Article ID: 1000 - 8152(2004)03 - 0447 - 06

Iterative learning neural network control for robot learning from demonstration

JIANG Ping[1], LI Zi-yu[1], CHEN Yang-quan[2]
(1. Department of Information and Control Engineering, Tongji University, Shanghai 200092, China;
2. Center for Self-Organizing and Intelligent Systems, Department of Electrical and Computer Engineering, Utah State University, UT 84322 - 4160, USA)

Abstract: Learning from demonstration is an efficient way for transferring movement skill from a human teacher to a robot. Using a camera as a recorder of the demonstrated movement, a learning strategy is required to acquire knowledge about the nonlinearity and uncertainty of a robot-camera system through repetitive practice. The purpose of this paper is to design a neural network controller for vision-based movement imitation by repetitive tracking and to keep the maximum training deviation from a demonstrated trajectory in a permitted region. A distributed neural network structure along a demonstrated trajectory is proposed. The local networks for a segment of the trajectory are invariant or repetitive over repeated training and are independent of the other segments. As a result, a demonstrated trajectory can be decomposed into short segments and the training of the local neural networks can be done segment-wise progressively from the starting segment to the ending one. The accurate tracking of the whole demonstrated trajectory is thus accomplished in a step-by-step or segment-by-segment manner. It is used for trajectory imitation by demonstration with an unknown robot-camera model and shows that it is effective in ensuring uniform boundedness and efficient training.

Key works: iterative learning control; neural network control; visual servoing; imitation learning
CLC number: TP183 **Document code**: A

迭代学习神经网络控制在机器人示教学习中的应用

蒋 平[1], 李自育[1], 陈阳泉[2]
(1.同济大学 信息与控制工程系,上海 200092;
2. 尤他州立大学 电气与计算机工程系 自组织与智能系统中心,美国 尤他州 84322 - 4160)

摘要: 示教学习是机器人运动技能获取的一种高效手段.当采用摄像机作为示教轨迹记录部件时,示教学习涉及如何通过反复尝试获得未知机器人摄像机模型问题.本文力图针对非线性系统重复作业中的可重复不确定性学习,提出一个迭代学习神经网络控制方案,该控制器将保证系统最大跟踪误差维持在神经网络有效近似域内.为此提出了一个适合于重复作业应用的分布式神经网络结构.该神经网络由沿期望轨线分布的一系列局部神经网络构成,每一局部神经网络对对应期望轨迹点邻域进行近似并通过重复作业完成网络训练.由于所设计的局部神经网络相互独立,因此一个全程轨迹可以通过分段训练完成,由起始段到结束段,逐段实现期望轨迹的准确跟踪.该方法在具有未知机器人摄像机模型的轨迹示教模仿中得到验证,显示了它是一种高效的训练方法,同时具有一致的误差限界能力.

关键词: 迭代学习控制; 神经网络控制; 视觉伺服; 模仿学习

1 Introduction

In order to control a complex system with an unknown model, neural networks (NN) have been introduced to learn and reconstruct the unknown nonlinearities[1~3]. Most of them adjusted the network weights based on linear adaptive theory. The desired operation is supposed to be continual and, within an approximation region Ω, the convergence can be reached when the time t goes to infinity. For a non-periodic trajectory with a finite time interval, which is widely applied for practical systems such as cutting, painting, robot pick-and-place operation, multi-times resetting then tracking becomes a choice to keep the training persistently. However, the multi-times resetting violates the assumption of continual operating along the time horizon in usual adaptive control. The iterative learning control (ILC)[4~6] is a more special technology for this kind of application and can be considered as an efficient alternative.

In this paper, we present an NN iterative training scheme so that the control performance can be improved by using previous tracking experience. Unlike the adaptive NN's, every point (every servo period in application) along the desired trajectory is given an independent local NN such that any segment training does not cause any interference to the other segments. In common with other on-line training schemes, during the early stages of learning, the tracking errors may be quite large so that the states would leave the region for neural network approximation. Due to the independence of the proposed

Received date: 2001 - 03 - 30; Revised date: 2002 - 12 - 31.
Foundation item: supported by the National Natural Science Foundation of China (60175028).

local NN structure, a segmented training scheme is further presented to ensure that the tracking error during training does not exceed the admissible region of NN approximation. It always measures the maximum tracking error along the trajectory during the training process. If the error at any trajectory point exceeds a predefined bound then the trajectory is divided into two segments from this point on. After that, only the networks related to the first segment are trained repetitively so that the whole tracking of this segment reaches the desired precision. Then the training can be extended to the remained segments. So it works in a step by step or segment-by-segment manner, and finally the whole trajectory tracking can reach the desired precision. It makes the tracking error during the training not exceed the predefined bound for the NN approximation.

The application background of this research is for robot learning from demonstration with the help of a camera (or cameras). It includes two phases, teaching phase and training phase. At 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 describe a desired movement. At the training phase, a manipulator grasps the same object to dotracking repetitively. With the aid of the proposed NN iterative training visual servoing scheme, an exact replay of the demonstrated trajectory can be achieved gradually. It is a flexible way for robot trajectory programming.

2 Iterative learning neural network control

For a non-periodic desired trajectory $x_d(t)$ with a finite time interval $0 \leqslant t \leqslant t_f$, the objective of ILC is to force the state of the subsequent affine nonlinear system to follow the desired trajectory exactly after a series of iterative training:

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

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

The system (1) is described in terms of both continuous time t and discrete iterative number i. Although the time t is within a finite interval, the ILC can ensure that the entire trajectory converges to the desired one by infinite iterations.

First, we employ a linear parametric approximation model, which may be GRBF NN[1], CMAC NN[2], fuzzy NN[3], or even any mathematical approximation model with linear parameters, to express the nonlinear functions $f(t,x)$ and $G(t,x)$ as follows:

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

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

While the adaptive neural networks[1~3] distribute neurons with unknown but constant optimal weights over a region Ω containing the whole desired trajectory, as shown in Fig. 1, the above neural networks are further distributed along the t axis around the corresponding points (every sample period in application) of the desired trajectory as shown in Fig. 2. The optimal weights $W_f(t)$ and $W_l(t)$ may be time-varying and the networks can be used to approximate time-varying but repeated uncertainties. In adaptive NN control, the weight training is carried out along the time axis t but, in Fig. 2, the weight training is carried out along the iteration axis i by repetitive tracking or previous tracking experiences. In fact, each local NN tries to learn an approximation model in a neighborhood around a specific point of the desired trajectory.

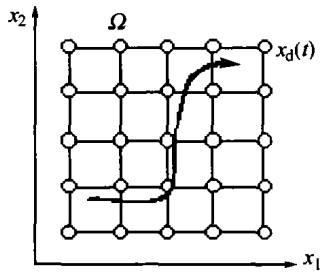

Fig. 1 Adaptive NN

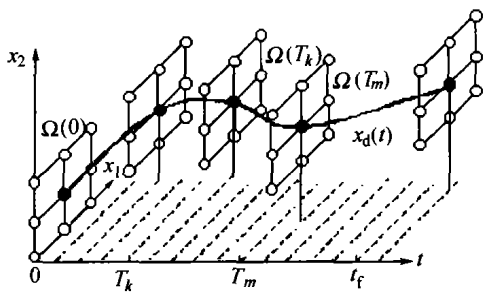

Fig. 2 Iterative learning NN

Therefore, we can expect that, at first, due to a series of time-varying local NN's, 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. In addition, although we distribute a series of local NN's over all trajectory points, it does not mean a great increase of the size of the networks because a local network with simpler structure can be chosen for approximation of a small region around a specific point. At last, an efficient "segmented training" can be accomplished by the ILC. Since each point along the desired trajectory has an independent local NN, the training of a

segment $x_d(t), 0 \leqslant t \leqslant T_k$, only adjusts the weights of the corresponding local networks but has not any interference to the other segments $x_d(t), T_k \leqslant t \leqslant T_f$. Here a segment is part of the desired trajectory described by a time interval within 0 to 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.

For a segment of $x_d(t), 0 \leqslant t \leqslant T_k$, first, let the i-th tracking error be $e(t,i) = [e_1(t,i), \cdots, e_n(t,i)]^T = x_d(t) - x(t,i)$. In order to deal with the modeling error and the initial resetting error, we define a modified error vector with deadzone as

$$\begin{cases} e_\Delta(t,i) = e(t,i) - \phi(t,i), \\ \phi(t,i) = \\ [\varepsilon_{f1}\mathrm{sat}(e_1(t,i)/\varepsilon_{f1}), \cdots, \varepsilon_{fn}\mathrm{sat}(e_n(t,i)/\varepsilon_{fn})]^T, \end{cases} \tag{3}$$

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

Furthermore, we define the following networks:

$$\begin{cases} \hat{f}(t,x) = \hat{W}_f(t,i)\varphi(t,x), \\ \hat{G}(t,x,i) = \begin{bmatrix} \varphi(t,x)^T\hat{W}_1(t,i) \\ \vdots \\ \varphi(t,x)^T\hat{W}_n(t,i) \end{bmatrix}, \end{cases} \tag{4}$$

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

We make the following assumption on the modeling errors.

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

We make the following assumption for control singularity avoidance.

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 and 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 $\| E \|_M < |\varepsilon_f|_m / |\varepsilon_f|_M \leqslant 1$, where $|\varepsilon_f|_m$ and $|\varepsilon_f|_M$ are the minimum width and the maximum width in the deadzone vector, respectively.

With the aid of the networks (4), a control law with least norm can be proposed as

$$u(t,x,i) = \hat{G}(t,x,i)^+ (\dot{\hat{x}}_d(t,i) - \hat{f}(t,x,i) + K_1 e(t,i) + K_2 e(t,i)), \tag{5}$$

where $\hat{G}(t,x,i)^+$ denotes the pseudo-inverse matrix of $\hat{G}(t,x,i)$, $\dot{\hat{x}}_d(t,i)$ is an estimated velocity of the desired trajectory. If $\dot{x}_d(t)$ is available to use, we should directly assign $\dot{x}_d(t)$ to $\dot{\hat{x}}_d(t,i)$. 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.

Theorem 1 If a weak initial resetting condition $|e_j(0,i)| < \varepsilon_{fj}, j = 1, \cdots, n$, for $\forall i$ is satisfied, under control of (5) with the following weight updating laws:

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

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

$$\hat{W}_l(t,i) = -\sum_{j=0}^{i} F_l(i-j)e_{\Delta l}(t,j)\varphi(t,x)u^T(t,x,j),$$

$$l = 1, \cdots, n,$$

where $F(i-j)$, $F_f(i-j)$, and $F_l(i-j)$ are any positive definite discrete matrix kernels, and the feedback gains satisfying:

$K_1 = \mathrm{diag}(K_{11}, \cdots, K_{1n})$, where $K_{1i}\varepsilon_{fi} > \| \varepsilon_1 \|_\infty$,

$$K_2 = \frac{\| E \|_M}{|\varepsilon_f|_m - \| E \|_M |\varepsilon_f|_M} \| u' \| > 0,$$

where $u' = \| \dot{\hat{x}}_d(t,i) - \hat{f}(t,x,i) + K_1 e(t,i) \|$, then the tracking error of every point along the segment $x_d(t), 0 \leqslant t \leqslant T_k$, will converge to the deadzone, i.e. $\lim_{i\to\infty} |e_j(t,i)| \leqslant \varepsilon_{fj}, \forall 0 \leqslant t \leqslant T_k$, when the system tracks the desired trajectory repetitively.

Proof See [7].

3 Segmented training neural network control

A system to be controlled is usually not allowed to deviate from the desired trajectory too much. If we have a 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 of approximation. So, we have to restrict this deviation during the training. The above training algorithm is helpful to do so because the NN's along the trajectory are independent of each other.

First, we define the following regions around the desired trajectory as shown in Fig.3.

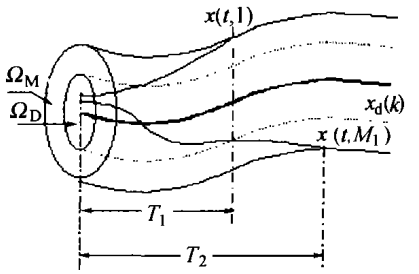

Fig. 3 Segmented training

Permitted region Ω_M This region can be defined by the maximum allowed error ε_M as $\Omega_M = \{x: \| e(t,i) \| \leqslant \varepsilon_M\}$ when NN's are trained. It means that the

training can be conducted continuously when the tracking error remains in it. If the tracking error exceeds it, the training has to stop. This region should belong to the compact region Ω for approximation.

Desired region Ω_D This region describes the desired tracking accuracy for segmented training. It can be defined as $\Omega_D = \{x: \| e(t,i) \| \leqslant \varepsilon_D\}$. When every point along the segment of the desired trajectory has been controlled into this region, the training can be extended to the next segment. This region should be greater than the deadzone, i.e. $\varepsilon_D > \| \varepsilon_f \|$, such that we can reach this goal through finite iterations.

We define an expected length of the segment $t_e = t_f$ at the beginning, which means that we wish the length of the segment is the whole trajectory initially. The segmented training process is described by the following steps:

1) Let the initial state be located within the deadzone $| e_j(0,i) | < \varepsilon_{fj}, j = 1,\cdots,n$, for $\forall i$, as the beginning of a segment tracking;

2) Control the system (1) based on Theorem 1 until the system has finished tracking the segment 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 $x_d(t), 0 \leqslant t \leqslant t_e$, forms a new segment;

3) Go to step 1) until all points along the segment $x_d(t), 0 \leqslant t \leqslant t_e$, are kept in the region Ω_D;

4) If $t_e < t_f$, then begin to extend the expected segment with $t_e = t_f$ and go to step 1), else the whole trajectory has been controlled into the desired region Ω_D.

The system can continue the whole trajectory tracking repetitively so that the whole trajectory converges to the deadzone and the trained networks can be employed as a compensation for the nonlinear uncertainties in further use. Now, we are going to explain this segmental training scheme. In the first tracking, the unknown system is controlled by the NN's with initial weights. Under the control of this kind of poorly trained NN's, usually the tracking error will increase along with 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 , the tracking error has reached the boundary of Ω_M as shown in Fig. 3. Then, from the Theorem 1, if we repeat the training continuously from time 0 to T_1 only, we have

$$\lim_{i\to\infty} | e_j(t,i) | \leqslant \varepsilon_{fj}, \quad \forall 0 \leqslant t \leqslant T_1.$$

This implies that, for any 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, then the error of every point of this segment will be always kept in the region Ω_D for further iterations. Then we can go ahead to the next segment until the tracking error has reached 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 state of the system moving out of Ω_M from Ω_D. Therefore, we know that $T_2 \geqslant T_1 + T \geqslant 2T$. From the same reason, after the N-th training ($N \geqslant M_1 + M_2$), the errors at every point along the segment $x_d(t)$, $0 \leqslant t \leqslant T_2$, will be controlled into the region Ω_D for further iterations. Suppose that there is an integer k satisfying $k \geqslant t_f/T$, then, in general, there exists an integer $M = \sum_{i=1}^{k} M_i$ such that the tracking error of the whole trajectory can be controlled into the region Ω_D if the sum of the repeated times is greater than M.

We obtained the above result under the assumption of bounded control input $u(t,x,i)$. This can be satisfied by a projection method if we have a rough knowledge about the range of the parameters. In this case, the weights are kept to be bounded but never change the convergent result.

4 Visual servoing for robot motion imitation

For the time being, programming a motion of a robot still relies on "hard-coding". The task has to be carefully analyzed and added to the robot program by a human, which is usually difficult or even impossible for a robot with high degrees-of-freedom such as a humanoid robot. A learning approach is a potential method to overcome the need of "hard-coding"[8]. In humans, a teacher demonstrate a movement, we are capable of repeating through learning.

In this section, we are going to use the proposed NN control scheme for robot trajectory imitation by visual servoing. For a manufacturing task or simply a pick-and-place operation, the imitating process can be described as: First, a teacher grasps a tool or simply an object, and does a demonstration as shown in Fig.4. At the same time, a static camera records the trajectories of some selected features of the object on the image plane as shown in Fig.5, which are trajectories of 4 corner points of the cube, $p_1(t), p_2(t), p_3(t)$ and $p_4(t)$, in Fig.4. These describe the desired trajectory of the object. Then, let the manipulator grasp the same object and do the training repetitively as shown in Fig. 6. With the aid of the proposed NN controller, a perfect replay (within the desired precision region Ω_D) of the demonstrated trajectory can be achieved ultimately. It is a task of a 3D-trajectory visual tracking with only a single camera.

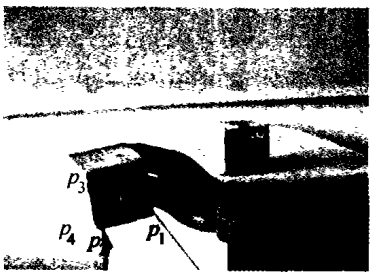

Fig. 4 Teacher's demonstration and feature points

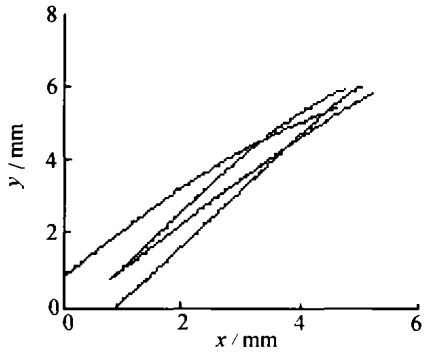

Fig. 5 Desired trajectories of the features

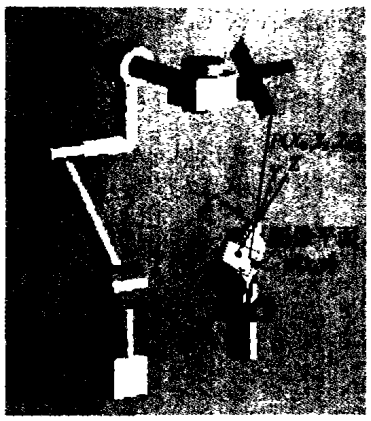

Fig. 6 Manipulator trajectory imitation

Suppose that the optical equation of the selected feature points is

$$\dot{x} = J(\theta)\dot{\theta}, \tag{6}$$

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

The system (6) 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. We select the basis function vector in equation (2) to be

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

When $f(t,x) = 0$, the control law becomes

$$u(t,x,i) = \hat{J}(t,x,i)^+ (\dot{\hat{x}}_d(t,i) + K_2 e(t,i)). \tag{7}$$

The training can be implemented through a table, where each row of the table corresponds to the weights for a point along the desired trajectory. The table is constructed with an increment of a sample period τ and is updated segment-by-segment. In this simulation, the increment τ is 20 ms. At first, the lower part of the table is updated repetitively, where 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. After all rows of the table have been trained, we have achieved our control goal as well.

Suppose that a static camera (16 mm) records the trajectories $x_d(t)$ of 4 feature points of an object on the image plane within 5 s as shown in Fig.5, which are demonstrated by a teacher. Then let a manipulator grasp the same object in front of the camera and try to track the trajectories repetitively such that the trajectories of the features coincide with the demonstrated trajectories on image plane. The feedback and the adaptive gains of the control law are selected to be

$$K_2 = 3, \quad F(i-j) = 10, \quad F_l(i-j) = 2.$$

In order to avoid singularity of the proposed control law, the bounded identification law is adopted for estimating the input matrix $J(x(t))$, namely, $\hat{J}(x_d(t), i)$ can only be adjusted within a known bound. This bound is with a maximum ± 50% deviation around the nominal one. The permitted region and the desired region are assumed to be $\Omega_M = \{x: \| e(t,i) \| \leqslant 3 \times 10^{-4}\}$, $\Omega_D = \{x: \| e(t,i) \| \leqslant 3 \times 10^{-5}\}$, respectively. Finally, let the width of the deadzone in (3) be $\varepsilon_{fi} = 5 \times 10^{-7}$, then we can begin the training.

Let the initial weights of the neural networks be

$$\hat{J}_{ij}(x_d(t),0) = | \hat{J}_{ij}(x_d(t)) |_{min},$$

$$\hat{J}'(x_d(t),0) = 0, \quad \hat{x}_d(t,0) = 0.$$

Fig. 7 shows the control errors of feature point 1 in the Cartesian space at the 1st iteration. The tracking is stopped at 1.3 s with a maximum deviation of [0.0178, −0.0363, 0.0191] m. Then we repeat training the segment of $t = 0 \sim 1.3$ s. After another 2 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, we continue our training for the rest of the trajectory. In order to speed up learning and reduce the segmented number, we let the initial value of the estimated velocity for the new segment be equal to an average over a region of the end of the trained segment, for example, an average between $\dot{\hat{x}}_d(t = 1.2)$ and $\dot{\hat{x}}_d(t = 1.3)$. It reduces the segmented number effectively. In our case, we need not further segment the rest of the trajectory. After the 15th training, the control errors are depicted in Fig. 8 with a maximum tracking error of 0.84 mm for every point along the trajectory.

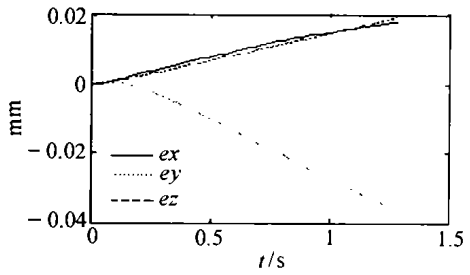

Fig. 7 The 1st control errors of the feature point 1

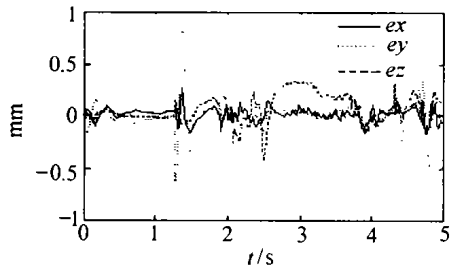

Fig. 8 Control errors of the feature point 1 after 15 iterations

5 Conclusions

We have proposed a local network structure for nonlinear system trajectory tracking with uncertainties that are invariant over iterations. The proposed method can be used for any approximation model with linear parameters. In contrast to the adaptive neural network control scheme, our training algorithm is derived from the viewpoint of ILC such that every local neural network for a particular point of the trajectory is independent of the others. This makes the repetitive segmented training become possible and a segmented training strategy to retain the training only in an effect region is presented. A visual servoing controller is further designed based on the proposed method and shows to be effective for vision-guided motion imitation without an exact model.

References:

[1] SANNER R M, SLOTINE J J E. Gaussian networks for direct adaptive control [J]. *IEEE Trans on Neural Networks*, 1992, 3(6): 837 - 863.

[2] JAGANNATHAN S. Discrete-time CMAC NN control of feedback linearizable nonlinear system under a persistence of excitation [J]. *IEEE Trans on Neural Networks*, 1999, 10(1): 128 - 137.

[3] LEU Y G, WANG W Y, LEE T T. Robust adaptive fuzzy-neural controllers for uncertain nonlinear systems [J]. *IEEE Trans on Robotics and Automation*, 1999, 15(5): 805 - 817.

[4] ARIMOTO S, KAWAMURA S, MIYAZAKI F. Bettering operation of robots by learning [J]. *J of Robotics System*, 1984, 1(2): 123 - 140.

[5] CHEN Y, WEN C, GONG Z, et al. An iterative learning controller with initial state learning [J]. *IEEE Trans on Automatic Control*, 1999, 44 (2): 371 - 376.

[6] XU J, QU Z. Robust iterative learning control for a class of nonlinear systems [J]. *Automatica*, 1998, 34(8): 983 - 988.

[7] JIANG P, UNBEHAUEN R. Iterative learning neural network control for nonlinear system trajectory tracking [J]. *Neurocomputing*, 2002, 48 (1/4): 141 - 153.

[8] SCHAAL S. Is imitation learning the route to humanoid robots [J]. *Trends in Cognitive Sciences*, 1999, 3(6): 233 - 242.

作者简介:

蒋 平 (1963 —), 男, 博士, 同济大学信息与控制工程系教授, 博士生导师, 主要研究领域为机器人控制与智能控制, E-mail: p.jiang@bradford.ac.uk;

李自育 (1963 —), 男, 同济大学职业技术教育学院工程师, 研究领域为机电一体化;

陈阳泉 (1966 —), 男, 博士, 美国尤他州州立大学电气与计算机工程系助理教授, 研究领域为控制理论及应用.