^N e_{\Delta}^T(t, i) (\varepsilon_1(t, x) + K_1 e_{\Delta}(t, i) + K_1 \phi(t, i)) \quad (\text{using (4)}) \\ & \leq \sum_{i=0}^N \left(-e_{\Delta}^T(t, i) K_1 e_{\Delta}(t, i) + |e_{\Delta}(t, i)|^T \|\varepsilon_1\|_M - |e_{\Delta}(t, i)|^T K_1 \varepsilon_f \right) \end{aligned}$$

where $|e_{\Delta}(t, i)| = [|e_{\Delta 1}(t, i)|, \dots, |e_{\Delta n}(t, i)|]^T$, $\|\varepsilon_1\|_M = \|\varepsilon_1\|_{\infty} [1 \dots 1]^T$.

If we let the feedback gain matrix $K_1 = \text{diag}(K_{11}, \dots, K_{1n})$ and its multiplication with the width of the deadzone satisfies $K_{1i} \varepsilon_{fi} > \|\varepsilon_1\|_{\infty}$, $i = 1 \dots n$, then:

$$\begin{aligned} & - \sum_{i=0}^N e_{\Delta}^T(t, i) (\varepsilon_1(t, x) + K_1 e(t, i)) \\ & \leq - \sum_{i=0}^N e_{\Delta}^T(t, i) K_1 e_{\Delta}(t, i) \leq 0 \end{aligned} \quad (15)$$

For the last term, at first, we rewrite the control law in equation (6) to be $u(t, x, i) = \hat{G}(t, x, i)^+ (u' + K_2 e(t, i))$, where K_2 is a positive scalar and u' is measurable. Remember $E = \varepsilon_2(t, x) \hat{G}(t, x, i)^+$ as defined in Assumption 2; then:

$$\begin{aligned} & - \sum_{i=0}^N e_{\Delta}^T(t, i) (\varepsilon_2(t, x) u(t, x, i) + K_2 e(t, i)) = \\ & - \sum_{i=0}^N e_{\Delta}^T(t, i) (E u' + E K_2 e(t, i) + K_2 e(t, i)) \\ & = - \sum_{i=0}^N (e_{\Delta}^T(t, i) E K_2 e_{\Delta}(t, i) + e_{\Delta}^T(t, i) K_2 e_{\Delta}(t, i)) \\ & - \sum_{i=0}^N e_{\Delta}^T(t, i) (E u' + E K_2 \phi(t, i) + K_2 \phi(t, i)) \quad (\text{using (4)}) \\ & \leq - \sum_{i=0}^N K_2 (1 - \|E\|_M) \|e_{\Delta}\|^2 \\ & + \sum_{i=0}^N \left(\|e_{\Delta}(t, i)\| \|E\|_M \|u'\| + K_2 \|e_{\Delta}(t, i)\| \|E\|_M \|\phi(t, i)\|_{\infty} \right. \\ & \quad \left. - K_2 |e_{\Delta}(t, i)|^T \varepsilon_f \right) \\ & \leq \sum_{i=0}^N \|e_{\Delta}(t, i)\| (\|E\|_M \|u'\| + K_2 \|E\|_M |\varepsilon_f|_M - K_2 |\varepsilon_f|_m) \\ & \quad (\text{using (4) and Assumption 2}) \\ & = \sum_{i=0}^N \|e_{\Delta}(t, i)\| (-K_2 (|\varepsilon_f|_m - \|E\|_M |\varepsilon_f|_M) + \|E\|_M \|u'\|) \end{aligned}$$

From Assumption 2, if we let the control gain satisfy:

$$K_2 = \frac{\|E\|_M}{|\varepsilon_f|_m - \|E\|_M |\varepsilon_f|_M} \|u'\| > 0 \quad (16)$$

then:

$$- \sum_{i=0}^N e_{\Delta}^T(t, i) (\varepsilon_2(t, x) u(t, i) + K_2 e(t, i)) \leq 0 \quad (17)$$

Substituting equations (10), (12), (14), (15), and (17) into equation (8) yields:

$$\sum_{i=0}^N e_{\Delta}^T(t, i) \dot{e}(t, i) \leq \gamma_0^2, \quad \forall N, \forall 0 \leq t \leq T_k$$

From Lemma 1, we get the final result: If the system tracks the segment of the desired trajectory repetitively, the tracking error of every point along the trajectory will be less than a specified value described by the deadzone, that is, $\lim_{i \rightarrow \infty} |e_j(t, i)| \leq \varepsilon_{fj}$, $\forall 0 \leq t \leq T_k$.

For the control law (6), if the equation $\hat{G}(t, x, i)x = y$ has no solution, the system may face an uncontrollable

problem. The control law (6) provides a least-square error solution with the least norm.

In the next section, we use the above result to develop a segmented training scheme that guarantees uniform boundedness during training.

4. Uniformly Bounded Tracking with Segmented Training

A system to be controlled is usually not allowed to deviate from the desired trajectory too much. If we have less knowledge about the system, the first several times' training may result in a large tracking error such that it would leave the region Ω for approximation. The aforementioned repetitive training NNs are suitable to constrain the training only taken place within an admissible region because the proposed local NNs along the trajectory are independent of each other.

Figure 3. Definition of the permitted region Ω_M and the desired region Ω_D .

At first, we define the following regions around the desired trajectory as shown in Fig. 3:

Permitted region Ω_M : This region can be defined by the maximum allowed training error ε_M as $\Omega_M = \{x : \|e(t, i)\| \leq \varepsilon_M\}$. If the tracking error exceeds it, the training has to stop. The compact region Ω for approximation contains this region.

Desired region Ω_D : This region describes the desired tracking accuracy for segmented training as $\Omega_D = \{x : \|e(t, i)\| \leq \varepsilon_D\}$. When all points along a segment have been controlled into this region, the training can be extended to the next segment. This region should be bigger than the deadzone, that is, $\varepsilon_D > \|\varepsilon_f\|$, so that we can reach this accuracy through finite iterations.

An expected length of the segment $t_e = t_f$ at the beginning of the training, which means that we want the length of the segment to be the whole trajectory initially. The segmented training can be described by the following steps:

1. The initial state is controlled within the deadzone $|e_j(0, i)| < \varepsilon_{fj}, j = 1 \dots n, for \forall i$, as the beginning of a segment tracking.
2. Update the NN weights with (9), (11), and (13) and control the system (1) based on (6) until the system has finished the segment tracking with $t = t_e$ or the control error reaches the boundary of Ω_M at a time $t = T_k$; then $t_e = t$ and $x_d(t), 0 \leq t \leq t_e$, forms a new segment.
3. Go to step 1 until all points along the segment $x_d(t), 0 \leq t \leq t_e$, are controlled into the region Ω_D .
4. If $t_e < t_f$, then extend the expected segment with $t_e = t_f$ and go to step 1; else the whole trajectory tracking has been controlled into the desired region Ω_D .

We can continue the whole trajectory training repetitively for achieving higher accuracy, and the trained networks can be employed as a compensation of the nonlinear uncertainties in further use.

Now, we are going to explain this segmented training scheme. In the first time tracking, the unknown system is controlled by the NNs with initial weights. Under the control of these poorly trained NNs, usually the tracking error will increase along with the moving of the desired trajectory. If it does not exceed the permitted region Ω_M , we continue this tracking and train the corresponding networks by the updating laws presented in the last section. Suppose that, at time T_1 , the tracking error has reached the boundary of Ω_M as shown in Fig. 3. Then, from the result of Section 3, if we repeat the training continuously only from time 0 to T_1 , we have:

$$\lim_{i \rightarrow \infty} |e_j(t, i)| \leq \varepsilon_{fj}, \forall 0 \leq t \leq T_1$$

This implies that, for a given Ω_D with $\varepsilon_D > \|\varepsilon_f\|$, there exists a positive constant M_1 , if the number N of repeated training is greater than M_1 , the error of every point of this segment can be controlled into the region Ω_D . Then the segment can be extended until the tracking error reaching the boundary of Ω_M again at time T_2 as shown in Fig. 3.

Because the system works on a compact region Ω , if the control input $u(t, x, i)$ in equation (1) is bounded, then $\|\dot{x}\|_{\max}$ is bounded. This means that it will take at least a time $T \neq 0$ for the states of the system running out of Ω_M from Ω_D . Therefore, we know that $T_2 \geq T_1 + T \geq 2T$. Similarly, after the N^{th} training ($N \geq M_1 + M_2$), the errors at every point along the segment $x_d(t), 0 \leq t \leq T_2$, can be controlled into the region Ω_D for further iterations. Suppose that there is an integer k satisfying $k \geq t_f/T$; then,

in general, there exists an integer $M = \sum_{i=1}^k M_i$ such that the tracking error of the whole trajectory will be controlled into the region Ω_D if the sum of the repeated times is greater than M . Furthermore, if we repeat the training infinitely, the whole trajectory tracking error can be controlled into the deadzone, that is, $\lim_{i \rightarrow \infty} e_\Delta(t, i) = 0, for 0 \leq t \leq t_f$.

The above result was obtained under the assumption of bounded control input $u(t, x, i)$. This is not true when using the updating laws (9), (11), and (13). However, the bounded input can be guaranteed if we have rough knowledge about the range of the parameters and use bounded identification instead.

For the vector of the desired velocity, suppose that we know the range of every component $|\dot{x}_{dh}|_{\min} < \dot{x}_{dh}(t) < |\dot{x}_{dh}|_{\max}, h = 1 \dots n$; then the updating law (9) can be modified by the following bounded identification law:

Please see Equation (18) top of next page.

where $F_h(i-l)$ is a positive-definite discrete kernel.

The inequality (10) can be met as well even when the estimation reaches its boundary. Similarly, for estimating the unknown $f(t, x)$ and $G(t, x)$, the updating laws (11) and (13) can be replaced by the following bounded identification laws:

$$\hat{x}_{dh}(t, i) = \begin{cases} |\dot{x}_{dh}|_{\max} & \hat{x}_{dh} \geq |\dot{x}_{dh}|_{\max} \text{ and } e_{\Delta h}(t, i) > 0 \\ |\dot{x}_{dh}|_{\min} & \hat{x}_{dh} \leq |\dot{x}_{dh}|_{\min} \text{ and } e_{\Delta h}(t, i) < 0 \\ \sum_{l=0}^i F_h(i-l)e_{\Delta h}(t, l) & \text{else} \end{cases} ; h = 1 \cdots n \quad (18)$$

$$\hat{W}_{fjk}(t, i) = \begin{cases} |W_{fjk}|_{\max} & \hat{W}_{fjk}(t, i) \geq |W_{fjk}|_{\max} \text{ and } e_{\Delta j}(t, i)\varphi_k(t, x) < 0 \\ |W_{fjk}|_{\min} & \hat{W}_{fjk}(t, i) \leq |W_{fjk}|_{\min} \text{ and } e_{\Delta j}(t, i)\varphi_k(t, x) > 0 \\ -\sum_{l=0}^i F_{fjk}(i-l)e_{\Delta j}(t, l)\varphi_k(t, x) & \text{else} \end{cases} ; j = 1 \cdots n, k = 1 \cdots L \quad (19)$$

$$\hat{W}_{hjk}(t, i) = \begin{cases} |W_{hjk}|_{\max} & \hat{W}_{hjk}(t, i) \geq |W_{hjk}|_{\max} \text{ and } e_{\Delta h}(t, i)\varphi_j(t, x)u_k(t, i) < 0 & h = 1 \cdots n, \\ |W_{hjk}|_{\min} & \hat{W}_{hjk}(t, i) \leq |W_{hjk}|_{\min} \text{ and } e_{\Delta h}(t, i)\varphi_j(t, x)u_k(t, i) > 0 & j = 1 \cdots L, \\ -\sum_{l=0}^i F_{hjk}(i-l)e_{\Delta h}(t, i)\varphi_j(t, x)u_k(t, i) & \text{else} & k = 1 \cdots m \end{cases} \quad (20)$$

With the bounded identification laws (18), (19), and (20) instead of (9), (11), and (13), the stability analysis in Section 3 is kept the same. The bounded learning also exhibits several advantages in control. First, because we consider the control problem on a compact region Ω only, if weights of the NNs are bounded then the control input $u(t, x, i)$ is bounded. This is the assumption for the segmented training. Second, in this case, u' is also bounded; then the linear feedback gain K_2 in (16) can be a constant. This makes the control law (6) clearly composed of an NN compensation and a linear feedback with a constant gain. In addition, in Assumption 2 we have assumed that $\hat{G}(t, x, i)x = y$ has a solution. If we have prior knowledge that the equation $G(t, x)x = y$ is solvable on the permitted region Ω_M around the desired trajectory $x_d(t)$, then, using the bounded identification law (20), we can restrict the parameters of the identification within an adequate region so that the equation $\hat{G}(t, x, i)x = y$ is solvable, and then the control singular problem[19] can be avoided. Finally, just as the sigmoid function is widely used in neural networks to saturate outputs, the above algorithms also saturate the weights with bounded identification.

5. Example: Visual Servoing for Robot Motion Imitation

A camera can be used as a motion generator for robot manipulator in a “teaching by showing” way. For point-to-point positioning, [21] presented a neural network-based transformation from visual cues to manipulator control inputs. A well-trained neural network is able to generate expected control signals to reach the demonstrated position by means of image features. For continuous trajectory tracking, a trajectory imitation scheme by means of visual servoing was proposed [19]. First, a human teacher grasps a tool or simply an object and carries out a demonstration as shown in Fig. 4. At the same time, a static camera

(16mm) records the trajectories of some selected features of the object in the image plane as shown in Fig. 5, which are trajectories of four corner points of the cube within five seconds. The arrows in Fig. 5 represent direction of the motion. Then, a manipulator grasps the object to perform tracking repetitively. A perfect replay of the demonstrated trajectory can be achieved after a sufficient number of repetition. This requires learning from repetitive tracking for accurate visual servoing.

Figure 4. Teacher’s demonstration and feature points.

Figure 5. Demonstrated trajectories of the features in image plane.

The key to the design of a visual servoing controller is how to obtain the image Jacobian $J(x)$, called interaction matrix in visual servoing approach. Due to the inherent nonlinearities of a camera-robot system, the Jacobian matrix involves inaccurate camera parameters and unknown depths of image features. Usually, it requires a priori knowledge about the camera-robot model through offline calibration. This kind of model-based calibration may be sensitive to unmodelled factors. Instead of analytic model-based calibration, neural networks, such as PDP neural networks [21] and neural gas [22], have been used as the general mapping to approximate the relationship between image data and robot motion. They are model independent but still belong to the offline modelling methodology, where the networks training and control work in two separated phases. As an offline modeling method, the training may be tedious and inflexible for frequently changed applications. Thus adaptive visual servoing was proposed to improve flexibility and adaptability from a task-oriented viewpoint, where online identification and control com-

bins together to avoid offline calibration. Some work in the literature [23–25] has supposed a known system structure, that is, known camera-robotic kinematics and ideal pinhole camera model, and further represented the model with unknown linearized parameters, such as an ARMAX mode for applying the linear adaptive technique. In order to control vision guided robot with less prior knowledge, model-independent visual servoing schemes were proposed [26–28], where the camera-robot model was approximated by a linearized affine model and the Jacobian matrix was identified by using linear adaptive technique. When applying time-domain linear adaptive technique, the uncertainties must be constant or slow time-varying. As a result, all of them have some assumptions such as low speed or low acceleration motion. For the robot motion imitation that often involves a nonperiod demonstrated trajectory, the end of a last tracking has less similarity to the start of a new tracking. Time-based experience accumulation cannot accumulate knowledge efficiently. In this section, the proposed repetitive segmented NN control is used for robot motion imitation and compares with the dynamic Bryden’s method for visual servoing [27].

Suppose that the optical equation of the selected feature points is:

$$\dot{x} = J(x)\dot{\theta} \quad (21)$$

where the general coordinates $\theta \in R^m$, the features $x \in R^n$, and Jacobian matrix $J(\theta) \in R^{n \times m}$.

The dynamic Bryden’s method was proposed for vision-based moving-target tracking without calibration, where the Jacobian was estimated using Bryden estimation:

$$\begin{aligned} \hat{J}(k\tau) &= \hat{J}((k-1)\tau) + \frac{(\Delta e(k\tau) - \hat{J}(k\tau)\Delta\theta(k\tau) + \dot{x}_d(k\tau)\tau)\Delta\theta^T(k\tau)}{\Delta\theta^T(k\tau)\Delta\theta(k\tau) + s} \\ \theta((k+1)\tau) &= \theta(k\tau) - \hat{J}^+(k\tau)(Ke(k\tau)) - \dot{x}_d(k\tau)\tau \end{aligned} \quad (22)$$

where $\tau = 20\text{ms}$ is the sample period and two parameters s and K are introduced for practical concerns. Obviously, it is a time-domain adaptive law for a secant Jacobian estimation. For a slow time-varying Jacobian, the estimation can work well to adapt its change. However, in terms of multiple resetting and then tracking for nonperiodic trajectory training, the method is not able to improve its performance from repetitive practice. The reason is that the method cannot deal with fast time-varying or discontinuous Jacobian. The dashed line of Fig. 6 shows the RMS errors of 25 times’ tracking by the control of (22). For each tracking, the initial Jacobian is set to the Jacobian at the end of the last tracking. The parameters are selected as $s = 0.01$ and $K = 13$. In order to avoid singularity of the control law, the bounded identification that is similar to (20) is adopted. As a result, the $\hat{J}(k\tau)$ in (22) can be adjusted only within a known bound. This bound is with a maximum 50% deviation around the desired trajectory:

$$\begin{aligned} \left| \hat{J}_{ij}^{\ell}(x_d(t)) \right|_{\min} &= J_{ij}^{\ell}(x_d(t)) - 0.5 \left| J_{ij}^{\ell}(x_d(t)) \right| \\ \text{and } \left| \hat{J}_{ij}^{\ell}(x_d(t)) \right|_{\max} &= J_{ij}^{\ell}(x_d(t)) + 0.5 \left| J_{ij}^{\ell}(x_d(t)) \right| \end{aligned} \quad (23)$$

Figure 6. The RMS errors of ILC NN control and dynamic Bryden’s method.

From Fig. 6, we see that the RMS errors declined only in the first five trackings, and then the declination stops even with about 9mm errors. Further repetition makes no contribution to improving the tracking performance.

Now, we are going to demonstrate how the proposed iterative neural network control can help uncalibrated visual servoing and gain knowledge from repetitive trials. The system (21) is in the form of equation (1) with $f(t, x) = 0$ and with an unknown Jacobian matrix $G(t, x) = J(x)$. As mentioned in Section 2, a lot of different basis functions can be selected to do approximation. For example, in the region Ω_M around a point on the desired trajectory $x_d(t)$, we can express the Jacobian matrix in a form of a Taylor series:

$$J_i^T(t, x) = J_i^T(x_d(t)) + (x_d(t) - x)^T J_i'(x_d(t)) + \dots$$

where $J_i^T(t, x)$ is the i^{th} row of the Jacobian matrix and $J_i^{\ell}(x_d(t)) = \left(\frac{\partial J_i^T(t, x)}{\partial x} \right)_{x_d(t)}$.

If we approximate it with linear terms only, then:

$$J_i^T(t, x) = \left[1 \ (x_d(t) - x)^T \right] \begin{bmatrix} J_i^T(x_d(t)) \\ J_i'(x_d(t)) \end{bmatrix} = \varphi^T(t, x)W_i(t)$$

where basis function vector in equation (2) becomes:

$$\varphi(t, x) = \left[1 \ (x_d(t) - x)^T \right]^T$$

and the unknown weight is:

$$W_i(t) = \begin{bmatrix} J_i^T(x_d(t)) \\ J_i'(x_d(t)) \end{bmatrix}$$

So, the training of the weights with (20) is to identify the unknown $J_i^{\ell}(x_d(t))$ and $J_i'(x_d(t))$ along the desired trajectory. When $f(t, x) = 0$, the control law (6) becomes:

$$u(t, x, i) = \hat{J}(t, x, i)^+ \left(\hat{x}_d(t, i) + K_2 e(t, i) \right) \quad (24)$$

Table 1
Training Table of NNs

$k\tau$	$\hat{x}_d(k\tau)$	$\hat{J}_i(x_d(k\tau))$	$\hat{J}_i'(x_d(k\tau))$
...
3τ	$\hat{x}_d(3\tau)$	$\hat{J}_i(x_d(3\tau))$	$\hat{J}_i'(x_d(3\tau))$
2τ	$\hat{x}_d(2\tau)$	$\hat{J}_i(x_d(2\tau))$	$\hat{J}_i'(x_d(2\tau))$
τ	$\hat{x}_d(\tau)$	$\hat{J}_i(x_d(\tau))$	$\hat{J}_i'(x_d(\tau))$

The neural networks can be implemented through a table as shown in Table 1, where each row of the table corresponds to a point along the desired trajectory with an increment of a sample period τ . In this simulation, the increment τ is 20ms. For training, the lower part of the table is updated repetitively when the corresponding tracking errors are within the permitted region Ω_M . After every point corresponding to this part of the table is controlled within the desired region Ω_D , the training can be extended to the upper part of the table. When all rows of the table have been trained, we have achieved our control goal.

For imitating the demonstrated trajectory in Fig. 5, the feedback and the learning gains of the control law are selected to be:

$$K_2 = 3 \text{ in equation (24); } F(i - j) = 10 \text{ in equation (9); } F_{hjk}(i - l) = 2 \text{ in equation (20).}$$

In order to avoid singularity of the proposed control law, the bounded identification law (20) is adopted for estimating the input matrix $J(x(t))$. As treated in the simulation of the dynamic Broyden's method, it is applied to $\hat{J}(x_d(t), i)$ with a maximum $\pm 50\%$ deviation from the desired one, that is, (23).

Figure 7. The first control errors of feature point 1.

The first-order term $\hat{J}'_i(x_d(t), i)$ and the desired trajectory $\hat{x}_d(t, i)$ are updated by the laws of (13) and (9) without bounded limitation. The permitted region and the desired region are assumed to be $\Omega_M = \{x : \|e(t, i)\| \leq 3 \times 10^{-4}\}$ and $\Omega_D = \{x : \|e(t, i)\| \leq 3 \times 10^{-5}\}$, respectively. Finally, let the width of the deadzone in (4) be $\varepsilon_{fi} = 5 \times 10^{-7}$; then we can start the training.

Let the initial weights of the neural networks be $\hat{J}_{ij}(x_d(t), 0) = \left| \hat{J}_{ij}(x_d(t)) \right|_{\min}$, $\hat{J}'(x_d(t), 0) = 0$, and $\hat{x}_d(t, 0) = 0$. Fig. 7 shows the control errors of feature point 1 in the Cartesian space at the first iteration. The tracking is stopped at 1.3 (sec) with a maximum deviation of $\left[0.0178 \ -0.0363 \ 0.0191 \right]$ (m). Then the segment of $t = 0 \sim 1.3$ (sec) is trained repeatedly. After another two iterations, the control error of every point along this segment is controlled within an error band of $|E(t, 3)| < 3.1(mm)$ in the Cartesian space. Now the segment is extended to the rest of the trajectory. In order to speed up learning and reduce the segmented number, we can adopt the concept of generalization in CMAC for initiating the weights of the new segment, where a group of memory cells are updated that are close to a selected memory cell [29]. We let the initial value of the estimated velocity for the new segment be equal to an average over a region at the end of the trained segment, for example, an average between $\hat{x}_d(t = 1.2)$ and $\hat{x}_d(t = 1.3)$. It reduces the segmented number effectively. In our case, further segmentation is not needed, as shown in Fig. 8. The extension of the segment leads to a big increase in the RMS error in Fig. 6. After the 25th training, the control errors are depicted in Fig. 9 with a maximum tracking error of 0.5(mm) for

every point along the trajectory.

Figure 8. The control errors of beginning a new segment training.

Figure 9. The control errors of feature point 1 after 25 iterations.

The proposed control law is derived for a continuous system only. However, when we use it to design a visual servoing controller, compared with the time-constant of a mechanical system, the video sampling rate is usually too slow to guarantee high-speed tracking. So, the control performance may be greatly affected by the problem of the so-called one-step delay caused by discretization. Therefore, in the implementation of the above control law, we made 9) for alleviating the effect of the one step delay:

$$\hat{x}'_d(k\tau, i) = \hat{x}_d((k+1)\tau, i-1) + F(0)e_\Delta(k\tau, i)$$

and the control law (24) becomes:

$$u(k\tau, i) = \hat{J}(k\tau, x, i)^+ \left(\hat{x}'_d(k\tau, i) + K_2 e(k\tau, i) \right)$$

This means that we use the history information of one step ahead (the $(k+1)^{th}$ sample period) to generate the control input for the period $k\tau$. The simulation results show that this modification improves the tracking performances at the beginning of a new segment. Otherwise, a big control error may tend to be occurred near the switching point of segments.

The RMS errors for 25 repetitive tracking are shown in Fig. 6, which clearly illustrates that the proposed model-free visual servoing controller can effectively improve tracking performance based on the previous experience. The accuracy can reach 10 times higher than the time-domain adaptive control of the Broyden's method in this simulation.

6. Conclusion

We have proposed a local network model for nonlinear system trajectory tracking with uncertainties that are invariant over iterations. In contrast to the adaptive neural network control, our training algorithm is derived from the viewpoint of iterative learning control. Each local network for a specific time instant is updated ~~alone iteration index~~ based on previous tracking history at the same time instant. This makes the local networks time independent and yields no cross-talk between different segments. The training law can be described by a history accumulation in a form of positive-definite discrete matrix kernel. It guarantees the stability of the tracking system through sufficient repetitions. Then a segmented training strategy with the property of bounded training is presented, which is necessary for a learning-based control system to keep

a system within its permitted region for neural networks approximation and, in practice for a robot system, to avoid collision in a complex environment. An iterative learning controller for visual servoing is further designed according to the proposed method and is shown to be effective for vision-guided trajectory imitation with an uncalibrated camera.

Appendix

Proof of Lemma 1:

We define an energy function of the i^{th} learning as $V_i(t) = \frac{1}{2}e_{\Delta}^T(t, i)e_{\Delta}(t, i)$. It is differentiable and $\dot{V}_i(t) = e_{\Delta}^T(t, i)\dot{e}(t, i)$. If the weak initial resetting condition is satisfied $|e_j(0, i)| < \varepsilon_{fj}, j = 1 \cdots n$, for $\forall i$, then $V_i(0) = 0$ and we have:

$$V_i(t) = \int_0^t \dot{V}_i(t)dt = \int_0^t e_{\Delta}^T(t, i)\dot{e}(t, i)dt, \text{ for } 0 \leq t \leq T_k$$

Then we accumulate the total energy for any instant t , $0 \leq t \leq T_k$, from the 1st to the N^{th} training:

$$\sum_{i=0}^N V_i(t) = \sum_{i=0}^N \left[\int_0^t e_{\Delta}^T(t, i)\dot{e}(t, i)dt \right] = \int_0^t \sum_{i=0}^N e_{\Delta}^T(t, i)\dot{e}(t, i)dt$$

If $\sum_{i=0}^N e_{\Delta}^T(t, i)\dot{e}(t, i) \leq \gamma_0^2, \forall N, \forall 0 \leq t \leq T_k$, where γ_0^2 is any bounded positive scalar, then:

$$\sum_{i=0}^N V_i(t) \leq \int_0^t \gamma_0^2 dt \leq \gamma_0^2 T_k, \text{ where } 0 \leq t \leq T_k$$

Because the time interval of a segment is finite, $\gamma^2 = \gamma_0^2 T_k$ is upper bounded, too. Thus $\sum_{i=0}^N V_i(t) \leq \gamma^2, \forall N$. When the repetitive time N goes to infinity, we get $\lim_{N \rightarrow \infty} \sum_{i=0}^N V_i(t) \leq \gamma^2$, which is upper bounded. As a result, $\lim_{i \rightarrow \infty} V_i(t) = 0, \forall 0 \leq t \leq T_k$, which means $\lim_{i \rightarrow \infty} e_{\Delta}(t, i) = 0, \text{ for } 0 \leq t \leq T_k$, namely, $\lim_{i \rightarrow \infty} |e_j(t, i)| \leq \varepsilon_{fj}$.

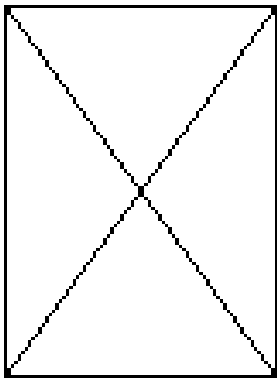
QED

References

- [1] Y.D. Landau, *Adaptive control: The model reference approach* (New York: Marcel Dekker, 1979).
- [2] S. Sastry & A. Isidori, Adaptive control of linearizable system, *IEEE Trans. Automatic Control*, 34(11), 1989, 1123–1131.
- [3] J.J.E. Slotine & S. Sastry, Tracking control of nonlinear system using sliding surface, with application to robot manipulators, *International Journal of Control*, 38(2), 1983, 465–492.
- [4] M. Corless & G. Leitmann, Continuous state feedback guaranteeing uniform ultimate boundedness for uncertain dynamic systems, *IEEE Trans. Automatic Control*, 26(5), 1981, 1139–1144.
- [5] R.M. Sanner & J.J.E. Slotine, Gaussian networks for direct adaptive control, *IEEE Trans. Neural Networks*, 3(6), 1992, 837–863.

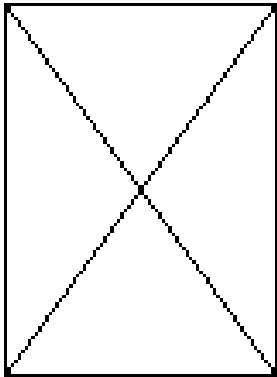
- [6] A. Yesildirek & F.L. Lewis, Feedback linearization using neural networks, *Automatica*, 31(11), 1995, 1659–1640.
- [7] S. Jagannathan, Discrete-time CMAC NN control of feedback linearizable nonlinear system under a persistence of excitation, *IEEE Trans. Neural Networks*, 10(1), 1999, 128–137.
- [8] Y.G. Leu, W.Y. Wang, & T.T. Lee, Robust adaptive fuzzy-neural controllers for uncertain nonlinear systems, *IEEE Trans. Robotics and Automation*, 15(5), 1999, 805–817.
- [9] S. Arimoto, S. Kawamura, & F. Miyazaki, Bettering operation of robots by learning, *Journal of Robotics System*, 1(2), 1984, 123–140.
- [10] S.S. Saab, On the P-type learning control, *IEEE Trans. Automatic Control*, 39(11), 1994, 2298–2302.
- [11] Y. Chen, C. Wen, Z. Gong, & M. Sun, An iterative learning controller with initial state learning, *IEEE Trans. Automatic Control*, 44(2), 1999, 371–376.
- [12] G. Heizinger, D. Fenwick, B. Paden, & F. Miyazaki, Stability of learning control with disturbances and uncertain initial conditions, *IEEE Trans. Automatic Control*, 37(1), 1992, 110–114.
- [13] D. Wang, Convergence and robustness of discrete time nonlinear systems with iterative learning control, *Automatica*, 34(11), 1998, 1445–1448.
- [14] Ping Jiang, Chen Huitang, Wang Yuejuan, Stability analysis for iterative learning control and its application to manipulators, *ACTA Automatica Sinica*, 23(4), 1997, 462–467.
- [15] J. Xu & Z. Qu, Robust iterative learning control for a class of nonlinear systems, *Automatica*, 34(8), 1998, 983–988.
- [16] M.M. Polycarpou, Stable adaptive neural control scheme for nonlinear systems, *IEEE Trans. Automatic Control*, 41(3), 1996, 447–451.
- [17] G.P. Liu, V. Kadiramanathan, & S.A. Billings, Variable neural networks for adaptive control of nonlinear systems, *IEEE Trans. Systems, Man, and Cybernetics, Part C*, 29(1), 1999, 34–43.
- [18] E.B. Kosmatopoulos, M.M. Polycarpou, M.A. Christodoulous, & P.A. Ioannou, High-order neural network structures for identification of dynamical systems, *IEEE Trans. Neural Networks*, 6(2), 1995, 422–431.
- [19] P. Jiang & R. Unbehauen, Robot visual servoing with iterative learning control, *IEEE Trans. on System, Man, and Cybernetics, Part A: System and Humans*, 32(2), 2002, 281–287.
- [20] J.J.E. Slotine & W. Li, *Applied nonlinear control* (Englewood Cliffs, NJ: Prentice-Hall, 1991).
- [21] H. Hashimoto, T. Kubota, M. Kudou, & F. Harashima, Self-organizing visual servo system based on neural networks, *IEEE Control Systems Magazine*, 12(2), 1992, 31–36.
- [22] J.A. Walter & K.J. Schulten, Implementation of self-organizing neural networks for visuo-motor control of an industrial robot, *IEEE Trans. on Neural Networks*, 4(1), 1993, 86–95.
- [23] J.T. Feddema & C.S.G. Lee, Adaptive image prediction and control for visual tracking with a hand-eye coordinated camera, *IEEE Trans. Systems, Man, and Cybernetics*, 20(5), 1990, 1172–1183.
- [24] N.P. Papanikolopoulos & P.K. Khosla, Adaptive robot visual tracking theory and experiments, *IEEE Trans. Automatic Control*, 38(3), 1993, 429–445.
- [25] N.P. Papanikolopoulos, B.J. Nelson, & P.K. Khosla, Six degree of freedom hand/eye visual tracking with uncertain parameters, *IEEE Trans. Robotics and Automation*, 11(5), 1995, 725–732.
- [26] K. Hosoda & M. Asada, Versatile visual servoing without knowledge of true Jacobian, *Proc. IEEE/RSJ Conf. on Intelligent Robots and Systems*, 1994, 186–193.
- [27] J.A. Piepmeier, G.V. McMurray, H. Lipkin, A dynamic quasi-Newton method for uncalibrated visual servoing, *Proc. Int. Conf. on Robotics and Automation*, 1999, 1595–1600.
- [28] J.A. Piepmeier, B.A. Gumpert, & H. Lipkin, Uncalibration eye-in-hand visual servoing, *Proc. IEEE Int. Conf. on Robotics and Automation*, Washington, DC, 2002, 568–573.
- [29] J.S. Albus, A new approach to manipulator control: The cerebellar model articulation controller (CMAC), *Trans. ASME, Journal of Dynamics, Systems, Measurement, and Control*, 97, 1975, 220–227.

Biographies



Ping Jiang received the B.Eng., M.Eng., and Ph.D. degrees in information and control engineering from Xi'an Jiaotong University, Xi'an, P.R. China, in 1985, 1988, and 1992, respectively. He was appointed as a Lecturer in the Department of Electrical Engineering, Tongji University, Shanghai, in 1992 and as an Associate Professor in 1994. Since 1997 he has been a Professor in the Department

of Information and Control Engineering, Tongji University. From 1998 to 2000 he worked as an Alexander von Humboldt Research Fellow in Lehrstuhl fuer Allgemeine und Theoretische Elektrotechnik, Universitaet Erlangen-Nuernberg, Germany. From 2002 to 2003 he was a Senior Research Fellow in Computing at Glasgow Caledonian University, U.K. In 2003 he accepted a lecturer appointment in cybernetics and virtual systems at the University of Bradford, U.K. His research interests include control theory and applications, learning control and neural networks, distributed control systems, robotics, robot vision, virtual organization, multi-agents, and product configuration management.



YangQuan Chen is an assistant professor in the Electrical and Computer Engineering Department and the Acting Director for CSOIS (Center for Self-Organizing and Intelligent Systems, www.csois.usu.edu) at Utah State University. He obtained his Ph.D. from Nanyang Technological University, Singapore, in 1998, an M.Sc. from Beijing Institute of Technology (BIT) in

1989, and a B.Sc. from the University of Science and Technology of Beijing (USTB) in 1985. Dr Chen has had 12 U.S. patents granted and two U.S. patent applications published. He published more than 160 academic papers and authored/co-authored more than 50 industrial reports. His recent books include *Solving Advanced Applied Mathematical Problems Using Matlab* (with Dingyu Xue, Tsinghua University Press), *System Simulation Techniques with Matlab/Simulink* (with Dingyu Xue, Tsinghua University Press), and *Iterative Learning Control: Convergence, Robustness and Applications* (with Changyun Wen, Springer-Verlag). His current research interests include autonomous navigation and intelligent control of a team of unmanned ground vehicles, machine vision for control and automation, distributed control systems (MAS-net: mobile actuator-sensor networks), fractional order control, interval computation, nanomechatronics and biomechatronics, and iterative/repetitive/adaptive learning control. Dr Chen has been an Associate Editor on the Conference Editorial Board of IEEE Control Systems Society since 2002. In 2003

he was a founding member of the ASME subcommittee of Fractional Dynamics. He is a senior member of IEEE, a member of ASME, and a member of ISIF (International Society for Information Fusion).