PERGAMON

Neural Networks 11 (1998) 1331–1344

Neural Networks

1998 Special Issue

# A tennis serve and upswing learning robot based on bi-directional theory

Hiroyuki Miyamoto[1,*], Mitsuo Kawato[2]

[1]*Japan Science and Technology Corporation, Kawato Dynamic Brain Project, Kyoto, Japan*
[2]*ATR Human Information Processing Research Laboratory, Kyoto, Japan*

Received and accepted 30 April 1998

**Abstract**

We experimented on task-level robot learning based on bi-directional theory. The via-point representation was used for 'learning by watching'. In our previous work, we had a robot learn kendama (a Japanese game) in order to demonstrate a single simple task. Our approach can be applied to a wide variety of motor behavior. However, some difficulties still remain. In this paper, we address two problems: (1) how to attain a final goal of complex movement when it consists of a sequence of subgoals, and (2) how to adapt to changes in behavior and the environment. To examine how to solve these problems, we propose two methods: (1) selecting the proper via-points for a control variable for each subgoal, and (2) re-estimating the relation between the via-points and the task during learning without conducting extra trials. We adopted a tennis serve and a pendulum upswing for our complicated tasks. © 1998 Elsevier Science Ltd. All rights reserved.

*Keywords:* Bi-directional theory; Learning by watching; Task-level learning

## 1. Introduction

It is time-consuming to make a robot perform a job requiring human skill by a conventional 'teaching playback' method. There are other difficulties, such as transforming a knack or an intuition of skilled movement into numerical values, or overcoming fluctuations and environmental uncertainty. Recently, learning from demonstration, in this case 'learning by watching' or 'teaching by showing', has been explored in the robotics field for higher level task learning (e.g. Aboaf et al., 1988; Kuniyoshi et al., 1991; Kang and Ikeuchi, 1993; Atkeson and Schaal, 1997).

The approach we are pursuing is based on the bi-directional theory (Kawato, 1992; Kawato et al., 1994). This theory provides a general computational framework for sensory-motor integration, and derives generic representations for a wide variety of motor behavior. One of the essential factors in this theory is the representation of behavior in a sparse via-point representation. Using a Forward Inverse Relaxation Model (FIRM), Wada and Kawato (1993, 1995) developed an algorithm to approximately extract the minimum number of via-points, $\mathcal{S} = \{P_1, P_2, \cdots, P_N\}$, from a given trajectory, $X_{\text{data}}$, with a level of error threshold $\delta$. Generally, (see Kawato et al., 1994 and Miyamoto et al., 1996 for more detail), our approach for learning by watching is as follows. First, a learner estimates the via-points $\mathcal{S}_{\text{teacher}} = \left\{X^1_{\text{via}}, X^2_{\text{via}}, \cdots, X^N_{\text{via}}\right\}$, as the internal representation of the teacher's movement plan by watching the demonstration. Next, the learner incorporates the internal representation $\mathcal{S}_{\text{learner}} = \left\{\tilde{X}^1_{\text{via}}, \tilde{X}^2_{\text{via}}, \cdots, \tilde{X}^N_{\text{via}}\right\}$. Then, the learner completes the task by controlling this internal representation. The actual trajectory executed by the robot arm is expressed as follows: $X_{\text{real}} \equiv \chi(\mathcal{S}_{\text{learner}})$, where $\chi$ includes the optimal trajectory calculation and feedback (and/or feedforward) controller. If the optimization principle adopted here does not differ much from the human's, and if the via-points representation is appropriate, the resulting realized movement of the robot arm is similar to the demonstrated movement of the human subject. The task is on the verge of successful execution at this point. Since the realized trajectory $X_{\text{real}}$ is smooth and parameterized by a small number of control variables $\mathcal{S}_{\text{learner}}$, various learning algorithms (in our case a Newton-like method) can be efficiently used for task-level learning.

In a previous study, we showed how this approach can be applied to learning the Japanese game of kendama by first extracting via-points from a human demonstration, and then transferring this knowledge to an anthropomorphic robot arm which must perform the same task (Kawato et al., 1994; Miyamoto et al., 1996). We chose the game of

* Corresponding author at Hikaridai 2-2, Seika-cho, Soraku-gun, Kyoto 619-02, Japan. Tel.: 0081 774 95 1233; Fax: 0081 774 95 3001; E-mail: miyamo@erato.atr.co.jp

kendama as a simple task example. The game's movement is relatively simple but it requires quick and dynamic motion. Movement in kendama consists of two parts, throwing a ball up on an attached string and catching it in a cup. In this case, it was possible to achieve the task by a simple representation as follows: the horizontal location of the ball falling at the height of the cup was chosen as the task variable and the via-points were the control variable which influenced the task variable. We adopted a Newton-like method for a learning scheme which uses a relation of the control and task variables, that is, a linear approximation (Jacobian). In the kendama task, there were few changes in the Jacobian while the learning was in progress, therefore we only used the Jacobian for first movement trajectory.

When we apply the above-mentioned learning to a more generalized movement, there are more problems to be considered. In this paper we will discuss two of them. First, the final goal of the task is not always attained in a single learning phase, even if a generalized sequential movement has one final goal. It may have a hierarchical structure, that is, there may be a number of subgoals (subtasks) which must be sequentially attained to achieve the final task goal. The learning process becomes more difficult when each subtask influences another, as in the tennis serve task (described in Section 2). Second, the relation of the control variable and the task variable (Jacobian) is usually inaccurate and changeable. We cannot avoid inaccuracy in the Jacobian estimation of a realistic situation because of fluctuations in robot motion and measurement error. Change and inaccuracy in the Jacobian make the learning process impossible or unstable. The Jacobian dramatically changes with the learning progress if the controlled object has strong nonlinear characteristics as in the pendulum upswing task (described in Section 3). In order to examine how to solve these problems, we had the robot perform a tennis serve and the pendulum upswing. Both tasks are more difficult than kendama.

In Section 2 we examine how to attain a final goal of sequential movement consisting of a sequence of subgoals. The final goal of the tennis serve is to strike a ball to put it in a goal. The learning procedure is divided into two phases with empirical knowledge. The robot must achieve these two temporally continuing subtasks in order: (1) to throw the ball so the ball hits the racket, and (2) to strike the ball so it falls into the goal. In the first implementation of our experiment, subtask 1 (throwing the ball up) was adversely affected by the modification of via-points for achieving subtask 2 (striking the ball). To avoid this problem, we carefully selected via-points for the control variable among all of the extracted via-points according to each learning phase. We intuitively selected via-points in Section 2.1. In Section 2.2, we describe a method for rationally selecting proper via-points by analysing the Jacobian.

In Section 3 we examine how to adapt to changes in the behavior of a controlled object or the environment. In the tennis serve, even if the robot arm follows exactly the same movement trajectory, the ball has a slightly irregular orbit after striking because of movement in the installation part of the racket and the rough surface of the ball. The ball's irregular behavior negatively affects the accurate and stable estimation of the Jacobian (Section 3.2). Learning the pendulum upswing is difficult compared with kendama because the pendulum has strong nonlinear characteristics. The pendulum shows many changes in behavior even if the change of the control input is minimal. When conducting learning only using the Jacobian estimated for the first movement trajectory, learning becomes unstable or impossible because the learning changes the movement trajectory. Thus the Jacobian changes dramatically during the learning progress (Section 3.3). In order to adapt to changes in the behavior and the environment, we propose a method for re-estimation of the Jacobian during learning, without conducting extra trials, in Section 3.1.

For simplicity, we used a simulated robot instead of a real robot, except in Section 2.1.

## 2. Learning sequential movements

The task we adopted in this section is to hit the ball so it falls into the goal. The goal was approximately 2 m from the player. Initially, the player holds the racket with the ball on a cup attached to the tennis racket grip. The player swings the racket to yank the ball over the racket. After a few hundred milliseconds, the ball is hit softly with the racket. If everything goes well, the ball falls into the goal.

We divided the tennis serve into two subtasks: (1) properly throwing the ball up to hit the ball with the racket, and (2) hitting the ball at a suitable angle. The two subtasks are performed consecutively. Since there is interaction between the first subtask and the second subtask, learning the tennis serve is more difficult than learning kendama (Miyamoto et al., 1996). In order to avoid interaction between subtasks, we carefully selected via-points for the control variable among all of the extracted via-points according to each learning phase. We intuitively selected via-points in the real robot experiment (Section 2.1). Following that, we made a better selection of via-points for a proper control variable (Section 2.2).

### 2.1. Tennis serve robot experiment

#### 2.1.1. Experimental setup

*Measurement of human demonstration.* Fig. 1 shows a schematic diagram of the tennis serve-learning robot. While a subject was performing a tennis serve, we measured the positions of the shoulder, elbow, wrist, and upper and lower sides of the tennis racket with the OPTOTRAK, a three-dimensional vision system which uses infrared light-emitting diode markers and three cameras. In this experimental setting, the precision of the position sensing was approximately a few millimeters and the sampling

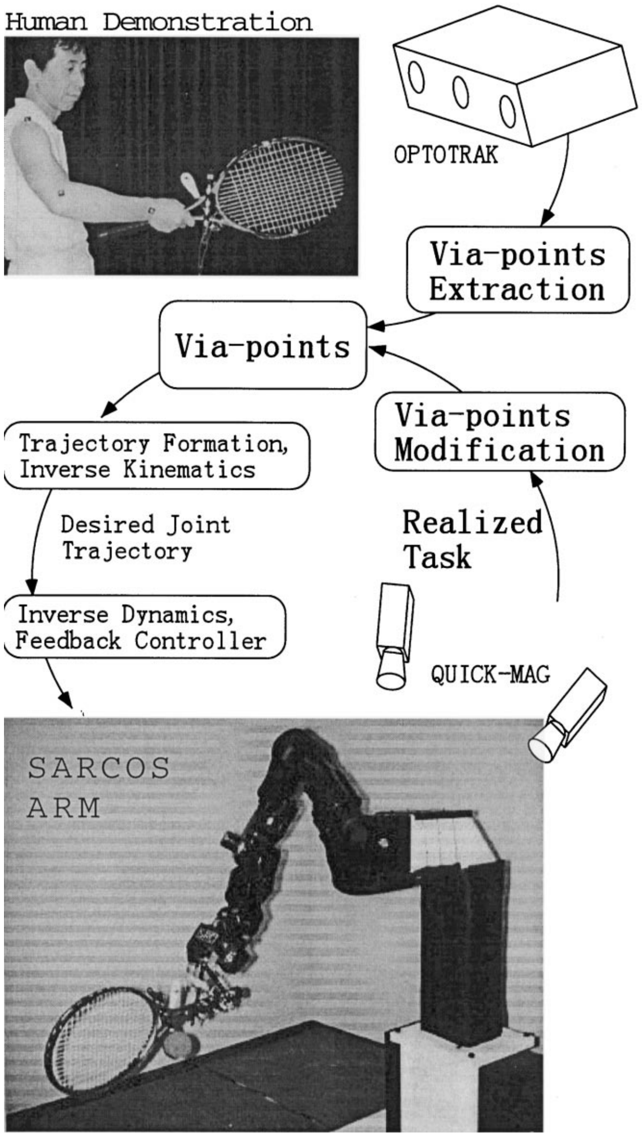


Fig. 1. Schematic diagram of tennis serve learning robot.

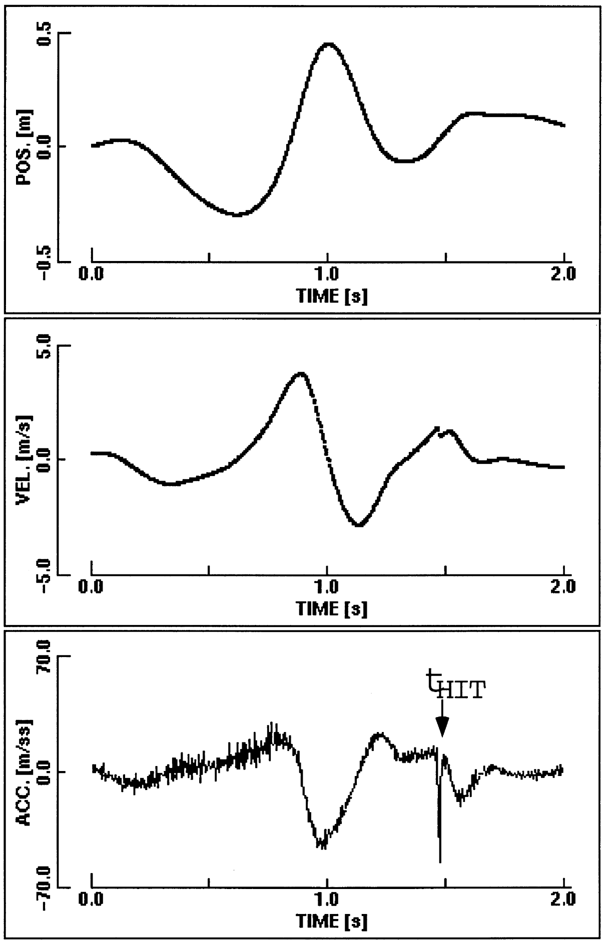


Fig. 2. Time course of the position, velocity, and acceleration at the racket center. Velocity and acceleration trajectories were calculated using numerical differentiation.

frequency was 250 Hz. We calculated the Cartesian racket position and orientations $X_{data}$ which is a six-dimensional vector from these position data. We determined the moment of impact $t_{HIT}$ of the ball on the racket by detecting the acceleration trajectory spike as shown in Fig. 2.

*Via-points extraction*. We extracted the Cartesian via-points $X_{via}$ from $X_{data}$ using the FIRM algorithm (Wada and Kawato, 1993, 1995). Fig. 3 illustrates how the algorithm for extracting the via-points works using an example of a tennis serve. Based on the FIRM model, the via-points are sequentially extracted as: (1) FIRM generates an optimal trajectory between the start point and the end point using the minimum principle for the approximated linear dynamics (point mass model) (left column). In this case, the generated trajectory is equivalent to the minimum-jerk trajectory, i.e. the fifth order polynomial (Flash and Hogan, 1985). (2) FIRM selects the first via-point at the point of maximal squared error between the given trajectory and the generated trajectory. Then, FIRM generates a new trajectory passing through this via-point (middle column). (3) This procedure continues (right column) until the error between the given trajectory and the generated trajectory becomes sufficiently small.

Fig. 4 shows the via-points extracted from the human demonstration and the optimal trajectories generated from these via-points in Cartesian space.

*Transforming the human movement to that for the robot*. The Cartesian position and orientation were transformed into the robot's configuration space, i.e. joint angle. The robot is the SARCOS dexterous slave arm which has almost the same kinematic structure (seven degrees of freedom) as the human arm. When we transform the desired trajectory in Cartesian space to that in joint space, we can precisely calculate the joint trajectory by calculating inverse kinematics at the all sampling point. However the smoothness of the joint trajectory is not guaranteed, and the inverse kinematics calculation takes a long time. For smoothness of desired joint trajectory, and for rapid and precise calculation, we used the following method, schematically illustrated in Fig. 5. The broken lines in (i) of the figure indicate the

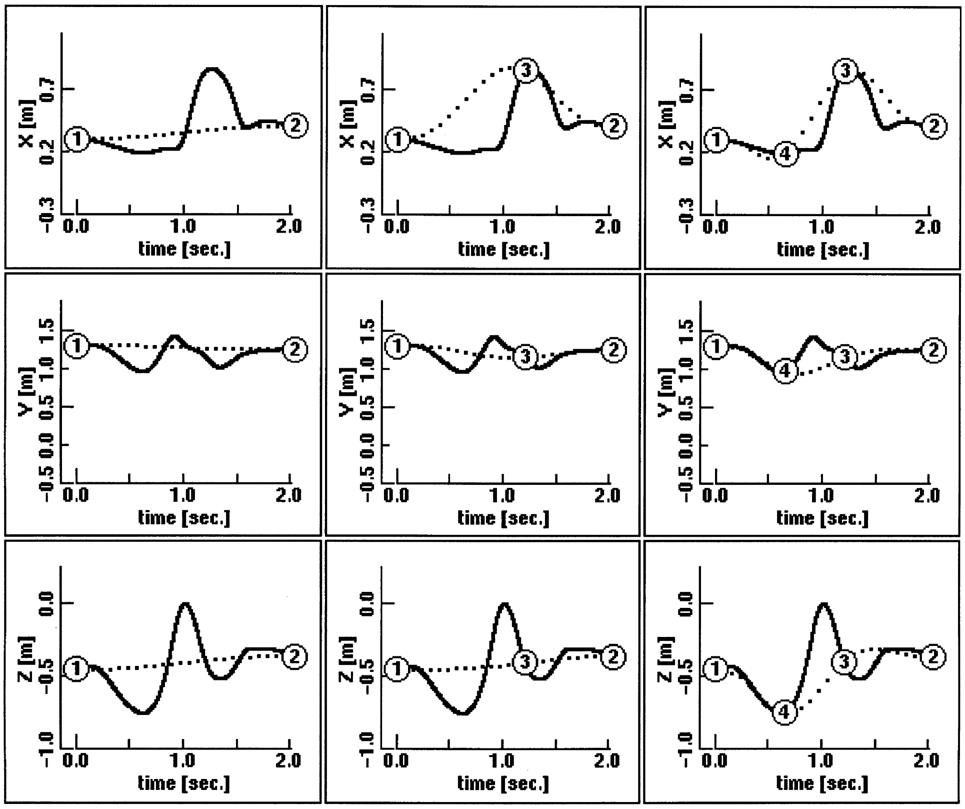


Fig. 3. Schematic illustration of algorithm for extracting via-points. This algorithm approximately minimizes squared error of given trajectory (solid line) and generated trajectory (dotted line) while extracting via-points. *Left*: generation of optimal trajectory which connects start (1) and end (2) points. *Middle*: extracting first via-point (3). *Right*: extracting next via-point (4).

desired trajectory generated from via-points in Cartesian space. First, we obtain via-points in joint space by inverse kinematics (IK) from the Cartesian via-points. Then we generate an optimal trajectory, solid line in (ii), from these via-points in joint space. The expected Cartesian trajectory, solid line in (iii), is calculated by forward kinematics (FK) from this optimal trajectory in joint space at the all sampling points. Second, we add the supplemental via-points, A and B in (iv), in joint space at the points (displayed by vertical arrows) of maximal squared error between the desired (broken line) and the expected (solid line) trajectory in Cartesian space. The supplemental via-points (A and B) locations are calculated by inverse kinematics from Cartesian desired trajectory at these points. Finally, the optimal trajectory in joint space is generated from these via-points (1, A, 2, B, and 3). The resulting expected Cartesian trajectory, the solid line in (v), agrees well with the Cartesian desired trajectory.

*Measurement of robot execution*. The ball, racket, and goal positions were measured by a three-dimensional visual sensing system during the robot's task execution (QUICKMAG: tracking color blobs with two cameras). This system can sample the center of a specific color blob at a sampling rate of 60 Hz, and the precision of the position sensing was approximately 5–10 mm in this experimental setting.

The left column in Fig. 6 shows the first trial executed by the SARCOS arm. The ball was not thrown properly so the robot missed the ball.

### 2.1.2. Learning subtask 1 (throwing the ball up)

For the SARCOS arm to hit the ball in the center of the racket, we modified the via-points $S_H$ around the ball throwing. To adjust the ball throwing so that the ball fell exactly in the center of the racket, we selected the two via-points that are located before and after the release of the ball from the cup.

$$S_H = (x_3, y_3, z_3, x_4, y_4, z_4)^T$$

The desired task $T_{Hd}$ and the realized task $T_H$ are the position at the center of the racket and the position of the ball at the moment of impact $t_{HIT}$ on the racket, respectively.

$$T_{Hd} = (x_{Hd}, y_{Hd}, z_{Hd})^T$$

$$T_H = (x_H, y_H, z_H)^T$$

To improve the robot's performance, the via-point locations are modified by the following Newton-like method.

$$S_H^{n+1} = S_H^n + J_H B_H (T_{Hd} - T_H^n) \quad (1)$$

Here, $S_H^n$ denotes the Cartesian via-points of the SARCOS arm in the *n*th iteration of the Newton-like method. $T_{Hd}$

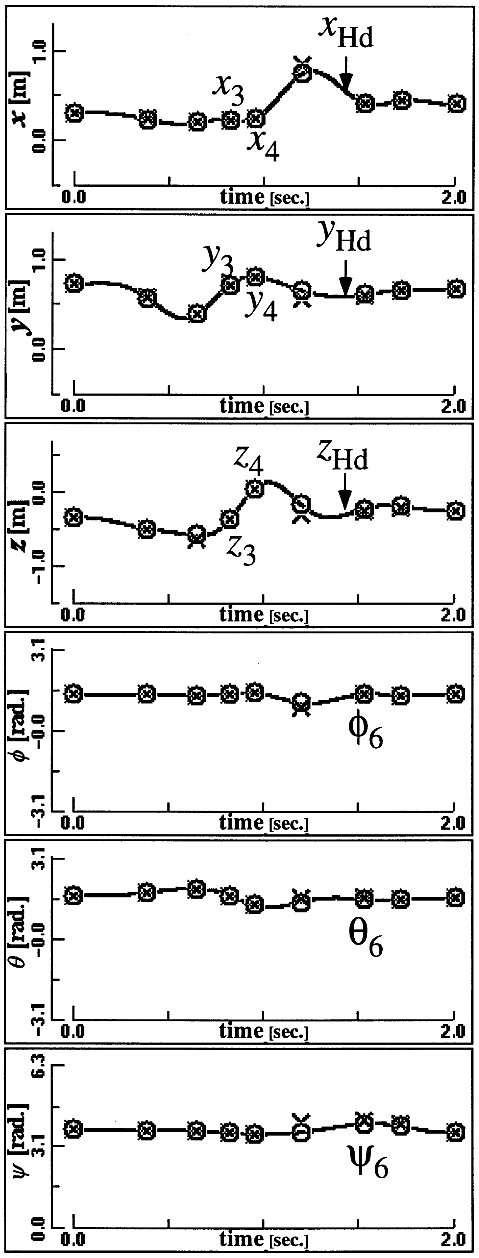


Fig. 4. Via-points extracted from human demonstration and optimal trajectories generated from these via-points in Cartesian space. $x, y, z$ are Cartesian position ($x$ positive rightward, $y$ positive anterior, and $z$ positive upward). $\phi$, $\theta$, $\psi$ are orientation (Z–Y–Z Euler angle) measured in radian. Each rotation is performed about an axis attached to the racket. Rotate about $z_{\text{racket}}$ by angle $\phi$, then rotate about $y_{\text{racket}}$ by angle $\theta$, and then rotate about $z_{\text{racket}}$ by angle $\psi$ (see Craig (1989)).

and $T_H^n$ are the desired and realized task representations, respectively. The superscript # denotes the simply regularized generalized inverse matrix. Since it is difficult to analytically compute the Jacobian matrix $J_H = (\partial T_H/\partial S_H)$, we use the estimated Jacobian matrix $J_H = (\delta T_H/\delta S_H)$. To stabilize the convergence of learning, we set the gain matrix $B_H = \text{diag}(0.5, 0.5, 0.5)$.

To estimate the matrix $J_H$, we observed the changes in the behavior of the subtask 1 $\delta T_H$ when the perturbation

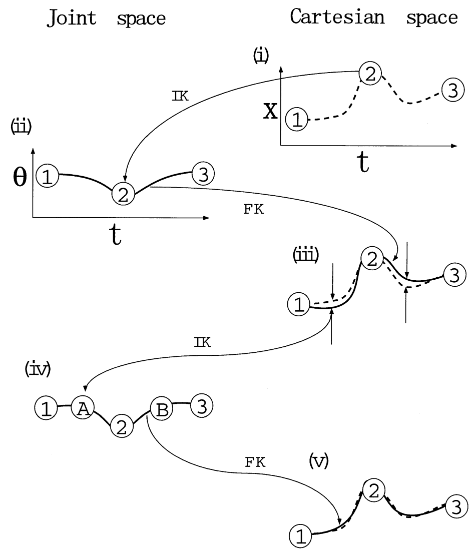


Fig. 5. Schematic illustration of algorithm for transforming human movement to that of a robot. 1, 2, and 3 show via-points. Broken lines show desired trajectory in Cartesian space generated from Cartesian via-points. Solid lines (left culums) show optimal trajectory generated from via-points in joint space. Solid lines (right culums) show expected trajectory calculated by forward kinematics (FK) from joint trajectory at all sampling points. Arrows in (iii) indicate points at maximal squared error between desired trajectory in Cartesian space and expected trajectory. A and B show supplemental via-points.

$\pm\,\delta x_3, \cdots, \pm\,\delta z_4$ (0.01 m magnitude) is added to the via-points (shown in lower part of Fig. 7 by '$*$').

$$J_H = \frac{\delta T_H}{\delta S_H} = \begin{pmatrix} \frac{\delta x_H}{\delta x_3} & \frac{\delta x_H}{\delta y_3} & \frac{\delta x_H}{\delta z_3} & \frac{\delta x_H}{\delta x_4} & \frac{\delta x_H}{\delta y_4} & \frac{\delta x_H}{\delta z_4} \\ \frac{\delta y_H}{\delta x_3} & \frac{\delta y_H}{\delta y_3} & \frac{\delta y_H}{\delta z_3} & \frac{\delta y_H}{\delta x_4} & \frac{\delta y_H}{\delta y_4} & \frac{\delta y_H}{\delta z_4} \\ \frac{\delta z_H}{\delta x_3} & \frac{\delta z_H}{\delta y_3} & \frac{\delta z_H}{\delta z_3} & \frac{\delta z_H}{\delta x_4} & \frac{\delta z_H}{\delta y_4} & \frac{\delta z_H}{\delta z_4} \end{pmatrix} \quad (2)$$

The middle panel of Fig. 6 shows the 25th trial. These trials include the trials for estimating $J_H$. The robot was able to hit the ball in the center of the racket, but the ball did not fall into the goal.

### 2.1.3. Learning subtask 2 (hitting the ball)

When the robot began to hit the ball in the center of the racket, we started learning subtask 2. At this time, we modified the via-point $S_G$ which was near the time of ball impact as shown in Fig. 4, and subsequently affected the direction of the ball.

$$S_G = (\phi_6, \theta_6, \psi_6)^T$$

Here $\phi_6, \theta_6, \psi_6$ are the orientation: Z–Y–Z Euler angle, measured in radian. Each rotation is performed about an axis attached to the racket. Rotate about $z_{\text{racket}}$ by angle $\phi$, then rotate about $y_{\text{racket}}$ by angle $\theta$, and then rotate about $z_{\text{racket}}$ by angle $\psi$ (see Craig, 1989).

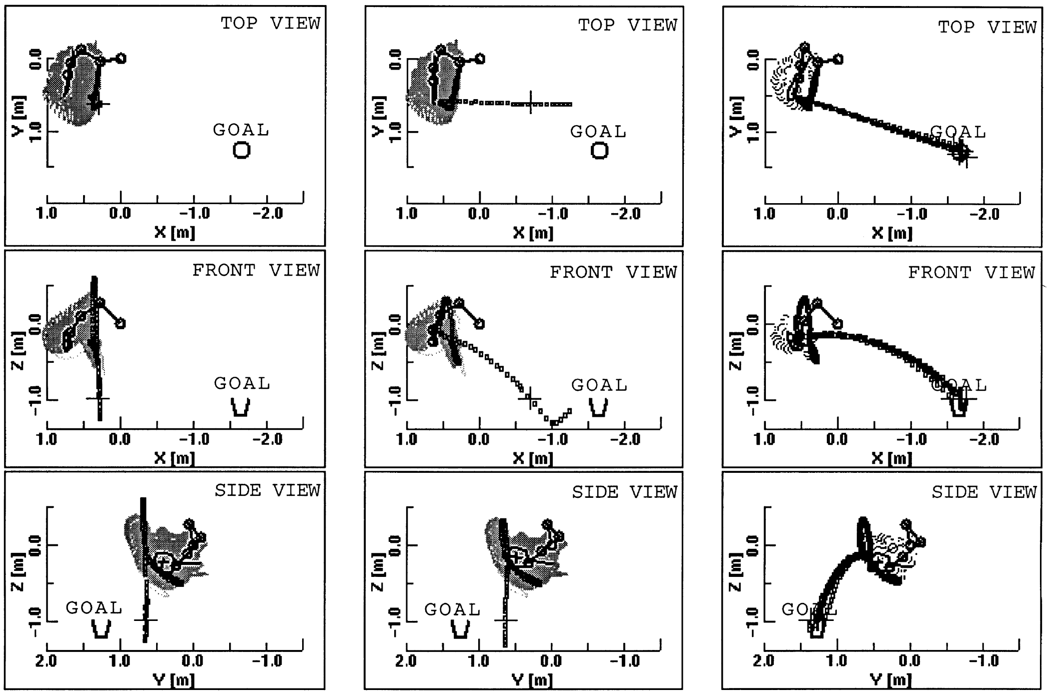


Fig. 6. Three-dimensional views of first (left), 25th (middle), and 60th to 65th (right) trial(s) of a tennis serve executed by the SARCOS arm. Six trials are superimposed in the right panel. The stick figure displays the SARCOS arm posture at hit time. Stick figure was made from realized joint angles of the SARCOS arm by a forward-kinematics equation. Top, middle, and bottom rows are projections into $x$–$y$, $x$–$z$, and $y$–$z$ planes, respectively.

The desired task $T_{\mathrm{Gd}}$ is the goal position. The realized task $T_{\mathrm{G}}$ is the ball position when the ball falls to the same height as the upper ring of the goal.

$$T_{\mathrm{Gd}}=(x_{\mathrm{Gd}},y_{\mathrm{Gd}})^{\mathrm{T}}$$

$$T_{\mathrm{G}}=(x_{\mathrm{G}},y_{\mathrm{G}})^{\mathrm{T}}$$

To estimate the matrix $J_{\mathrm{G}}$, we observed the changes in the behavior of subtask 2, $\delta T_{\mathrm{G}}$, when the perturbation $\pm\,\delta\phi_6$, $\pm\,\delta\theta_6$, $\pm\,\delta\psi_6$ (1° magnitude) is added to the via-points (shown in lower part of Fig. 7 by '$\oplus$').

$$J_{\mathrm{G}}=\frac{\delta T_{\mathrm{G}}}{\delta S_{\mathrm{G}}}=\begin{pmatrix}\dfrac{\delta x_{\mathrm{G}}}{\delta\phi_6} & \dfrac{\delta x_{\mathrm{G}}}{\delta\theta_6} & \dfrac{\delta x_{\mathrm{G}}}{\delta\psi_6}\\[2ex] \dfrac{\delta y_{\mathrm{G}}}{\delta\phi_6} & \dfrac{\delta y_{\mathrm{G}}}{\delta\theta_6} & \dfrac{\delta y_{\mathrm{G}}}{\delta\psi_6}\end{pmatrix} \qquad (3)$$

The via-point locations are modified by the following Newton-like method.

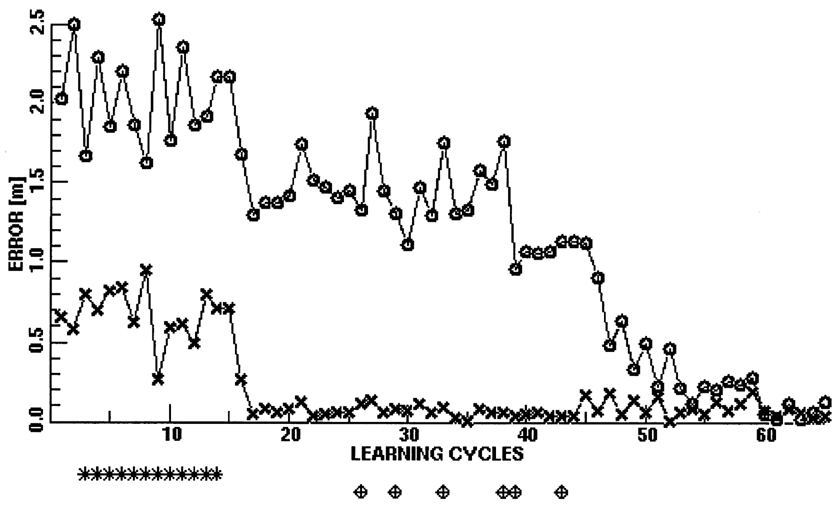


Fig. 7. Learning curves of tennis serve robot experiment. ' × ' indicates distance from ball to racket center when ball should be hit by racket. 'O' indicates distance from ball to goal when ball falls at the same height as goal. Abscissa indicates learning cycles.

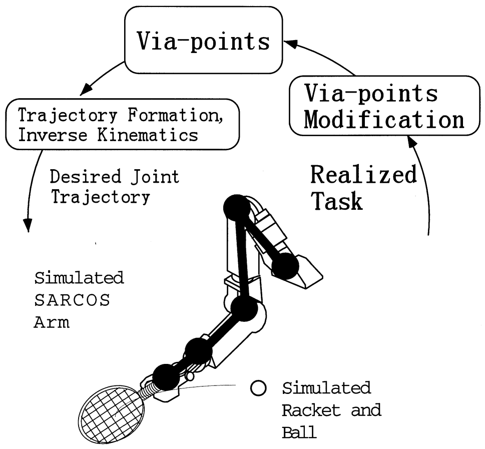


Fig. 8. Schematic diagram of a simulated tennis serve learning robot.

$$S_G^{n+1} = S_G^n + J_G B_G (T_{Gd} - T_G^n) \tag{4}$$

$$B_G = \mathrm{diag}(\alpha, \alpha)$$

$$\alpha = \begin{cases} 0.4 & \text{if } d < 0.05 \\ 0.88\text{–}d & \text{if } d > 0.05 \text{ and } d < 0.1 \\ 0 & \text{otherwise} \end{cases}$$

Here, $d = |T_{Hd} - T_H|$ is the distance from the center of the racket to the ball at the moment of impact $t_{HIT}$. If $d < 0.1$ m, then the orientation of the racket is modified, otherwise it is not modified (instead, the learning for subtask 1, Eq. (5) is performed).

Fig. 7 shows the learning curve. To estimate $J_H$ and $J_G$, we observed changes in behavior in subtask 1 and subtask 2 (shown in lower part of graph by '$*$' and '$\oplus$', respectively) when perturbation was added on via-points. Trials 26 to 43 were performed for estimate of $J_G$. Because we wanted to estimate $J_G$ as correctly as possible, if $d <$ 0.1 m then we adopted this trial for estimation of $J_G$, otherwise learning for subtask 1 was performed. Since the ball position varies at the time of impact, we had to skip some trials to estimate $J_G$. Eleven trials out of 17 (trials 26 to 43) were wasted.

After $J_G$ was estimated, we started to perform learning for subtask 2 (trials 45 to 60). After trial 60 was finished, the next five trials were performed by using the same desired trajectory as trial 60 but without the learning procedure. The right panel in Fig. 6 shows trials 60 to 65 (superimposed in the figure). The ball direction was dispersed within the goal diameter (about 250 mm).

### 2.2. Selecting via-points for the effective control variable

In Section 2.1 we intuitively selected the via-points for a control variable, and the selection is unfounded. It is difficult to find the optimal choice of via-points which gives the most rapid convergence of learning. However, we can find a better selection of via-points using the following method. In this section, all experiments were conducted using computer simulation. Fig. 8 shows a schematic diagram of a simulated tennis serve leaning a robot. The demonstrated trajectory of human subject is the same as that in Section 2.1.

#### 2.2.1. Selecting via-points for subtask 1 (throwing the ball up)

To estimate the Jacobian, we observed the changes in the behavior of the subtasks $\delta T$ when the small perturbation $\delta S$ (0.05 m magnitude for the $x$,$y$,$z$ component and 5° magnitude for the $\phi, \theta, \psi$ component) is added to the via-points.

We first estimate the full Jacobian matrix from the first to the eighth via-points components as follows.

$$\frac{\delta T_H}{\delta S} = \begin{pmatrix} \frac{\delta x_H}{\delta x_1} & \frac{\delta x_H}{\delta y_1} & \cdots & \frac{\delta x_H}{\psi_8} \\ \frac{\delta y_H}{\delta x_1} & \frac{\delta y_H}{\delta y_1} & \cdots & \frac{\delta y_H}{\delta \psi_8} \\ \frac{\delta z_H}{\delta x_1} & \frac{\delta z_H}{\delta y_1} & \cdots & \frac{\delta z_H}{\delta \psi_8} \end{pmatrix} = (\boldsymbol{h}_1, \cdots, \boldsymbol{h}_{48}) \tag{5}$$

Let us consider the norm of the column vector for modification of the via-points in the subtasks. For example, the norm of the eighth column vector is the modification of the $y$ component of the second via-point in subtask 1.

$$\|\boldsymbol{h}_8\| = \sqrt{\left(\frac{\delta x_H}{\delta y_2}\right)^2 + \left(\frac{\delta y_H}{\delta y_2}\right)^2 + \left(\frac{\delta z_H}{\delta y_2}\right)^2} \tag{6}$$

We can select a number of components of via-points that have large values of $\mathbf{h}_j$ for the control variable.

In the upper panel of Fig. 9, the upper abscissa is a number of via-points. The lower abscissa is the number of columns $j$. The ordinate is a value of $\|\boldsymbol{h}_j\|$. The dark area in the figure corresponds to the $x, y, z$ component, and the white area corresponds to the $\phi, \theta, \psi$ component.

From the upper panel of Fig. 9 we can see that columns 13, 14, 15, 19, 20 and 21 are the most effective components in subtask 1. Therefore, we can select the following via-points for the control variable.

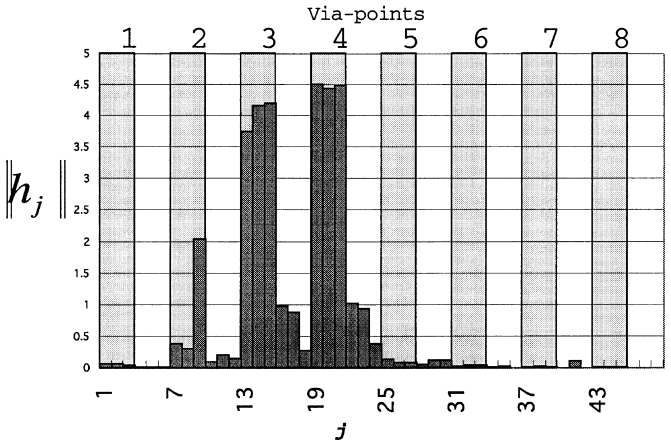
$$S_H = (x_3, y_3, z_3, x_4, y_4, z_4)^T$$

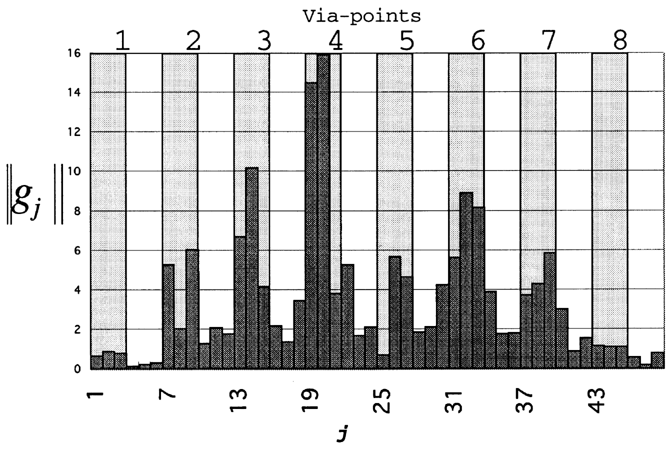
The third and fourth via-points are near the time of the ball release from the cup. This result completely agrees with the ad hoc choice of via-points made in the real robot experiment.

#### 2.2.2. Selecting via-points for subtask 2 (hitting the ball)

We performed the computer simulation by using the via-points selected in the previous section. When the ball hits the center of the racket face, we estimate the Jacobian for subtask 2 as:

$$\frac{\delta T_G}{\delta S} = \begin{pmatrix} \frac{\delta x_G}{\delta x_1} & \frac{\delta x_G}{\delta y_1} & \cdots & \frac{\delta x_G}{\delta \psi_8} \\ \frac{\delta y_G}{\delta x_1} & \frac{\delta y_G}{\delta y_1} & \cdots & \frac{\delta y_G}{\delta \psi_8} \end{pmatrix} = (\boldsymbol{g}_1, \cdots, \boldsymbol{g}_{48}) \tag{8}$$





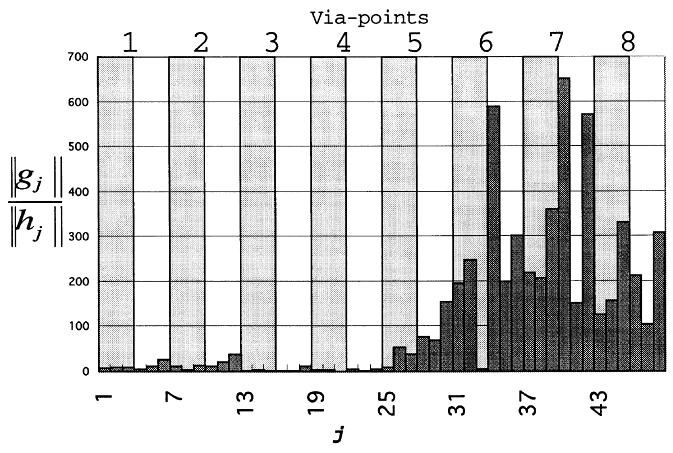


Fig. 9. Modification effect of via-points on subtask 1 (upper panel) and subtask 2 (middle and lower panels).

In the same way as described in Section 2.2.1, the middle panel in Fig. 9 shows $\|\boldsymbol{g}_j\|$. We performed the computer simulation with the via-points that had a large value of $\|\boldsymbol{g}_j\|$. The learning did not converge. The reason is that the modification of these via-points affects subtask 2 as well as subtask 1. Since we do not want to disturb subtask 1, we must select via-points that only affect subtask 2. In order to do so, the ratio of $g_j$ to $h_j$ is plotted. The lower panel in Fig. 9 shows $\|\boldsymbol{g}_j\|/\|\boldsymbol{h}_j\|$. From this, we see that columns 34, 40 and 42, i.e. $\phi_6, \phi_7, \psi_7$, are the most effective components for subtask 2. Thus, we selected the via-points for subtask 2 as:

$$S_G = (\phi_6, \phi_7, \psi_7)^T \tag{9}$$

By using these components of via-points for the control variable, learning converges in a stable manner.

In the real robot experiment, we chose the components of via-points

$$S_G = (\phi_6, \theta_6, \psi_6)^T$$

which are near the time of ball impact. We do not need to select $\theta_6$ for the control variable. It scarcely affects the direction of the bounced ball because $\theta_6$ is the rotation about an axis perpendicular to the racket face.

This suggests that although the components of the seventh via-points are not used for the real robot learning, they are important for learning. More rapid learning is possible if we select the via-points for this section in the real robot experiment.

## 3. Adapting to change in the behavior and the environment

Uncertainty in the environment such as motion fluctuation or noise of measurement imposes negative effects on estimation of the Jacobian. When we attempt to make a robot learn with a nonlinear controlled object such as a pendulum, the learning is difficult because the Jacobian changes dramatically with the learning progress.

We can stabilize the learning if we re-estimate the Jacobian during learning. However, automatically determining the re-estimation timing is difficult. Moreover, executing trials for perturbed data which are not used in the learning itself, but only for re-estimation of the Jacobian, is wasteful.

In Section 3.1 we describe the method for re-estimating the Jacobian without extra trials. In Section 3.2 we examine how to adapt to changes in the environment in the computer simulation of the tennis serve task. In Section 3.3 we examine how to adapt to changes in the behavior of the controlled object in the computer simulation of the pendulum upswing task.

### 3.1. Automatically re-estimating the Jacobian

Before we started the learning trials, we executed the perturbed trials for the initial estimation of the Jacobian. Once we completed the first estimation, we had good data for re-estimating the Jacobian (data from past learning records). We describe a method for re-estimating the Jacobian using past learning records instead of perturbed data.

Let us express the $K+1$ sets of the via-points $S = (x_1, \cdots, x_L)$ and the realized task $T = (\tau_1, \cdots, \tau_M)$ from the past $(n-K)$th to $n$th trials as:

$$\{S^{n-K}, T^{n-K}\}, \cdots, \{S^n, T^n\}$$

From the above $K+1$ sets of the via-points and the realized task, we obtain two matrices

$$\mathcal{X} = \begin{pmatrix} \delta x_1^1 & \cdots & \delta x_L^1 \\ & \vdots & \\ \delta x_1^j & \cdots & \delta x_L^j \\ & \vdots & \\ \delta x_1^K & \cdots & \delta x_L^K \end{pmatrix} \tag{10}$$

$$\mathcal{T} = \begin{pmatrix} \tau x_1^1 & \cdots & \tau x_L^1 \\ & \vdots & \\ \tau x_1^j & \cdots & \tau x_L^j \\ & \vdots & \\ \tau x_1^K & \cdots & \tau x_L^K \end{pmatrix} \tag{11}$$

The $j$th row vectors of the above matrices are the difference between two trials as:

$$(\delta x_1^j, \cdots, \delta x_L^j) = S^i - S^{i-1}$$

$$(\delta\tau_1^j, \cdots, \delta\tau_M^j) = T^i - T^{i-1}$$

where $i = n - K + j - 1$. There is relationship $\mathcal{T}^T = J\mathcal{X}^T$ between $\mathcal{X}$, $\mathcal{T}$ and the Jacobian $J$. By transforming the above equation, we obtain the equation:

$$J = (\mathcal{X}\,\mathcal{T})^T \tag{12}$$

Here, the superscript # denotes the simply regularized generalized inverse matrix. We can re-estimate the Jacobian automatically with Eq. (12) using data from past learning records.

### 3.2. Tennis serve simulation

When the real robot hits the ball, there is some dispersion of the ball bouncing direction (the last six trials in Fig. 6 on the right). The dispersion of the operation or the noise of the measurement imposes negative effects on the Jacobian estimation. We can cancel the dispersion by calculating the average using the repeated trials. Although these trials are merely for learning preparation, it takes a long time to repeat the perturbed trials. Eq. (12) will have the same effect as averaging if $K$ is greater than $M$. In this section, we show the efficiency of the method described in Section 3.1 also for the improvement of the Jacobian accuracy.

The experimental setting is the same as described in Section 2.2 except for the dispersion of the ball. We dispersed the ball by adding random disturbance to the velocity of the ball when the ball was released from the cup and bounced from the racket. The disturbance range was randomly selected from −3% to +3%.

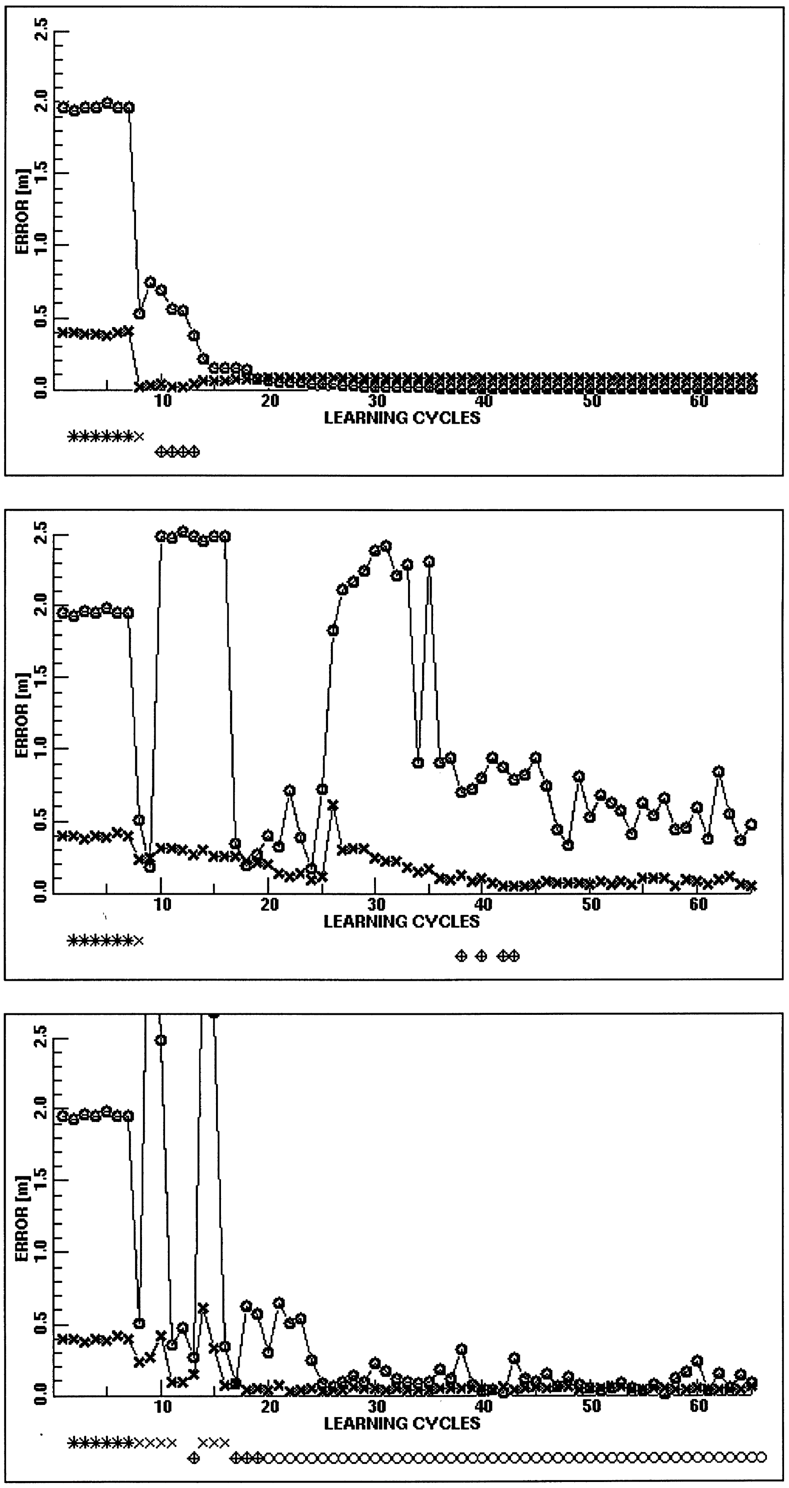


Fig. 10. Learning curves of tennis serve simulation. ' × ' indicates distance from ball to racket center when the ball should be hit by racket. '○' indicates distance from ball to goal when ball falls at the same height as goal. Abscissa indicates learning cycles.

The learning experiment was performed under the following three conditions.

**Condition 1:** The ball motion is not disturbed. The Jacobian matrix is estimated for the first part of the trials.

**Condition 2:** The ball motion is disturbed. The Jacobian matrix is estimated for the first part of the trials.

**Condition 3:** The ball motion is disturbed. The Jacobian matrix is estimated for the first part of the trials, and then it is automatically re-estimated using the record of the learning.

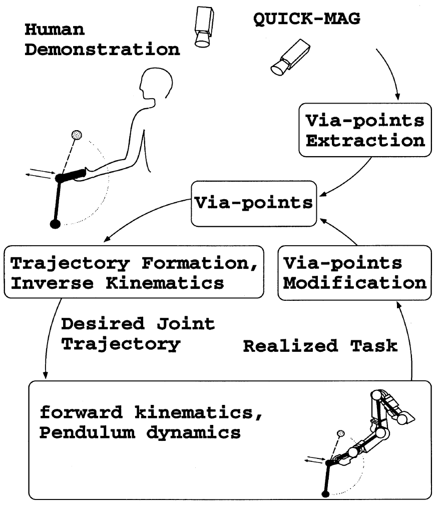


Fig. 11. Schematic diagram of a simulated pendulum upswing learning robot.

Under condition 3, we corrected the Jacobian in every trial by using Eq. (12) ($K = 15$ when $n \leq 15$ then $K = n$). We set the $j$th row vectors of Eqs. (10) and (11) as:

$$(\delta x_3^j, \cdots, \delta z_4^j) = S_H^i - S_H^{i-1}$$

$$(\delta x_H^j, \delta y_H^j, \delta z_H^j) = T_H^i - T_H^{i-1}$$

for subtask 1 and

$$(\delta \phi_6^j, \delta \phi_7^j, \delta \psi_7^j) = S_G^i - S_G^{i-1}$$

$$(\delta x_G^j, \delta y_G^j) = T_G^i - T_G^{i-1}$$

for subtask 2, where $i = n - K + j - 1$.

We observed the changes in the behavior of subtask 1 and subtask 2 (shown in the lower part of the graph by '$*$' and '$\oplus$' in Fig. 10, respectively) when the perturbation $\delta x_3, \delta y_3, \delta z_3, \delta x_4, \delta y_4, \delta z_4$ and $\delta \phi_6, \delta \phi_7, \delta \psi_7$ was added in the via-points in order.

We corrected the Jacobian of subtask 1 and subtask 2 (shown in the lower part of the graph by '$\times$' and '$\bigcirc$', respectively) using Eq. (12).

The upper panel in Fig. 10 shows learning convergence under condition 1. Learning is stable and rapidly converges.

The middle panel in Fig. 10 shows learning convergence under condition 2. Learning is unstable and converges slowly. As in the real robot (Fig. 7), we had to skip some trials to estimate $J_G$ because the ball position varied on impact. A great number of trials (from 9 to 43) were wasted.

The lower panel in Fig. 10 shows learning convergence under condition 3. Learning is stable and converges rapidly. We were able to use all trials to re-estimate the Jacobian.

From Fig. 10 we confirm that the auto-correction of the Jacobian is effective for the looseness of the controlled object's behavior.

## 3.3. Upswing simulation

In this section, we demonstrate learning of an upswing task for learning about the nonlinear system. Fig. 11 shows a schematic diagram of the upswing learning robot. The goal of the upswing task is to move the hand so that the pendulum, which initially hangs down, swings up to the inverted position. The hand holds the axis of the pendulum, and the pendulum rotates about this hinge with an angular movement.

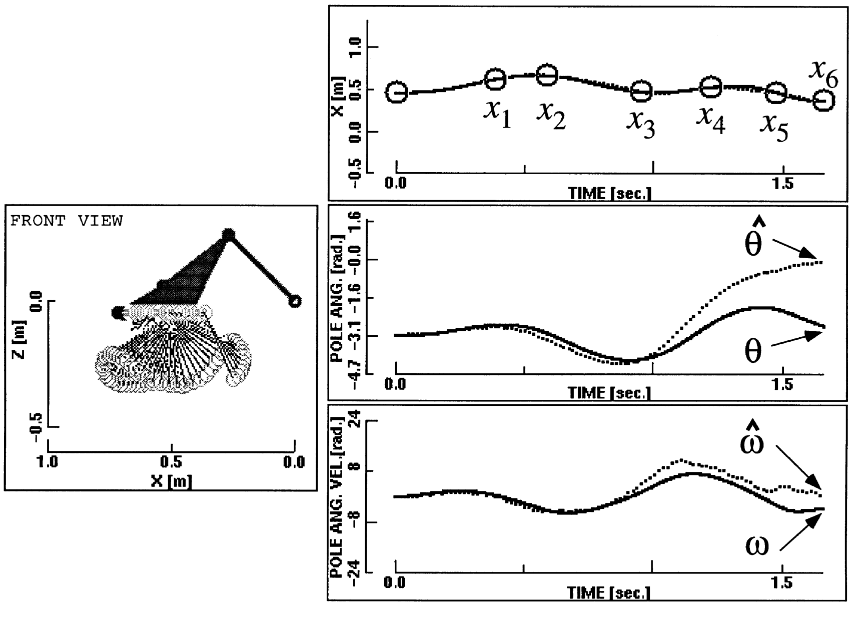


Fig. 12. First trial of upswing task. Left panel indicates pendulum movement. Upper panel on right side indicates time course of hand position. Broken line indicates time course of hand position. Dotted line indicates time course of position of pendulum tip. Circles indicates via-points. Middle and lower panels of right side indicate time course of pendulum angle and angular velocity of pendulum, respectively. Human demonstration (dotted lines) and robot execution (solid lines) are shown.

### *3.3.1. Experimental setup*

The human demonstration data of the upswing and the mathematical models of the pendulum are the same as those described in Atkeson and Schaal (1997). A general version of the upswing task allows both horizontal and vertical hand motion perpendicular to the pendulum axis. However, to simplify the task, we restricted the hand motion to a horizontal line with the pendulum axis perpendicular to this line. Humans naturally use both horizontal and vertical hand motions to do the task, but they can restrict their motion to mostly horizontal hand motions if asked to do so.

We extracted the via-points in the same way as in Section 2.1, although we only used the $x$ component in this experiment.

For simplicity, we used the mathematical models of the SARCOS arm with only the kinematics equation (the dynamics equation was not included). The pendulum starts at $\theta = -\pi$ and a successful upswing moves the pendulum to $\theta = 0$. A dynamic model of an idealized pendulum (all mass concentrated at the tip) attached on a horizontally moving hand is:

$$\dot{\theta}_{k+1} = (1-\alpha_1)\dot{\theta}_k + \alpha_2[\sin(\theta_k) + \ddot{x}_k\cos(\theta_k)/g]$$

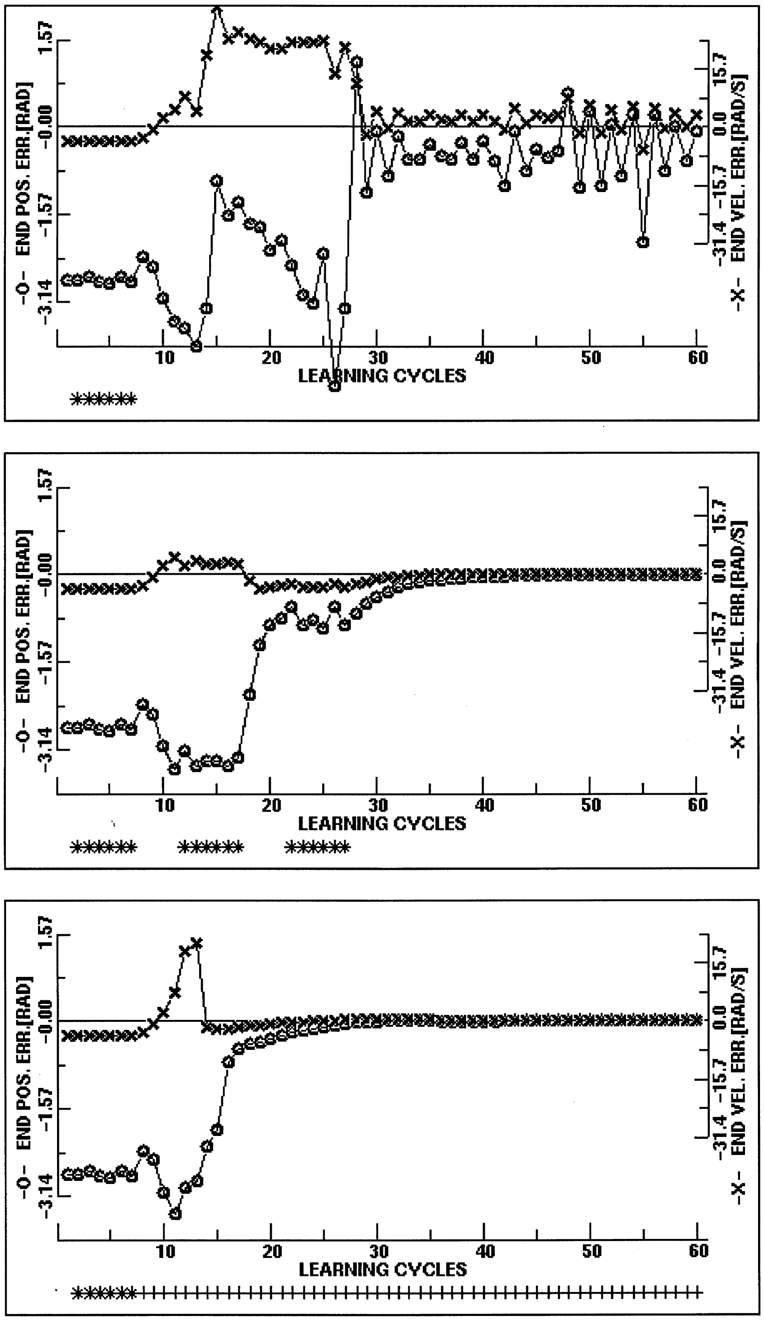


Fig. 13. Learning curves of upswing simulation and error at the end time. × and ○ indicate realized pole angle $\theta$ and realized pole velocity $\omega$ at the end of movement trajectory, respectively.

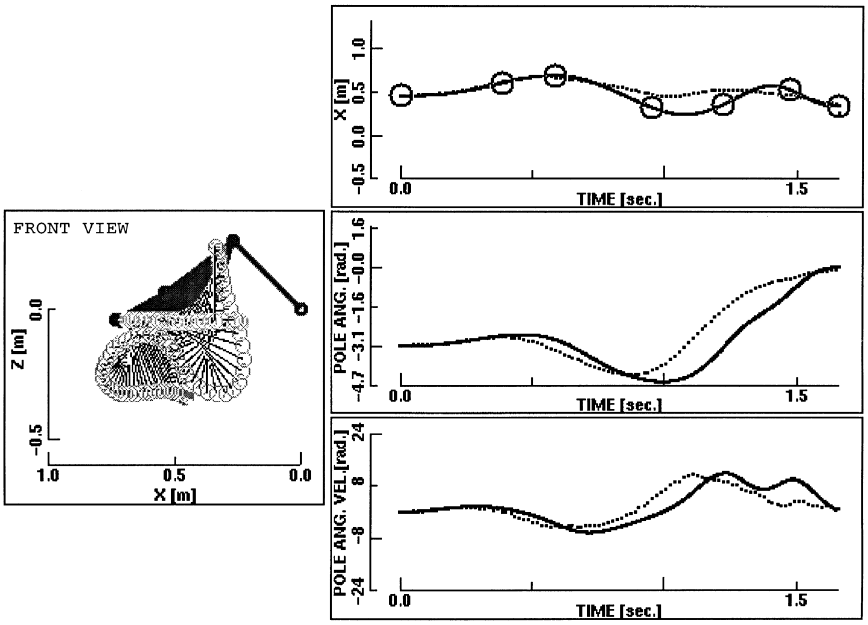


Fig. 14. 30th trial of upswing task executed by simulated SARCOS arm. See Fig. 12 for legend description.

where $\theta$ is the pendulum angle, $\dot{\theta}$ is the pendulum angular velocity, $\ddot{x}$ is the horizontal hand acceleration, $\alpha_1$ is the viscous damping, $\alpha_2$ is $\Delta g/l$, $\Delta$ is the time step 0.0167 s, $g$ is the gravitational acceleration 9.81 m s$^{-2}$, and $l$ is the length of the pendulum 0.35 m. Idealized values for the $\alpha$ based on these parameters are $\alpha_1 = 0$ and $\alpha_2 = 0.47$.

### 3.3.2. Learning upswing

Fig. 12 shows the first trial of the upswing task. The via-points were not modified in this trial. As the figure shows, if we make the robot execute the task using the extracted via-points from the demonstration, the robot fails.

We can list various candidates as the task goal for the upswing. However, in this experiment, we chose the following task goal as the most intuitive one.

Let us denote the desired task and the realized task

$$T_\mathrm{d} = (\hat{\theta}, \hat{\omega})^\mathrm{T},\ T = (\theta, \omega)^\mathrm{T} \tag{14}$$

respectively. $\hat{\theta}, \hat{\omega}$ are the angular position and the angular velocity of the pendulum of the demonstrated movement at the end time, respectively. $\theta, \omega$ are the angular position and the angular velocity of the pendulum of the realized movement by the simulated robot at the end time, respectively.

Let the modified via-points be:

$$S = (x_1, \cdots, x_6)^\mathrm{T}$$

To improve the robot's performance, the via-point locations were modified by the following Newton-like method.

$$S^{n+1} = S^n + J\,B(T_\mathrm{d} - T^n) \tag{15}$$

To stabilize the convergence of learning, we set the gain matrix $B = \mathrm{diag}(0.2, 0.02)$. To estimate the Jacobian $J$, we observed the changes in the behavior of the task $\delta T$ when the perturbation $\delta x_1, \cdots, \delta x_6$ (0.01 m magnitude) was added to the via-points.

$$J = \frac{\delta T}{\delta S} = \begin{pmatrix} \dfrac{\delta\theta}{\delta x_1} & \cdots & \dfrac{\delta\theta}{\delta x_6} \\ \dfrac{\delta\omega}{\delta x_1} & \cdots & \dfrac{\delta\omega}{\delta x_6} \end{pmatrix} \tag{16}$$

Fig. 13 shows learning convergences. We obtained the initial estimation of the Jacobian using Eq. (16) without learning in the first seven trials (shown in the lower part of the graph by '*' in Fig. 13). When conducting learning only by the Jacobian estimated for the first movement trajectory, the learning becomes very late or becomes unstable as shown in the upper panel in Fig. 13 because the learning changes the movement trajectory.

To stabilize the learning, as shown in the middle panel in Fig. 13 we re-estimate the Jacobian manually (shown in the lower part of the graph by '*' in Fig. 13) by adding the perturbation to the via-point during the learning. The timing of re-estimation is determined by trial and error.

Finally, we examined the automatic Jacobian re-estimation method proposed in Section 3.1. After the first seven trials, we re-estimated the Jacobian using the latest trials ($K = 12$, $K = n$ when $n \leq 12$). We set the $j$th row vectors of Eqs. (10) and (11)

$$(\delta x_1^j, \cdots, \delta x_6^j) = S^i - S^{i-1}$$

$$(\delta\theta^j, \delta\omega^j) = T^i - T^{i-1}$$

where $i = n - K + j - 1$. Using Eq. (12) at each learning cycle (shown in the lower part of the graph by '+' in Fig. 13) the upswing learning successfully converges, as shown in the lower panel in Fig. 13. Fig. 14 shows trial 30. The pendulum is in the inverted position.

## 4. Conclusion

We demonstrated that the via-points representation is useful for learning by watching. Our learning framework is applicable to simple tasks with a single goal as well as complicated tasks with a sequence of subgoals. Furthermore, this framework is adaptable to change in the behavior and in the environment.

We correct our movement trajectory using real time visual feedback when we perform a task. For example, when serving in tennis, we usually observe the ball to estimate the place where it will fall, and we correct the striking trajectory. In this paper, we use a purely feed-forward control, that is, we generated the whole movement trajectory before the robot executed the movement. Therefore, it is not possible to generate trajectory during the movement execution, and real time control with visual feedback is also impossible. In the future, in order to make real time trajectory correction possible, the local trajectory generation for the time must be integrated into our system.

We used empirical knowledge of how to divide a tennis serve task into subtasks. To make a robot divide a task automatically, a hierarchical learning scheme (dividing a task into subtasks at higher level, and achieving each subtask at the lower level) must be incorporated into our system. It is perhaps erroneous to think that all control objects and operation can only be handled with the Newton-like method used in this paper. It will be necessary to use a learning scheme such as reinforcement learning, feedback error learning, or to combine them and ascertain their generality and their learning potential.

## Acknowledgements

We would like to thank Professor Atkeson and the members of Department 3 of ATR Human Information Processing Research Laboratories and the members of the Kawato Dynamic Brain Project for useful discussions.

## References

Aboaf, E.W., Atkeson, C.G., & Reinkensmeyer, D.J. (1988). Task-level robot learning. *Proc. IEEE Int. Conf. Robot Auto.* April 24–29, Philadelphia, PA.

Atkeson, C.G., & Schaal, S. (1997). Robot learning from demonstration. *International Conference on Machine Learning.*.

Craig, J.J. (1989). *Introduction to robotics: Mechanics and control* (2nd ed). Reading, MA: Addison-Wesley.

Flash T., & Hogan N. (1985). The coordination of arm movements: An experimentally confirmed mathematical model. *Journal of Neuroscience, 5*, 1688–1703.

Kang S.B., & Ikeuchi K. (1993). Toward automatic robot instruction from perception—Recognizing a grasp from observation. *IEEE Trans. on Robotics and Automation*, *9*, 432–443.

Kawato, M. (1992). Optimization and learning in neural networks for formation and control of coordinated movement. In D. Meyer & S. Kornblum (Eds.), *Attention and performance, vol. 14: Synergies in experimental psychology, artificial intelligence, and cognitive neuroscience—a silver jubilee* (pp. 821–849). Cambridge, MA: MIT Press.

Kawato, M., Gandolfo, F., Gomi, H., & Wada, Y. (1994). Teaching by showing in Kendama based on optimization principle. In M. Marinaro & P. G. Morasso (Eds.), *Proceedings of the International Conference on Artificial Neural Networks* (pp. 601–606). Sorrento, Italy, May 26–29.

Kuniyoshi Y., Inoue H., & Inaba M. (1991). Teaching by showing: generating robot command sequences based on real time visual recognition of human pick and place actions. *JSRJ*, *9*, 295–303.

Miyamoto S., Schaal H., Gandolfo F., Gomi H., Koike Y., Osu R., Nakano E., Wada Y., & Kawato M. (1996). A kendama learning robot based on bi-directional theory. *Neural Networks, 9 (8),* 1281–1302.

Wada Y., & Kawato M. (1993). A neural network model for arm trajectory formation using forward and inverse dynamics models. *Neural Networks*, *6*, 919–932.

Wada Y., & Kawato M. (1995). A theory for cursive handwriting based on the minimization principle. *Biological Cybernetics*, *73 (1)*, 3–13.