A Kendama Learning Robot Based on Bi-directional Theory

HIROYUKI MIYAMOTO,1 STEFAN SCHAAL,1,2 FRANCESCA GANDOLFO,3 HIROAKI GOMI,4 YASUHARU KOIKE,5 RIEKO OSU,1,6 ERI NAKANO,1,7 YASUHiro WADA,8 AND MITSUO KAWATO1

1ATR Human Information Processing Research Laboratories, Kyoto Japan, 2Georgia Institute of Technology, 3Massachusetts Institute of Technology, 4NTT Basic Research Laboratories, Kanagawa, Japan, 5Toyota Motor Corporation, 6Kyoto University, 7Kobe University, and 8Kawasaki Steel Corporation

(Received 28 September 1995; revised and accepted 28 March 1996)

Abstract—A general theory of movement-pattern perception based on bi-directional theory for sensory-motor integration can be used for motion capture and learning by watching in robotics. We demonstrate our methods using the game of Kendama, executed by the SARCOS Dextrous Slave Arm, which has a very similar kinematic structure to the human arm. Three ingredients have to be integrated for the successful execution of this task: The ingredients are (1) to extract via-points from a human movement trajectory using a forward-inverse relaxation model, (2) to treat via-points as a control variable while reconstructing the desired trajectory from all the via-points, and (3) to modify the via-points for successful execution. In order to test the validity of the via-point representation, we utilized a numerical model of the SARCOS arm, and examined the behavior of the system under several conditions. Copyright © 1996 Elsevier Science Ltd.

Keywords—Teaching by showing, Task-level learning, Dynamic optimization.

1. INTRODUCTION

Several interesting findings in neuroscience indicate the close relationship between the active execution of a movement and the passive monitoring of a movement when performed by somebody else. For example, Perrett et al. (1989) found a variety of cell types in the superior temporal sulcus of the monkey brain that are selective for the sight of hand and body actions. di Pellegrino et al. (1992) reported that neurons of the rostral part of the inferior premotor cortex of the monkey discharge during goal-directed hand movements, and also discharge when the monkey observes specific meaningful hand movements performed by the experimenters. Decety et al. (1994) found that during the observation of hand movements, without actual movement, and during motor imagery, brain activation was mainly found in visual cortical areas, and also in areas related to the motor behavior of normal human subjects with positron emission tomography. These neurophysiological data suggest that observations and mental images of motor behaviors might play an important role in motor learning. Indeed, when we learn novel motor skills, we often turn our attention to a teacher who demonstrates new motor behaviors.

The idea of making use of a demonstration has been picked up by researchers in robotics for higher-level task learning (e.g., Kang & Ikeuchi, 1993; Kuniyoshi et al., 1994), a research trend away from simple trajectory following. When we make a robot perform a task, it is difficult to overcome fluctuations and environmental uncertainty with conventional teaching methods such as "teaching play back". In this paper we propose a higher level of teaching, "teaching by showing" (or "learning by watching"), in order to overcome some of the problems inherent in the play back approach.

The approach we will pursue is based on the bi-directional theory (Kawato, 1992, 1995), which gives a general computational framework for sensory-motor integration, and derives generic representations for a wide variety of motor behaviors (Section 2). One of the essential ingredients of this theory is to represent behaviors in terms of a sparse via-point...
representation (Section 3). In this paper, we will show 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 has to learn to perform the same task (Section 4). In Section 5 we will examine and discuss the validity of the via-point representation by means of several numerical simulations.

2. BI-DIRECTIONAL THEORY

Fast and coordinated arm movements should be executed under feedforward control since biological feedback loops, in particular those via the periphery, are slow and have small gains. Recent estimates show dynamic stiffness is not very high during movement (Bennett et al., 1992; Bennett, 1993; Gomi et al., 1992). Thus, internal neural models, such as an inverse dynamics model, are necessary (Katayama & Kawato, 1993; Kawato et al., 1993a). Analysis of the activity of single Purkinje cells suggests the existence of an inverse dynamics model in the cerebellum (Shidara et al., 1993). Trajectories of point-to-point arm movements using multi-joints are characterized by roughly straight hand paths and bell-shaped speed profiles (Morasso, 1981). Kinematic and dynamic optimization principles have been proposed to account for these invariant features so far (Flash & Hogan, 1985; Uno et al., 1989a; Kawato, 1992). Experimental data support the dynamic optimization theory, which requires both forward and inverse models of the motor apparatus and the external world (Kawato, 1995).

The problem of controlling goal-directed limb movements can be partitioned conceptually into a set of information-processing subprocesses: trajectory planning, coordinate transformation from extracorporeal space to intrinsic body coordinates, and motor command generation. These subprocesses are required to translate the spatial characteristics of the target or goal of the movement into an appropriate pattern of muscle activation. Over the past decade, computational studies of motor control have become much more advanced while concentrating on these three computational problems.

Many of the models can be broadly classified into one of two alternative classes of theories: unidirectional and bi-directional (see Figure 1). Both types assume a hierarchical arrangement of the three computational problems to be solved for visually-guided reaching and a corresponding hierarchy in the neural representations of these problems: that is, the trajectory planning problem, the coordinate transformation problem and the motor command generation problem on the one side and the desired trajectory in extrinsic space, the desired trajectory in intrinsic space, and the motor commands on the other.

In the unidirectional theory framework, the information flow is only downward, that is, unidirectional. On the other hand, in the bi-directional theory framework, both downward and upward information flow is allowed.

The downward information flow is mediated by the inverse kinematics model and the inverse dynamics model of the controlled object and the environment. The upward information flow is mediated by the forward kinematics model and the forward dynamics model.

2.1. Unidirectional versus Bi-directional Theories

In unidirectional theories, information flows only downward from the higher level to the lower level. As a result, the higher level computational problem is solved without any reference to the lower level computational problems. For example, trajectory planning is solved without using any knowledge about coordinate transformation or motor command generation. Thus, the three problems are solved sequentially step by step (Table 1). That is, first the trajectory planning problem is solved to compute the desired trajectory in the extrinsic space (in many cases task-oriented visual coordinates). Then, the coordinate transformation problem is solved to obtain the desired trajectory in the intrinsic space (joint angles,
TABLE 1
Comparison of the Unidirectional and Bi-directional Theories for Goal-Directed Arm Movements

<table>
<thead>
<tr>
<th>Theory</th>
<th>Unidirectional</th>
<th>Bi-directional</th>
</tr>
</thead>
<tbody>
<tr>
<td>How to solve three computational problems</td>
<td>Sequential</td>
<td>Simultaneous</td>
</tr>
<tr>
<td>Spaces where trajectory is planned</td>
<td>Extrinsic space (task-oriented visual coordinates)</td>
<td>Intrinsic space (body coordinates) and extrinsic space</td>
</tr>
<tr>
<td>Optimization principle (Example)</td>
<td>Kinematic (Minimum-jerk)</td>
<td>Dynamic (Minimum-torque-change)</td>
</tr>
<tr>
<td>Control</td>
<td>Virtual trajectory control</td>
<td>Inverse dynamics model</td>
</tr>
<tr>
<td>Internal models of motor apparatus and environment</td>
<td>Not necessary</td>
<td>Forward dynamics model and inverse dynamics model</td>
</tr>
<tr>
<td>Motor learning</td>
<td>—</td>
<td>Acquisition of internal models</td>
</tr>
</tbody>
</table>

muscle lengths, etc.) from the trajectory in the extrinsic space. Finally, the necessary motor commands for the desired trajectory in the intrinsic space are calculated by a controller.

On the other hand, in bi-directional theories, upward information flow as well as downward flow is allowed. In fact, the former actually is essential in order to solve ill-posedness in the three computational problems in a reasonably short time. As a result, the higher level computational problem can be solved while taking account of events which happen at the lower levels. For example, trajectory planning is influenced by the requirement for smooth motor commands. Thus, the three problems are solved simultaneously rather than sequentially.

One of the most fundamental differences between the two theories concerns the spaces where the trajectory is planned. Consequently, at present their is a controversy concerning which coordinate system, extrinsic (kinematic) or intrinsic (dynamic), is used for trajectory planning.

In the unidirectional theory, the trajectory is assumed to be planned solely in the extrinsic space (usually task-oriented visual coordinates), while all the kinematic and dynamic factors at the lower levels are neglected. On the other hand, in the bi-directional theory, trajectory planning involves both the intrinsic (body coordinates) and extrinsic space. Goals of movements such as the end point of reaching are given in the extrinsic space while necessary constraints to select a unique trajectory (i.e., to resolve the ill-posedness) are given in the intrinsic space. Thus, the two spaces are used simultaneously for trajectory planning.

Almost inseparably coupled to the spaces for trajectory planning, optimization principles for trajectory planning developed by the two theories are markedly different. In the unidirectional theory, because the planning process does not take the lower levels into account, the optimization principle has to be kinematic and all of the dynamic factors are therefore neglected. One representative example is the minimum-jerk model (Flash & Hogan, 1985). On the other hand, in the bi-directional theory, it is possible to use the dynamic optimization principle which takes lower levels into account. One representative example is the minimum-torque-change model (Uno et al., 1989a). Minimum-muscle-tension-change model (Uno et al., 1989b) and minimum-motor-command-change model (Kawato, 1992) are other examples in the bi-directional theory. Others could further be proposed in the bi-directional theory class.

In the purest example of a unidirectional theory, that which combines kinematic path planning with the virtual trajectory control hypothesis (Flash, 1987), the brain does not need to utilize any internal model of the motor apparatus or environment in trajectory planning and control. On the other hand, both the forward dynamics model and the inverse dynamics model are necessary for fast computation of trajectory planning under the bi-directional theory (Kawato, 1992; Wada & Kawato, 1993). In general, inverse models are necessary for fast computation, while forward models are necessary to resolve redundancy problems, or in more intuitive terms, to improve the adaptability of behaviors. These two kinds of models correspond to downward and upward information flows respectively.

These internal models should be learned and stored somewhere in the brain. We believe that this acquisition of internal models of the motor apparatus and the environment forms a major part of early motor learning. A biologically plausible learning scheme to acquire the inverse dynamics model was previously proposed (Kawato et al., 1987; Kawato & Gomi, 1992). We have already obtained some experimental evidence that internal models reside in the cerebellum (Shidara et al., 1993).

2.2. Forward-Inverse Relaxation Neural Network Model and Sequential Movements

We have developed several neural network models which can generate the dynamically optimal trajectory and control a motor apparatus to move along it (Kawato et al., 1989). Recently, we developed a new neural network model which can generate the trajectory within a few iterations (Kawato, 1992;
Wada & Kawato, 1993). It is called the forward-inverse relaxation neural network model (FIRM) because it contains both the forward dynamics model and the inverse dynamics model of the controlled object (Figure 2). Note that the FIRM can generate and control the optimal trajectory in real time according to any one of the dynamic optimization principles (e.g., minimum-torque-change model, minimum-muscle-tension-change model, minimum-motor-command-change model).

In reaching movements, the location of the start and end points and the specified movement duration provide the two-point boundary conditions for the optimization problem. The nonlinear dynamics of the arm govern the relationship between the control variable (joint torques) and the trajectory. The subject can pass through specified via-points. According to Marr’s three-levels of understanding brain function (Marr, 1982) we can summarize our study of reaching movements as follows: (1) The computational theory of reaching is the minimization of one of the intrinsic dynamic variables. (2) The algorithm involves relaxation in a distributed network and the task is represented by the start, via- and end points and the desired movement duration. (3) The hardware is the FIRM network and the associated controllers and actuators.

We hypothesized that this computational framework might be extended to other classes of voluntary movements such as handwriting or speech. For this extension of the theory, the first and third levels are easily transferred, but the representation level needs careful consideration. For speech, we believe that each phoneme determines a via-point location. For handwriting, we developed an algorithm to extract the minimum number of via-points from a given trajectory with some level of error threshold (Wada & Kawato, 1995, see also Section 3.1). If a fixed number of via-points are given and the arm dynamics are known, we can calculate the optimal trajectory passing through these via-points. Our via-point extraction algorithm uses FIRM again and this suggests a duality between movement pattern formation and movement pattern perception (see Section 2.3).

We succeeded in reconstructing a cursive handwriting trajectory quite accurately using about 10 via-points for each character (Wada & Kawato, 1995). The extracted via-points included not only kinematically-definable points with maximum curvature and lowest velocity but also other points not easily extracted by any purely kinematic method which does not take account of the dynamics of the arm or the dynamic optimization principle.

2.3. Bi-directional Motor Theory for Movement Pattern Perception

A simple system for recognizing words from cursive handwriting, based on the above via-point representation, worked without a word dictionary. When the same algorithm was applied to speech articulator motion during natural speech, the extracted via-points corresponded fairly well to the phonemes that were determined from a simultaneously recorded acoustic signal. Natural speech movements were reconstructed well from those phoneme-like via-points (Wada et al., 1995).

Thus, we have accumulated evidence that the bi-directional theory approach is a promising computational model for both generation and perception of several types of movements. We have proposed the same bi-directional architecture for fast computation in early vision, and fast and reliable integration of different vision modules in middle vision problems, including the integration of surface normal estimation, boundary detection and light source estimation in discriminating shape from shading (Kawato et al., 1991, 1993b; Hayakawa et al., 1994).

The mathematical structures of these models are almost identical. Thus, although we do not have any strong psychological, anatomical, or physiological data, it is very tempting to propose that the same bi-directional architecture might be applicable to integration of different modules ranging from sensory information processing to motor control as shown in Figure 3.

The basic assumption in this proposal is that the brain consists of a number of modules which roughly correspond to different cortical areas and these modules are organized in a hierarchical yet parallel manner. We do not assume a single gigantic map which plays a central role in integration. Rather, we
are proposing a parallel distributed way of integrating different modules. It is well known in neuroanatomy that if one area is connected to another area by a feedforward connection, backward or feedback connections always exist. Our main proposal is that the downward information flow implemented by the feedforward connection constitutes an approximate inverse model of some physical process outside the brain such as the kinematic transformations, the dynamic transformations and the optics, that is the image generation process. On the other hand, the upward information flow implemented by the feedback connections provides a forward model of the corresponding physical process.

What are the advantages of this bi-directional architecture? The first advantage is its fast computation. The cascade of inverse models gives a feedforward and one-shot calculation. This can execute reflexes and fixed action patterns triggered by specific stimuli. However, if we do not have forward models, and if we must rely solely on computational machinery provided by the unidirectional theory, our behavior repertoire should be very limited. Equifinality, optimality, or adaptability to different environmental situations can be realized only if the brain uses some kind of internal forward models, in other words an emulator or predictor of external events. The network relaxation in the circle of the inverse and forward models converges very rapidly, within a few iterations, to a suboptimal solution. This is the second computational advantage. Third, integration of parallel modules can be done rapidly. Thus, bi-directional theories may provide an understanding about how the large number of different visual cortical areas can nevertheless produce a coherent percept of the visual world. Finally, the bi-directional theory might give a concrete computational algorithm for the motor theory of movement pattern perception.

In the motor-theory of speech perception (Liberman et al., 1967; Liberman & Mattingly, 1985), a neural network for motor control is hypothesized to play an essential role in the perception of speech. Our algorithm gives one specific computational realization of this psychological theory. Data resulting from human movement, either visual data (e.g., handwriting, biological motion) or auditory data (e.g., speech) are very severely constrained by the dynamics of the controlled objects and interactions with the external world as well as the motor control strategy adopted by the central nervous system. Thus, any efficient motor-pattern perception scheme must either implicitly or explicitly take account of these physical and physiological constraints. Here, we advocate a rather radical stand-point by stating that the movement-pattern generation network actively participates in movement-pattern perception in a dualistic way. The proposed theory of movement-pattern perception based on dynamic optimization can be used for movement pattern recognition (handwriting or speech), telecommunication and teleoperation, and learning-by-imitation in robotics.

In the bi-directional theory framework, we have two opposite flows in the hardware, thus the same algorithm can be applied equally to motor control as well as to perception of movement patterns, as shown in Figure 4.
3. LEARNING BY WATCHING

Table 2 describes different strategies for learning by watching. If the learner had perfect intelligence, s/he would be able to understand the will or motor intention of the teacher from perceiving the teacher’s demonstration. Subsequently, this information could be translated into a stream of actual motor commands by taking into account the laws of physics describing the properties and environment of the teacher and learner.

This strategy can be conceived as the highest level, the most “cognitive” way of learning by watching. In contrast, if one wanted to avoid this high level of intelligence, one could employ a strategy solely based on imitating the position and force trajectory demonstrated by the teacher with accuracy. Indeed, such an indiscriminate imitation could be an efficient strategy if the following conditions, at least, were satisfied. (1) The teacher and the learner have exactly the same kinematic and dynamic properties. (2) The teacher and the learner have exactly the same environment. (3) The learner has extremely precise measurement and control ability. However, these requirements are not satisfied in almost all realistic situations. Hence, a more abstract understanding of the teacher’s motor behaviors at a higher level is essential.

Learning algorithms, such as reinforcement learning and genetic algorithms, can be efficiently used for task-level learning if adequate representations of the task are selected. However, this selection is one of the most difficult and critical parts of motor learning. Assuming the pre-existence of proper representations amounts to having solved a major part of the problem a priori and fails to deal with the full problem of motor learning (Schaal et al., 1992). Good representations have to fulfill a variety of prerequisites. The representations should take into account the dynamics of the controlled object and the external world as well as computational principles adopted by the central nervous system in motor control.

In this paper we want to demonstrate that via-point representations, extracted from movement trajectories based on a dynamic optimization principle, are an attractive candidate for a generic representation in motor control.

### 3.1. Via-points Extraction and Trajectory Formation

Using FIRM, Wada and Kawato (1995) developed an algorithm to extract approximately the minimum number of via-points \( S = \{P_1, P_2, \ldots, P_N\} \) from a given trajectory \( X_{\text{data}} \) with some level of error threshold \( \delta \). The via-point extraction problem is to find the minimum number \( N \) such that the reconstructed trajectory \( \tilde{X}_{\text{opt}} \) does not deviate from the original trajectory more than the threshold:

\[
\|X_{\text{data}} - \tilde{X}_{\text{opt}}(S)\| < \delta.
\]

Note that this problem is a nonlinear optimization problem. If the error threshold \( \delta \) is satisfied for the minimum \( N \), then the problem is to find the via-point set \( \mathcal{S} \) that gives the minimum error level \( \|X_{\text{data}} - \tilde{X}_{\text{opt}}(\mathcal{S})\| \). If a fixed number of via-points is given and the arm dynamics is known as

\[
\frac{dX}{dt} = f(X, M),
\]

where \( M \) is the motor command, FIRM can calculate the optimal trajectory \( \tilde{X}_{\text{opt}}(S) \) passing through these via-points in a given time \( t_f \) and minimizing the following equation

\[
\int_0^{t_f} \left\| \frac{dM}{dt} \right\|^2 dt.
\]

Figure 5 illustrates how the algorithm for extracting via-points works using an example of Kendama. Based on the FIRM model, the via-points are extracted sequentially as follows: (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 (fifth order polynomial). (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.

3.2. Application to Robot Learning

We want to demonstrate the usefulness of the via-point representation for learning by watching in a robot learning example. The robot is the SARCOS dextrous slave arm (Figure 6) which has almost the same kinematic structure (seven degrees-of-freedom) as a human arm. However, there are two major differences in comparison to human subjects. First, the kinematic parameters of the robot arm are different, particularly due to differences in limb length. Second, the robot’s dynamic parameters, such as mass and inertia, are very different from the human arm’s (about 4–5 times heavier). Its mechanical joints and hydraulic actuators result in quite different compliance properties for the robot compared with those for human arms. Thus, even if the robot arm were able to follow exactly the same trajectory in Cartesian space as shown by a human teacher, it could not exert the same force when interacting with the environment, and, therefore, could not perform the task successfully. For manipulation tasks, force control is one of the most critical factors. This offers an interesting challenge for learning by watching since forces, of course, cannot be extracted from recording kinematic data.

The SARCOS arm is controlled by joint angle command. Because of the redundancy inherent in a seven degree-of-freedom kinematic structure, there are an infinite number of joint angles even if the hand position and orientation in Cartesian space are determined.

The above-mentioned difficulties can be solved with task-level learning using via-points as follows. Let $X_{data}$ and $\theta_{data}$ denote the trajectory of human movement (hand position and orientation in Cartesian space, and joint angles). Let $X_{via}(i = 1, 2, \ldots, N)$ denote the six-dimensional vectors (hand position and orientation in Cartesian space) that express via-points extracted from human movement and $\theta_{via}(i = 1, 2, \ldots, N)$ denote the seven dimensional vectors that express the same via-points in joint angle space. Let $X_{via}(i = 1, 2, \ldots, N)$ and
\( \dot{\theta}_{\text{via}}^i (i = 1, 2, \ldots, N) \) denote the via-points of the SARCOS arm. In general, variables without tilde denote those for the human demonstrator, while with tilde for the SARCOS robot. We require that robot via-points represented in Cartesian space are exactly equal to humans \( (\dot{X}_{\text{via}}^i = X_{\text{via}}^i) \), and robot via-points represented in joint angle coordinates are closest to humans while satisfying the above conditions strictly.

Let us mathematically formulate this below.

Let \( \ddot{L} \) denote the forward kinematics equation of the SARCOS arm. There are an infinite number of joint angles of the robot arm which satisfies \( \dot{X}_{\text{via}} = \ddot{L}(\dot{\theta}_{\text{via}}) = X_{\text{via}} \) because of redundancy in the SARCOS arm. As a solution, we can choose \( \dot{\theta}_{\text{via}} \) which satisfies the above condition and minimizes \( \| \dot{\theta}_{\text{via}} - \dot{\theta}_{\text{via}}^i \| \). By means of this optimization principle the robot joint angles are determined as close as possible to those of the human subject. The desired trajectory in the joint angle coordinate of the SARCOS arm passing through the via-points \( \dot{\theta}_{\text{via}}^i \) is obtained as the optimal trajectory \( \dot{\theta}_{\text{opt}} \) based on the smoothness principle.

Let us express the sets of the via-points of the human arm and SARCOS arm, respectively, as follows:

\[
\mathcal{S}_{\text{teacher}} = \{ \theta_{\text{via}}^1, \theta_{\text{via}}^2, \ldots, \theta_{\text{via}}^N \} \\
\mathcal{S}_{\text{learner}} = \{ \dot{\theta}_{\text{via}}^1, \dot{\theta}_{\text{via}}^2, \ldots, \dot{\theta}_{\text{via}}^N \}.
\]

Let \( \mathcal{E}, H, \) and \( J \) denote via-point extraction algorithm, the via-point determination algorithm, and optimal trajectory calculation, respectively. We get the following equations

\[
\mathcal{E}_{\text{teacher}} = E(\theta_{\text{data}}), \quad (6)
\]
\[
\mathcal{E}_{\text{learner}} = H(\mathcal{E}_{\text{teacher}}), \quad (7)
\]
\[
\dot{\theta}_{\text{opt}} = J(\mathcal{E}_{\text{learner}}). \quad (8)
\]

The realized trajectory executed by the SARCOS arm is expressed as follows:

\[
\dot{\theta}_{\text{real}} = G(\dot{\theta}_{\text{opt}}). \quad (9)
\]

Here \( G \) is nearly equal to an identity map if adequate feedforward and feedback controller are used. From the above equations, we obtain

\[
\dot{\theta}_{\text{real}} = G(\dot{\theta}_{\text{opt}}) = G(J(H(E(\theta_{\text{data}})))) = \Theta(\theta_{\text{data}}). \quad (10)
\]

The composite function \( \Theta \) is not so different from the
identity map, if the following two conditions are satisfied: (1) the optimization principle adopted here is not so different from the human’s and (2) representation of the via-points is appropriate. Eventually, the realized trajectory \(\theta_{\text{real}}\) is smooth and parameterized by a small number of control variables \(\mathcal{S}_{\text{learner}}\). Resultant realized movement by the robot arm is close to the demonstrated movement of the human subject. From this, it is only a step to the successful execution of the task.

Let \(T\) be the task target which estimates the success or failure of the task directly. For example, in the case of the Kendama task, the task variables are horizontal discrepancies (along the \(x\) and \(y\) axes; see the caption of Figure 5 for the coordinate system) between the ball and the cup when the ball falls to the same height as the cup (detailed description will be given in the next section). Because we can determine \(T\) uniquely from \(\theta_{\text{real}}\), the task variable is represented as \(T = F(\mathcal{S}_{\text{learner}})\) by means of eqn (10).

In order to make the realized task \(T\) agree with the desired task \(T_{\text{desired}}\), by means of the representation \(\mathcal{S}_{\text{learner}}\) we can use various learning schemes such as feedback-error-learning, forward-inverse-modeling, direct-inverse-modeling, genetic algorithm, and reinforcement learning.

We can summarize our teaching method generally as follows. (1) Teaching information via a human demonstration, (2) perceiving movement patterns based on the dynamics of the controlled object and an optimization principle, (3) extracting the via-points which reconstruct the movement pattern by means of a neural network model, (4) treating the via-points as the control variable with fixed trajectory planning and a control scheme, and (5) modifying spatial and temporal positions of the via-points so that the desired task will be realized successfully.

4. ROBOT EXPERIMENT

In this section, we demonstrate the performance of the Kendama learning robot in order to examine the potential of our method.

We chose the game of Kendama as a task which allows for a simple task-level representation. The Japanese toy Kendama consists of two parts connected by a thin string: a ball and a handle equipped with three cups of different sizes. There are several way to play Kendama. For simplicity, we consider the second-easiest one. In the initial condition, a player holds the handle with the ball hanging down. The player swings the handle up, yanking the ball to fly over the handle. After a few hundred milliseconds of flight, the ball is caught in the middle-size cup. In order to play Kendama, quick and dynamic motion is required. A robotic arm which has complex, uncompensated link interferences could not perform the highly precise trajectory following.

4.1. Overview of Kendama Learning Robot

If the SARCOS arm yanks the ball properly, the ball will fall onto the cup. However, the robot arm cannot reproduce the human Kendama task exactly, because of the difference in dynamic and kinematic properties between the human objects and the robot. Moreover, as shown in Figure 8 (later), some error remains even if the inverse-dynamics model is used for feedforward control with a conventional feedback control. Although the human subject grasps the handle of the Kendama softly, we attached the handle rigidly to the fixed forefinger of the SARCOS arm with a small vise. Due to the difference in the grasping of the handle, the ball will show a behavior quite unlike that of the human demonstration even if the SARCOS arm performs perfect trajectory following. The diameters of the ball and the cup are about 58 and about 33 mm, respectively. The length of the string is about 395 mm.

Figure 7 illustrates a schematic diagram of the Kendama learning experiment based on the mathematical and abstract formulation given in Section 3.2. We made the Kendama learning according to the following procedure: (1) We measured the human Kendama execution using a three-dimensional vision system (OPTOTRAK). We extracted Cartesian via-points \(X_{\text{via}}\) from a human demonstration. We relied on the joint angle via-points \(\theta_{\text{via}}\) of the human demonstration for the coordinate transformation from the Cartesian to the joint space of the SARCOS arm via-points \((X_{\text{via}} \rightarrow \theta_{\text{via}})\).

\(X_{\text{via}}\) were used as the initial state of the robot’s Cartesian via-points \(\bar{X}_{\text{via}}\). (2) The robot’s Cartesian via-points \(\bar{X}_{\text{via}}\) were transformed into the robot’s configuration space (joint angle \(\bar{\theta}_{\text{via}}\)) (see Sections 3.2 and 4.3). (3) The optimized trajectory \(\bar{\theta}_{\text{opt}}\) was generated as the desired trajectory of the SARCOS arm. (4) The Kendama task was executed by the SARCOS arm. The ball and the cup position trajectories \((X_{\text{ball}} \text{ and } X_{\text{cup}}\) respectively) were measured by a three-dimensional visual-sensing system (QUICKMAG). (5) To make the SARCOS arm yank the ball up properly and drop it onto the cup, \(\bar{X}_{\text{via}}\) were modified. Steps (2)–(5) were repeated until the Kendama task was executed successfully. Detailed descriptions of this procedure will be given in the following subsections [Section 4.2 for (1), Section 4.3 for (2) and (3), Section 4.4 for (4), and Section 4.5 for (5)].
4.2. Measurement of Human Demonstration

While a subject was playing Kendama, we measured the positions of the shoulder, elbow, wrist, and back of the human right hand by a three-dimensional vision system (OPTOTRAK: infrared light-emitting diode markers and three cameras, the precision of the position sensing is about a few millimeters in the current experimental setting). We aligned the position data so that the time at the peak hand height (maximum of z) was in the middle of a total movement time (2 s) (see also the caption of Figure 5). From these position data, we calculated the Cartesian hand positions and orientations $X_{\text{data}}$, which is a six-dimensional vector, and the trajectories of the seven joints’ angular positions $\theta_{\text{data}}$, a seven-dimensional vector. We extracted Cartesian via-points $X_{\text{via}}^i (i = 1, \ldots, N)$ from $X_{\text{data}}$ using the FIRM algorithm. For simplicity, we fixed the number of via-points $N$ at seven for the real robot experiment. Then we extracted joint angular via-points $\theta_{\text{via}}^i (i = 1, \ldots, N)$ from $\theta_{\text{data}}$ at the same temporal positions as $X_{\text{via}}^i$.

Figure 8 shows the via-points and trajectories of the human and SARCOS arm. In the figure, the desired movement of the SARCOS arm, and realized movement of the SARCOS arm, respectively. A cross and a circle indicate the via-point of the human arm and the SARCOS arm, respectively. Because of the difference in kinematic properties (e.g., link length) between the human subject and the robot, the joint trajectories of the SARCOS arm are much different from that of the human subject. As shown in Figure 8, the via-points extracted using the FIRM algorithm seem to be assigned at the point that represents the essential points of the movement, e.g., around the beginning of the upward motion of the hand (around the second, third, and fourth via-points. See third panel of right side in Figure 8).

In Figure 8, joint trajectories of the human and SARCOS arm show large differences, especially near $t = 0$. If we give the human joint trajectory to the SARCOS arm, the hand trajectory of the robot is different from the human one. It is caused by the differences in the kinematic parameter (e.g., link length). Conversely, if we adjust the hand trajectory in Cartesian space, the joint trajectory of the robot can be chosen arbitrarily because the robot and human arm have the redundancy (seven degrees of freedom). To determine the proper joint trajectory of the SARCOS arm, we describe the method for transforming the human movement to the robot’s in the following section.

4.3. Transforming the Human Movement to that for the Robot

As already explained in Section 3.2, when transforming the Cartesian via-points to joint space, the Cartesian coordinates are given as hard constraints which must be strictly satisfied, while the soft constraint is that the robot joint angles have to be as close as possible to those of the human subject. This can be accomplished using the following Newton-like method:

$$a^{-1} \dot{\theta}_{\text{via}}^i = a \dot{\theta}_{\text{via}}^i + J^i (\dot{a} \dot{\theta}_{\text{via}}) \{X_{\text{via}}^i - \hat{L}(\dot{a} \dot{\theta}_{\text{via}})\}, \quad (11)$$

where $a \dot{\theta}_{\text{via}}^i$ is the $i$th via-point of the SARCOS arm represented as the joint angle at the $n$th iteration of the Newton-like method. We adopted the “singularity low-sensitive motion resolution matrix” $J^i (\theta) = (\partial \hat{L}(\theta)/\partial \theta)^T$ proposed by Nakamura and Hanafusa (1984) as a generalized inverse matrix of the Jacobian. This matrix minimizes
\[ \|(I - J J^T)\|^2 + k \|J^T\|^2. \]

The first term of this performance index requires that the inverse transformation is mathematically exact, and the second term requires that the joint angles do not move too much. \(k\) is a weighting ratio of these two requirements. The concrete formula of \(J^T\) is given as follows.

\[ J^T = (J^T J + kI)^{-1} J^T. \] (12)

Here, \(J^T\) denotes the transpose of matrix \(J\). \(k\) is varied.
where the torque generated by the $j$th actuator $P^j$ and $\dot{\theta}^j$ are the inertia of the link and the acceleration of the $j$th joint angle, respectively. The first method is based on the spline function. In this experiment, we adopted the second method (see Wada & Kawato, 1995 for details). Using the second method, the trajectory passing through a via-point is produced sequentially. Using the minimum principle for the approximated linear dynamics such as eqn (14), FIRM can calculate the optimal trajectory $\theta_{opt}$.

4.4. Definition of Task Variable

By using the desired joint trajectory obtained in Subsection 4.3, that is, by simply imitating the human Kendama motion, the SARCOS arm fails to perform the Kendama task as shown in Figure 9. To make the SARCOS arm successfully execute the task, we adopted the concept of task-level learning. Aboaf et al. (1988) demonstrated that learning can proceed at the task-level, even though lower level modules do not perform perfectly. In our case, the control variables are via-points. We want to adjust the ball throwing so that the ball falls exactly onto the cup.

Let us first explain the more intuitively straightforward definition of the task-variable, then propose a more practical modification of that. We may choose the realized and the desired task representations as follows.

$$T^{\text{real}} = \begin{pmatrix} x_b^b \\ y_b^b \\ z_b^b \end{pmatrix}, \quad T^{\text{desired}} = \begin{pmatrix} x_h^b \\ y_h^b \\ z_h^b \end{pmatrix}. \tag{15}$$

Here, $(x_h^b, y_h^b, z_h^b)^T$ and $(x_b^b, y_b^b, z_b^b)^T$ represent the ball and the cup positions respectively in Cartesian coordinates when the ball falls to the same height as the cup $(z_h^b = z_b^b)$. $(x$ positive rightward, $y$ positive anterior, and $z$ positive upward). Figure 10 illustrates the definition of the variables used for the task representation. However, this task representation cannot determine the height of the ball flight. Moreover, there is a possibility that the vision system cannot measure the ball position correctly when the robot arm or the handle of the Kendama occludes the view of the ball. Then we choose the following representation as a more practical choice, which is essentially the same as the one above.

The task variables are the vertical and horizontal distance between the initial and the peak positions of the ball:

$$T^{\text{prac}} = \begin{pmatrix} x_p - x_b^b \\ y_p - y_b^b \\ z_p - z_b^b \end{pmatrix}, \tag{16}$$

where $(x_b^b, y_b^b, z_b^b)^T$ and $(x_p, y_p, z_p)^T$ represent the initial and the peak positions of the ball in Cartesian coordinates, respectively. Thus, $T^{\text{prac}}$ represents the deviation from the initial position to the peak position of the ball. Since our aim is that the ball falls onto the cup, we have to control the peak ball position so that the horizontal ball position exactly matches the horizontal cup position when the ball falls to the same height as the cup. After the ball begins to fall freely, the $x$ and $y$ components of the ball velocity are constant. Using linear interpolation, the $x$ and $y$ coordinates of the desired peak ball position are derived from the cup position when the ball falls to the same height as the cup (see Figure 10). Therefore, we set the desired task representation as follows:

$$T^{\text{prac}}_{\text{desired}} = \begin{pmatrix} x_p - x_b^b \\ y_p - y_b^b \\ \text{const.} \end{pmatrix}, \tag{17}$$

$x_p = x_b^b + (x_b^b - x_f^b)r_t.$ \tag{18}
\[ y_p^b = y_f^b + \left( y_h^b - y_f^b \right) r_t \]  \hspace{1cm} (19)

\[ r_t = \frac{t_p - t_f}{t_h - t_f}, \]  \hspace{1cm} (20)

where \( x_f^b \) and \( y_h^b \) represent the cup position when the ball falls to the same height as the cup, \( x_f^b \) and \( y_f^b \) represent the horizontal ball position when the ball is just beginning to fall freely, \( t_f \), \( t_p \), and \( t_h \) represent the time when the ball begins to fall freely, reaches the peak position, and falls to the same height as the cup, respectively. Since the ball shows straight locus in the \( x \) and \( y \) coordinates while the ball is falling freely, we can use the linear interpolation. We set the \( z \) component of \( \mathbf{T}_{\text{desired}} \) at a constant value so that the ball rises to a proper height (in this experiment, we set the desired peak height of the ball to range from about 0.5 to 0.8 m).

4.5. Modification of Via-points through Visuo-motor Learning

To improve the robot's performance, the via-point locations were modified by the following Newton-like
method. This learning scheme can be regarded as an extension of the task-level learning algorithm proposed by Aboaf et al. (1988), in the sense that the control variables are not necessarily the task target but are as abstract as the task target.

\[
\tilde{\mathcal{F}}_{n-1} = \tilde{\mathcal{F}}_n = \left( \frac{\partial T_n}{\partial \tilde{\mathcal{F}}_n} \right)^* B T_{\text{desired}} = T_n.
\]

Here, \(\tilde{\mathcal{F}}_n\) denotes the Cartesian via-points of the SARCOS arm in \(n\)th iteration of the Newton-like method, \(T_{\text{desired}}\), and \(T_n\) are the desired and realized task representations, respectively. We used the matrix \(\left( \frac{\partial T_n}{\partial \tilde{\mathcal{F}}_n} \right)^* = A^T \Lambda + kI)^{-1} A^T\), where \(A^T\) denotes the transpose of matrix \(A\). This matrix is the simply regularized \(g\)-inverse matrix (Rao & Mitra, 1971; Okamoto & Musha, 1992) which minimizes \(||I - AA^T||^2 + k||A^T||^2\). The first term of this performance index requires that the inverse transformation is mathematically exact, and the second term requires that the via-points do not move too much. \(k\) is a weighting ratio of these two requirements. In this experiment, we set \(k = 1\), because we wanted to make the SARCOS arm accomplish the task as close as possible to the human demonstration and also to modify the via-point as small as possible for safe execution of the SARCOS arm movement.

Because it is difficult to analytically compute the matrix \(\frac{\partial T}{\partial \tilde{\mathcal{F}}}\), we used the matrix \(\Delta Y_{\text{real}}/\Delta Y_{\text{via}}\) for \(\frac{\partial T}{\partial \tilde{\mathcal{F}}}\). To estimate the matrix, we observed the changes in the behavior of the Kendama task \(\Delta T_{\text{real}}\) when the perturbation \(\Delta Y_{\text{via}}\) is added on the via-points. In this experiment, we added \(\pm \Delta Y_{\text{via}}\) (0.01 m in magnitude) changes only to the three-dimensional position \((x, y, \text{and} z)\) of the via-points. Thus, observation of the perturbed behaviors of the Kendama task totalled \(3 \times 2 \times 7 = 42\).

Since the amplitude of the Kendama movement along the vertical direction is larger than that along the horizontal direction, a weighting constant had to be introduced to balance the importance of the individual components in the update equations. This is accomplished by the diagonal matrix \(B = \text{diag}(1, 1, 0.2)\). Because the sixth and the seventh via-points are around the period of catching the ball, the modification of these via-points made the learning too hard. Hence we did not modify the sixth and the seventh via-points.

4.6. Results

During the robot’s task execution, the ball and the cup positions were measured by a three-dimensional visual sensing system (QUICKMAG: tracking color blobs with two cameras). This system can sample the center of a blob of a specific color at the sampling rate of 60 Hz, and the precision of the position sensing is about 5–10 mm in the current experimental setting. The positions of the ball and the cup were fed to the SARCOS controller via an 8-bit parallel port at the 500 Hz sampling rate. After the execution of the task, the recorded ball and cup trajectories were smoothed using a Butterworth second order digital low-pass filter with a low-pass cut-off of 5 Hz. From these trajectories the distance between the initial and the peak position of the ball, \(T\), was calculated.

After seven learning cycles, as shown in Figure 11, the SARCOS arm executed Kendama successfully. We tried several executions with this desired trajectory, and got 2–3 successful executions out of ten.

5. NUMERICAL SIMULATION

As mentioned in Subsection 4.2, the FIRM algorithm extracts via-points at a time which corresponds to segmentation of the Kendama movement into several essential motor primitives. These motor primitives and corresponding segmentation should be common features irrespective of differences among individual human demonstrators. To confirm this expectation, we controlled the simulated Kendama robot using several trials from different human demonstrators. Furthermore, in order to explore the validity of the via-point extraction of the FIRM algorithm, we altered the conditions of the via-point extraction. Since the real robot shows dangerous behavior at an eccentric trajectory, and since the real robot experiment takes much time, we used a numerical model of the robot to examine the Kendama task with several via-point extraction methods.

Figure 12 shows a schematic diagram of the Kendama learning simulation. The system is almost the same as the Kendama learning experiment (Figure 7), except for the robot arm and the Kendama equipment.

5.1. Analytical Model

In order to simulate the robot arm, we make use of analytical models of the forward dynamics, inverse dynamics, and forward kinematics of the SARCOS dextrous slave arm. The dynamics models were derived in a recursive Newton Euler formulation by adopting the spatial vector arithmetic approach of Featherstone (1987). The forward kinematics are obtained as a by-product of these calculations. In the simulation, the forward dynamics are numerically integrated by means of Euler’s method with a time step of 0.5 ms. The inverse dynamics equations are used for feedforward control together with a conventional PD controller. It should be noted that despite the fact that we have perfect models for
FIGURE 11. The seventh trial of the Kendama task executed by the SARCOS arm. See Figure 9 for legend description.

FIGURE 12. A schematic diagram of Kendama learning simulation. Only the part different from Figure 7 is shown.

simulation, the coarse Euler integration of the forward model as well as the finite servo rate of our controller (500 Hz) still introduce some differences between the desired trajectories and the actually realized ones, particularly for fast movements, such as those as necessary for Kendama.

In order to simulate the ball behavior, we used the following simplified Kendama dynamics model:

\[
\begin{align*}
\|F\| &= \begin{cases} 
K(l - l_0) & \text{if } l \geq l_0, \\
0 & \text{otherwise}, 
\end{cases} 
\end{align*}
\]

\[
l = \sqrt{(x_h - x)^2 + (y_h - y)^2 + (z_h - z)^2},
\]

\[
m\ddot{x} = -f_x = -\|F\|(x_h - x)/l, \quad (24)
\]

\[
m\ddot{y} = -f_y = -\|F\|(y_h - y)/l, \quad (25)
\]

\[
m\ddot{z} = -f_z - mg = -\|F\|(z_h - z)/l - mg, \quad (26)
\]

where \(K\) represents the elastic constant of the string, \(F\) represents the force vector which acts upon the string, \(l\) and \(l_0\) represent the current length and the natural length of the string, respectively, \(x_h, y_h\) and \(z_h\) represent the Cartesian position of the origin of the string at the handle, \(m\) and \(g\) represent the mass of the ball and acceleration by gravity, respectively, \(x, y,\) and \(z\) represent the Cartesian position of the ball. The double dots represent the second order time derivative. These equations were integrated numerically by means of Euler's method with a time step of 2 ms. For simplicity, we considered the ball as a point mass and the cup as a hemispherical inner surface (the equation of the interaction between the cup and the ball is not shown). Although the behavior of this simplified Kendama model is slightly different from
the behavior of the real Kendama, and better approximate models were developed (e.g., Fujii et al., 1993), the qualitative behavior of our simulated Kendama is good enough for our purpose. The behavior of the ball is qualitatively similar to the real one as shown in the next subsection.

5.2. Using a Few Instances of Human Demonstration

In this simulation, we used three sets of human demonstration data. The measurement was performed as described in Subsection 4.2. Two righthanded subjects (subjects A and C) and one lefthanded subject (subject B) participated in this measurement. We asked subject B to perform Kendama with the right hand, since our robot has only a right arm.

The data of subject A were exactly the same as the data used in Section 4. Figures 9 and 11 show the executions by the SARCOS arm before and after learning, respectively. Figures 13 and 14 show the simulated robot’s performance based on data demonstrated by subject A. Comparing the two pairs of figures (Figures 9, 13 and Figures 11, 14), the simulated Kendama shows quite similar behavior to the real Kendama executed by the real robot.

Figures 15 and 16 show the simulated execution after five learning cycles using data demonstrated by subject B and subject C, respectively. The task performance was successful when using the data of subject B, as well as when using the data of subject A. The simulated robot, however, failed when using the data of subject C.

In Figures 14, 15, and 16, it can be seen that the via-points extracted by the FIRM algorithm are assigned at similar locations among the individual demonstrators. This suggests that the FIRM algorithm extracts a characteristic common to all the demonstrated movements as a representation of segmentation of movements into the essential parts. As shown in Figure 16, using the data of subject C, although the task target (the ball falling into the cup) was accomplished almost perfectly, the ball dropped from the cup. Note that, as shown in the figure, there was a considerably large error between the desired trajectory and the realized trajectory especially in the latter half of the movement. In order to examine whether the imperfection of the control mainly caused the failure, we used an ideal robot instead of the dynamics model. Using the ideal robot whose realized joint trajectory is identical to the desired one, the Kendama was executed successfully. Thus, we concluded that the movement of subject C was too quick for the simulated SARCOS arm (also for the real robot) to follow the desired trajectory with small error. This coincides with the observation that when

![Figure 13. Simulated trajectories of the Kendama task before learning, in the case of subject A. (Left) Time courses of the desired cup (thin line), realized cup (dotted line), and the ball (thick line) trajectories, respectively. A cross and a circle show the via-points of the human arm and the SARCOS arm, respectively. (Right) See Figure 9 for legend description.](image-url)
FIGURE 14. Simulated trajectories of the Kendama task after five learning cycles, in the case of subject A. See Figure 13 for legend description.

FIGURE 15. Simulated trajectories of the Kendama task after five learning cycles, in the case of subject B. See Figure 13 for legend description.
using the data demonstrated by subject C, joint limits were hit and torque commands were saturated. Using data of a trial in which subject C was asked to perform Kendama as gently as possible, the Kendama could be executed successfully by the simulated SARCOS arm.

5.3. Changing the Number of Via-points

In Section 4, we fixed the number of via-points at 7 for simplicity. In this subsection, we changed the number of via-points and investigated how the number of via-points affects the task.

Figure 17 (left) shows the case of 4 via-points. The rising ball was intercepted by the hand, because the avoiding motion of the hand was not reproduced sufficiently. In the case of the number of via-points ranging from 5 to 9, the Kendama task was successfully executed within 5 learning cycles. In the case of 10 via-points, the Kendama task was successfully executed, too, but ten learning cycles were needed.

Figure 17 (right) shows the case of 11 via-points. As shown in the figure, the desired trajectory was corrugated and the error of the realized trajectory was considerably large.

An adequate number of via-points is the minimum number of via-points needed to represent the task. A smaller number of via-points will increase the error between the human trajectory and the reconstructed trajectory, because of the insufficiency of the information for the movement. In contrast, an excessive number of via-points overfits the trajectory: as the number of via-points increases over the optimal, trivial points which are nonessential for the task will be selected. Such unnecessary via-points will introduce undesired effects and cause difficulties in control.

In the case of 11 via-points [Figure 17 (right)], the desired trajectory was corrugated and the error of the realized trajectory is considerably large. We suspect that this notched desired trajectory is caused by the FIRM algorithm. The interval time between a via-point and its adjoining via-point becomes too short if the FIRM algorithm extracts an excessive number of via-points. Consequently the desired joint trajectory tends to be bumpy since the trajectory formation is based on the fifth order polynomial (see Subsection 3.1). If the desired trajectory is not smooth, acceleration changes rapidly and quite a large torque is required to follow the desired trajectory. As the learning cycles are repeated, the degree of notching of the desired trajectory increases. This undesirable modification is probably a result of the imprecise estimation of the matrix $\Delta T_{\text{real}}/\Delta X_{\text{via}}$ for $\partial T/\partial S$ in eqn (21). It should be noted that using an ideal robot (whose realized joint trajectory is identical with the desired one), in the case of 11 via-points, the
Kendama was executed successfully even when the desired trajectory was rugged.

5.4. Selecting Via-points at Equi-time Intervals

When the FIRM algorithm extracts the via-points, FIRM selects the via-point at the point of maximal squared error between the given trajectory and the generated trajectory, as described in Subsection 3.1. In this subsection, via-points were distributed uniformly at equal temporal intervals in order to examine the validity of the FIRM algorithm.

Figure 18 (left) shows the case of 7 via-points at equi-time intervals. The hand cannot avoid the rising ball. Figure 18 (right) shows the case of 9 via-points at equi-time intervals. In this case the Kendama task was successfully executed. Note that more via-points are required when the via-points are selected at equi-time intervals.

We aligned the position data so that the time at the peak hand height (maximum of z) was in the middle of the total movement time (2 s). When we chose an odd number of via-points, the middle via-point is assigned at a time when the hand is just in the position at peak height. In the present way of playing Kendama, the major constituents of hand movement are an upward motion to yank up the ball, a rightward motion to avoid the rising ball, and a leftward motion to catch the falling ball. In the case of 9 via-points at equi-time intervals, the via-points happen to be located around these points.

5.5. An Index of Manipulation of Task Learning

We compared an index of ability of manipulation of task learning under various conditions. Here, we consider the following scalar value as the index of ability

\[ w = \sqrt{\det AA^T}, \]

where \( A = \Delta T_{\text{real}}/\Delta X_{\text{via}} \). \( A^T \) denotes the transpose of matrix \( A \). Figure 19 shows the ability under various conditions.

In the case of 3 via-points with the FIRM algorithm, the ability is very low. The reconstructed trajectory cannot reproduce the trajectory demonstrated by the human; even if the via-points are modified, it is very difficult to control the behavior of the ball. Moreover, an excessively small value of \( w \) causes inaccuracy of the inverse-transformation in eqn (11). In the case of 4 via-points with the FIRM algorithm, the ability is sufficiently high. Although the task target (yanking the ball up properly was accomplished, the motion to avoid the rising ball (rightward: x positive direction) was not reproduced and the simulated robot failed Kendama [see Figure
In the case of 5–10 via-points with the FIRM algorithm, the ability is sufficiently high, resulting in successful execution. In the case of 11 via-points with the FIRM algorithm, even though the ability was sufficiently high, the desired trajectory was corrugated as learning progressed.

When we chose an even number of via-points at equi-time intervals, the ability was too small because of the reason mentioned in Subsection 5.4. In the case of 5 and 7 via-points at equi-time intervals, the motion to avoid the rising ball was not reproduced and the simulated robot failed Kendama, although the ability was sufficiently high. In the case of 9 and 11 via-points (11 case is not shown) at equi-time intervals, the ability was sufficiently high and the simulated robot executed Kendama successfully, because, as mentioned previously, the via-points happen to be located around the major constituents of hand movement.

Generally speaking, the ability is greater with the FIRM algorithm than with the equi-time intervals, as shown in Figure 19. It is conceivable that an adequate
number of via-points extracted by the FIRM algorithm is a good representation for the human movement pattern in the case of Kendama.

6. CONCLUSION

We demonstrated that a general theory of movement pattern perception based on a dynamic optimization theory can be used for motion capture and learning by watching in robotics. In the present experiment, we chose the toy task Kendama to demonstrate the potential of our methods. In the future, we are planning to extend our studies to even more complicated tasks, such as a tennis serve.

So far, we have not made full use of visual feedback to improve the performance of our robot. Visual feedback was only employed to obtain an error measurement after an open-loop execution of the task. Clearly, when humans perform Kendama, they modify their hand positions during the flight of the ball based on visual information, which improves their performance significantly. How does biology coordinate such modifications? Reaching movements of monkeys and humans have been studied in the past decade (e.g., Georgopoulos et al., 1981). Several theoretical models were proposed to account for the modification of aimed movement in response to a displacement of the target (e.g., Massone & Bizzi, 1989; Flash & Henis, 1991; Flanagan et al., 1993; Hoff & Arbib, 1993). Before adding visual feedback control to our system, more work will be necessary to gain a better understanding of the mechanisms at work in visuo-motor coordination.

REFERENCES


