Hidden Markov Model Approach to Skill Learning and Its Application to Telerobotics

Jie Yang[1] Yangsheng Xu C.S. Chen[2]

CMU-RI-TR-93-01

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

January 1993

DTIC
ELECTE
S JUL13 1993
E **D**

DISTRIBUTION STATEMENT
Approved for public release
Distribution Unlimited

---

[1]Visiting Scientist from The University of Akron.
[2]Department of Electrical Engineering, The University of Akron.

REPORT DOCUMENTATION PAGE

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	January 1993	technical

4. TITLE AND SUBTITLE

Hidden Markov Model Approach to Skill Learning and Its Application to Telerobotics

5. FUNDING NUMBERS

6. AUTHOR(S)

Jie Yang, Yangsheng Xu, and C.S. Chen

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

8. PERFORMING ORGANIZATION REPORT NUMBER

CMU-RI-TR-93-01

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release;
Distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

In this paper, we discuss the problem of how human skill can be represented as a parametric model using a hidden Markov model (HMM), and how a HMM-based skill model can be used to learn human skill. HMM is feasible to characterize two stochastic processes—measurable action and immeasurable mental states—which are involved in the skill learning. We formulated the learning problem as a multi-dimensional HMM and developed a programming system which serves as a skill learning testbed for a variety of applications. Based on "the most likely performance" criterion, we can select the best action sequence from all previously measured action data by modeling the skill as HMM. This selection process can be updated in real-time by feeding new action data and modifying HMM parameters. We address the implementation of the proposed method in a teleoperation-controlled space robot. An operator specifies the control command by a hand controller for the task of exchanging Orbit Replaceable Unit, and the robot learns the operation skill by selecting the sequence which represents the most likely performance of the operator. The skill is learned in Cartesian space, joint space, and velocity domain. The experimental results demonstrate the feasibility of the proposed method in learning human skill and teleoperation control. The learning is significant in eliminating sluggish motion and correcting the motion command which the operator mistakenly generates.

14. SUBJECT TERMS

15. NUMBER OF PAGES

28 pp

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
unlimited	unlimited	unlimited	unlimited

Contents

List of Figures

Abstract

In this paper, we discuss the problem of how human skill can be represented as a parametric model using a hidden Markov model (HMM), and how a HMM-based skill model can be used to learn human skill. HMM is feasible to characterize two stochastic processes – measurable action and immeasurable mental states – which are involved in the skill learning. We formulated the learning problem as a multi-dimensional HMM and developed a programming system which serve as a skill learning testbed for a variety of applications. Based on "the most likely performance" criterion, we can select the best action sequence from all previously measured action data by modeling the skill as HMM. This selection process can be updated in real-time by feeding new action data and modifying HMM parameters. We address the implementation of the proposed method in a teleoperation-controlled space robot. An operator specifies the control command by a hand controller for the task of exchanging Orbit Replaceable Unit, and the robot learns the operation skill by selecting the sequence which represents the most likely performance of the operator. The skill is learned in Cartesian space, joint space, and velocity domain. The experimental results demonstrate the feasibility of the proposed method in learning human skill and teleoperation control. The learning is significant in eliminating sluggish motion and correcting the motion command which the operator mistakenly generates.

1 Introduction

1.1 Skill and Skill Learning

Human skill is the ability to apply past knowledge and experience in performing various given tasks. Skill can be gained incrementally through learning and practicing. To acquire, represent, model, and transfer human skill or knowledge has been a core objective for more than two decades in the fields of artificial intelligence, robotics, and intelligent control. The problem is not only important to the theory of machine intelligence, but also essential in practice for developing an intelligent robotic system.

The problem of skill learning is challenging because of the lack of a suitable mathematical model to describe human skill. Consider the skill as a mapping: mapping stimuli onto responses. A human associates responses with stimuli, associates actions with scenarios, labels with patterns, effects with causes. Once a human finds a mapping, intuitively he gains a skill. Therefore, if we consider the " stimuli" as input and " responses" as output, the skill can be viewed as a control system. This "control system" has the following characteristics:

- It is nonlinear, that is, there is no linear relationship between the *stimuli* and *responses*.

- It is time-variant, that is, the skill depends upon the environmental conditions from time to time.

- It is non-deterministic, that is, the skill is of inherently stochastic property, and thus it can only be measured in the statistical sense. For example, even the most skillful artist can not draw identical lines without the aid of a ruler.

- It is generalizable, that is, it can be generalized through a learning process.

- It is decomposable, that is, it can be decomposed into a number of low-level subsystems.

The challenge of skill learning depends not only upon the above mentioned inherent nature of the skill, but also upon the difficulty of understanding the learning process and transferring human skill to robots. Consider the following:

- A human learns his skill through an incrementally improving process. It is difficult to exactly and quantitatively describe how the information is processed and the control action is selected during such a process.

- A human possesses a variety of sensory organs such as eyes and ears, but a robot has limited sensors. This implies that not all human skills can be transferred to robots.

- The environment and sensing are subject to noises and uncertainty for a robot.

These characteristics make it difficult to describe human skill by general mathematical models or traditional AI methods.

1.2 The Related Work

Skill learning has been studied from different disciplines in science and engineering with different emphasis and names. The idea of *learning control* presented in [1, 2] is based on the observation that robot manipulators, being subject to "playback control mode", repeat their motions over and over in cycles. The research on learning control have been reviewed in [3, 4]. For a repeatable task operated over a fixed duration, each time the system input and response are stored, the learning controller computes a new input in a way that guarantees that the performance error will be reduced on the next trial. Under some assumptions, the P-, PI- and PD-type learning laws have been implemented. This approach is based on control theory, but the problem is certainly beyond the domain. According to the characteristics we discussed previously, it is obviously insufficient to approach such a comprehensive problem from only a control theory point of view.

The concept of *task-level learning* can be found in [5]; related studies are reported in [6, 7, 8, 9]. The basic idea is that a given task can be viewed as an input/output system driven by an input vector responding with an output vector. There is a mapping which maps task commands onto task performance. In order to select the appropriate commands to achieve a desired task performance, an inverse task mapping is needed. *Task-level learning* has been studied in great deal for "trajectory learning" to provide an optimum trajectory through learning and has been successful for some simple cases. For a more complicated case which is realistic in practice, the inverse task mapping is too difficult to obtain.

Both *learning control* and *task-level learning* emphasize achieving a certain goal by practice, and pay no attention to modeling and learning the skill. From a different angle, a research group at MIT has been working on representing human skill [10, 11, 12, 13, 14]. The pattern recognition method [10, 14] and process dynamics model method [12, 13] were used to represent the control behavior of human experts for a deburring process. In the pattern recognition approach, the form of IF-THEN relationship: $IF(signal\ pattern), THEN(control\ action)$ was used to represent human skill. Human skill pattern model is a non-parametric model and a large database is needed to characterize the task features. The idea of the process dynamics model method is to correlate the human motion to the task process state to find out how humans change their movements and tool holding compliance in relation to the task process characteristics. The problem with this approach is that human skill can not always be represented by the explicit process dynamics model and if there is no such model, or if the model is incorrect, this method will not be feasible.

Considerable research efforts have been directed toward learning control architectures using connectionist or Neural Networks [15, 16, 17]. Neural Network (NN) approaches are interesting because of the learning capacity. Most of the learning methods studied by connectionists are parameter estimation methods [16]. In order to describe the input/output behavior of a dynamic system, NN is trained using input/output data, based on the assumption that the nonlinear static map generated by NN can adequately represent the system behavior for certain applications. Although NNs have been successfully applied to various tasks, their behaviors are difficult to analyze and interpret mathematically. Usually, the performance of the NN approach is highly dependent on the architectures; however, it is hard to modify the

2

architecture to improve the performance.

The limitation of the previous work can be summarized as follows: (1) the lack of a feasible method to represent human skill, (2) the deterministic way to model a stochastic process, (3) the separation of modeling and transferring of human skill.

1.3   Why Hidden Markov Model

Before we address our approach to the problem, the following three issues must be noted. First, since a human performance is inherently stochastic, for a repeatable task, if an operator does it many times or many operators do the same task, and each of actions which represent the skill can be measured, then these measurements are definitely different. However, these measurements represent the same skill at the same task. One might think that there must be something to represent the characteristics of the skill for this particular task. The identical case can be found if we consider speech recordings of the same word recorded at different times or from different speakers. The recordings are definitely different, but there is obviously an inherent property hidden in these recordings that represents the same word. Otherwise, how can a human recognize the same word? Therefore, "skill learning" is, to a certain extent, the problem of uncovering the characteristics from recording data which represents the nature of the skill.

Second, to model human skill we need a definition of how good the model is. Currently there is not a systematic definition of the criterion to measure the skill model. We propose the *most likely* criterion as a measure of the performance. That is, if the performance is the one that an operator most likely performs for the given task, we claim that the performance is a good representation of the skill. The most likely criterion is appropriate in the sense of maximizing the expected average performance and rejecting the action noise. The concept of the most likely performance makes it possible to use stochastic methods to cope with uncertainties in both human performance and environment.

Third, modeling human skill and transferring the skill to robots are two separate issues, and have been treated differently. However, there is certainly a relationship between these two issues. Thus, if one has a clear mind to represent the skill for specific tasks, it would be easy for him to learn the skill. Therefore, it is desirable to consider these two problems as a whole and approach it with the same framework.

Fully understanding the above three issues makes it easy to find out why HMM is appropriate in representing the skill. HMM is a doubly stochastic process with an underlying stochastic process which is not observable, but which can be observed through another set of stochastic processes which produce the sequence of observations. The rationale to employ HMM to represent human skill is as follows:

- HMM is a doubly stochastic model which is appropriate for the two stochastic processes that skill learning must deal with, i.e., the mental state (or intention) which is hidden, and the resultant action which can be measured.

- HMM is a parametric model and its parameters can be optimized by efficient algorithms for the accurate estimation. Thus, it can be updated incrementally, which is desirable for "learning".

- HMM treats its observation on a symbolic level. This makes the fusion of different sensory signals possible regardless of their physical meaning. This fusion is appropriate considering the fact that a human has different sensing perceptions, such as vision, contact feeling, motion feeling, etc.

- The combination of HMMs is still a HMM. This allows a robot to learn a hierarchical skill as a human does.

- HMM can represent all the training data in a statistic sense by its parameters. This allows us to obtain the skill model that characterizes the most likely human performance from measurable human actions.

HMM has been successfully applied to speech recognition [18, 19, 20, 21]. Recently it has been studied in force analysis and mobile robot path planning simulations [22, 23]. In general, the concept of HMM can be used in solving three basic problems : the evaluation problem, the decoding problem, and the training problem. These three problems are all related to skill learning. In the training problem, we provide model parameters in such a way that the model possesses a high probability of generating the observation for a given model and a set of observations. Therefore, the training process is to establish a skill model according to the measured data, i.e., the learning process. In the evaluation problem we can score the measured data according to the trained model for selecting the measured data that represent the closest performance to the trained model. In the decoding problem we may find the best human mental state sequence for a given task.

2 Problem Formulation Using HMM

2.1 Mathmetical Background

A hidden Markov model is a collection of finite states connected by transitions. Each state is characterized by two sets of probabilities: a transition probability, and a discrete output probability distribution or continuous output probability density function which defines the condition probability of emitting each output symbol from a finite alphabet or continuous random vector given the state.

A HMM can be defined by:

- {S} – A set of states including an initial state S_I and a final state S_F

- A – The transition probability matrix, $A = \{a_{ij}\}$, where a_{ij} is the transition probability of taking the transition from state i to state j

- B – The output probability matrix, $B = \{b_j(O_k)\}$ for discrete HMM, $B = \{b_j(x)\}$ for a continuous HMM, where O_k stands for a discrete observation symbol, and x stands for continuous observations of k-dimensional random vectors

in this paper, we consider only a discrete HMM. For a discrete HMM, a_{ij} and $b_j(O_k)$ have the following properties:

$$a_{ij} \geq 0, \quad b_j(O_k) \geq 0, \quad \forall\, i,j,k \tag{1}$$

$$\sum_j a_{ij} = 1 \quad \forall\, i \tag{2}$$

$$\sum_k b_j(O_k) = 1, \quad \forall\, j \tag{3}$$

If the initial state has distribution $\pi = \{\pi_i\}$, a HMM can be written in a compact notation

$$\lambda = (A, B, \pi) \tag{4}$$

to represent the complete parameter set of a hidden Markov model.

To obtain a HMM, we need to compute $P(O|\lambda)$. In the following, we introduce an efficient algorithm known as the forward-backward algorithm [20] for computing $P(O|\lambda)$.

Forward algorithm

Define the forward variable $\alpha_t(i)$ as

$$\alpha_t(i) = P(O_1 O_2 \cdots O_t,\ S_t = i|\lambda) \tag{5}$$

This is actually the probability of a partial observation sequence to time t and state S_i which is reached at time t, given the model λ. This probability can be inductively computed as follows:

5

1. Initialization:
$$\alpha_1(i) = \pi_i b_i(O_1) \quad 1 \le i \le N. \tag{6}$$

2. Induction:
$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(O_{t+1}),$$
$$1 \le t \le T - 1, \quad 1 \le j \le N. \tag{7}$$

3. Termination:
$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{8}$$

In this way, the computation for $\alpha_t(j)$ is only on the order of $O(N^2 T)$.

In a similar way, a backward variable $\beta_t(i)$ can be defined as:

$$\beta_t(i) = P(O_{t+1} O_{t+2} \cdots O_T | S_t = i, \lambda) \tag{9}$$

i.e., the probability of partial observation sequence from $t + 1$ to the final observation T, given state i at time t and the model λ. This backward variable can be computed as follows:

Backward algorithm

1. Initialization:
$$\beta_T(i) = 1, \quad 1 \le i \le N. \tag{10}$$

2. Induction:
$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j),$$
$$t = T - 1, T - 2, \cdots, 1, \quad 1 \le i \le N. \tag{11}$$

The computation complexity of $\beta_t(i)$ is approximately same as that of $\alpha_t(i)$. Either the Forward or Backward algorithm can be used to compute $P(O|\lambda)$.

2.2 Problem Formulation

Having defined the most likely performance, we are able to use HMM to model human skill. We consider human action (movement, force, torque, etc.) as the observable stochastic process and human knowledge or strategy as the underlying stochastic process. We first discuss the unit representation problem, one of the most important issues to obtain HMM. In order to best represent the skill, we would like to use a unit which is as detailed as possible. The detailed unit, however, needs more data and computation to estimate the parameters. To compromise these two concerns, we can choose the unit according to the

application requirement. Unlike speech recognition, we don't have natural units such as words and phonemes. Tasks, subtasks and other artificial units are candidates of the unit for the skill learning. For example, if we use task as a unit and corresponding subtasks as states, then the trained HMM is a mapping from commands to output control signals. In this way, we can treat human mental states and actions under the same framework. Human intention or strategy for a given task can be represented by transition possibilities and output possibilities, and using the same model we can "learn" human intention or strategy. Alternatively, we could use more detailed units such as subtasks, and combine the subtask models into a task model when needed. It is noted that, even if a poor unit is selected, the HMM has an ability to absorb the suboptimal characteristics within the model parameters.

Consider a system which can be described at any time as being in one of a set of N distinct states $S_1, S_2, \ldots, S_N$, and the states are unobservable. We consider these states corresponding to the human mental states and the output symbols corresponding to his actions. The actual state at time t measured from action is denoted by q_t. When the system is in state $q_t = S_i$, M distinct output symbols $O_1, O_2, \ldots, O_M$ can be observed.

A discrete output probability distribution, $B = \{b_i(k)\}$, is associated with each state, where

$$b_i(k) = P[O_k \ at \ t \ |q_t = S_i], \quad 1 \le i \le N, 1 \le k \le M \tag{12}$$

At next time $t+1$, the system goes to state $q_{t+1} = S_j$ with transition probability a_{ji}, where

$$a_{ji} = P[q_{t+1} = S_j | q_t = S_i], \quad 1 \le i, j \le N. \tag{13}$$

The state transition coefficients have the following properties:

$$a_{ji} \ge 0 \tag{14}$$

$$\sum_{i=1}^{N} a_{ji} = 1. \tag{15}$$

In order to simplify the learning process, we consider only the first order HMM, which is based on the following assumptions: (1) *Markov assumption*: A new state is entered based only on the current state. (2) *Output-independent assumption*: The output probability distribution function depends only on the state at the particular time regardless of when and how the state is entered.

3   Learning Skill through HMM

3.1   Multi-dimensional HMM

In order to learn the most likely human performance, we have modeled human behavior as a HMM for a given task. Generally, it is desirable to employ a multi-dimensional HMM, in which there are more than one observable symbols at each time t, for the skill learning. First, robot motion, or force, is generally multi-dimensional. Therefore, the skill learning using motion or force measurements should be multi-dimensional. Second, for the purpose of fusion, the skill learning must deal with different sensory signals such as position, force, and vision. A multi-dimensional HMM provides a feasible approach to model these signals with different physical meanings.

For the learning trajectory, the learning procedure of a multi-dimensional HMM can be done in either Cartesian space or in joint space. If it is done in joint space , i.e., recording joint data and training the model in joint space, the mapping from Cartesian space to joint space is automatically avoided. Therefore, learning in joint space is extremely desirable for a kinematically redundant robot, so as to avoid the requirement of a task model and the expensive computation through optimization procedures. To deal with multi-dimensional data, the original HMM algorithms must be modified. For an R dimensional HMM, in state $q_t = S_i$, $M \times R$ distinct output symbols $O_1, O_2, \ldots, O_M$ can be observed, where R is the number of the joints and $O_k = [O_k(1), O_k(2), \ldots, O_k(R)]$. Since the trajectories in each DOF are independent, the output probability can be computed as the product of the output probability of each dimension. That is, the α computation can be modified as

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij} \right] \prod_{l=1}^{R} b_j(O_{t+1}(l)) \qquad (16)$$

where R is the number of DOF. Similarly, β can be computed as

$$\beta_t(i) = \left[\sum_{j=1}^{N} a_{ij} \beta_{t+1}(j) \right] \prod_{l=1}^{R} b_j(O_{t+1}(l)) \qquad (17)$$

3.2   Learning

Using multi-dimensional HMM, the objective of learning is to adjust the model parameters (A, B, π) to maximize the probability of the observation sequence. Our purpose is to obtain the parameters of the model from observations. If the model parameters are known, we can compute the probabilities of an observation produced by given model parameters and then update model parameters based on the current probabilities. These two algorithms can be combined for solving the learning problem as discussed below.

An iterative algorithm is used to update the model parameters. Consider any model λ with non-zero parameters. We first define the posterior probability of transitions γ_{ij}, from state

8

i to state j, given the model and the observation sequence,

$$
\begin{aligned}
\gamma_t(i,j) &= P(S_t = i,\ S_{t+1} = j | O, \lambda) \\
&= \frac{\alpha_t(i) a_{ij} \prod_{l=1}^{R} b_j(O_{t+1}(l)) \beta_{t+1}(j)}{P(O|\lambda)}
\end{aligned}
\tag{18}
$$

Similarly, the posterior probability of being in state i at time t, $\gamma_t(i)$, given the observation sequence and model, is defined as

$$
\begin{aligned}
\gamma_t(i) &= P(S_t = i | O, \lambda) \\
&= \frac{\alpha_t(i) \beta_t(i)}{\sum_{k=1}^{N} \alpha_T(k)}
\end{aligned}
\tag{19}
$$

$\sum_{t=1}^{T-1} \gamma_t(i)$ can be interpreted as the expected (over time) number of times that state S_i is visited, or, the expected number of transitions made from state S_i if time slot $t = T$ is excluded from the summation. Similarly, the summation of $\gamma_t(i,j)$ from $t = 1$ to $t = T - 1$ can be interpreted as the expected number of transitions from state S_i to state S_j.

Using the above formulas and the concept to count event occurrences, a new model $\overline{\lambda} = (\overline{A}, \overline{B}, \overline{\pi})$ can then be created to iteratively improve the old model $\lambda = (A, B, \pi)$. A set of reasonable reestimation formulas for π, A, and B is listed below:

$$
\overline{\pi} = \gamma_1
\tag{20}
$$

$$
\overline{a}_{ij} = \frac{\sum_{t=1}^{T-1} \gamma_t(i,j)}{\sum_{t=1}^{T-1} \sum_j \gamma_t(i,j)},
\tag{21}
$$

$$
\overline{b}_j^{(i)}(k) = \frac{\sum_{t \in O_t(i) = v_k^{(i)}} \gamma_t(j)}{\sum_t^{T} \gamma_t(j)},
\tag{22}
$$

$$
i = 1, 2, \dots R,
$$
$$
j = 1, 2, \dots N,
$$
$$
k = 1, 2, \dots M.
$$

where $v_k^{(i)}$ is observation symbol.

Equations (20) to (23) are the extension of Baum-Welch reestimation algorithm [24]. It has been proven that either the initial model λ defines a critical point of the likelihood function, where new estimates equal old ones, or will more likely produce the given observation sequence, i.e., model $\overline{\lambda}$ is more likely than model λ in the sense that $P(O|\overline{\lambda}) \geq P(O|\lambda)$.

If we repeat the above reestimation and use $\overline{\lambda}$ to replace λ, it is ensured that $P(O|\lambda)$ can be improved until a limiting point is reached. The Baum-Welch algorithm gives the maximum likelihood estimate of HMM and can be used to obtain the model which describes the most likely human performance for a given task.

Having determined the model representing the most likely human performance, we now look for the time sequence which best matches the trained model, i.e., find the time sequence with the highest $P(O|\lambda)$. The Forward-Backward algorithm can be employed to obtain the probability $P(O|\lambda)$ as discussed in the previous section. By scoring all time sequences, we can find the time sequence which best matches the trained HMM.
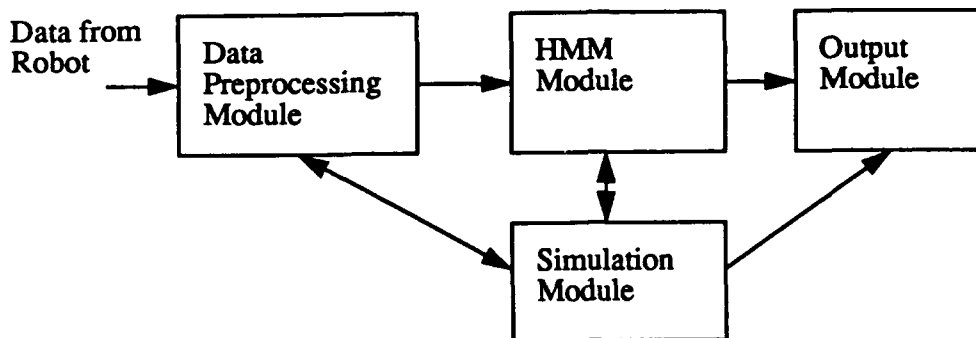
Figure 1: Block diagram of the programming system

4 Programming System

In order to demonstrate the concept of the most likely performance based on HMM, we have developed a programming system for which a block diagram is illustrated in Figure 1. The system can be used for skill learning in different tasks and in different domains, as long as the measurement can be described in HMM. The system can serve as a testbed to model and transfer human skill, or performance, and to verify the concept, computation, and efficiency of the learning methods. The system is composed of the following components. The Data Preprocessing Module converts the raw data from reading or Simulation Module into the symbols for training and recognition of HMMs. The main body of this module is the Short Time Fourier transformation (STFT) and the Vector Quantization (VQ). STFT is used to extract important features from the observable measurements with time localization and store those features in vectors. VQ technique is used to map a real vector onto a discrete symbol. The HMM Module integrates the software of HMM such as model construction, the algorithms for training and recognition of HMM, etc.The Simulation Module allows one to specify parameters and conditions for simulation study of HMM-based skill learning. The Output Module provides simulation and experiment results.

Implementation of the programming system provides a general tool for simulation and experimental studies using the HMM-based skill learning method. We address some implementation issues in the following subsections.

4.1 Feature Selection and Symbol Generation

In order to characterize human skill by HMM, we need to convert the measurements into finite symbols. A quantitative definition of symbols requires the definition of each symbol as a feature vector. For example, linear predictive coding (LPC) coefficient vectors [25] are usually used as the observed symbols in HMM-based speech recognition systems. For deciding the

extent of preprocessing a trajectory, the following factors are considered useful: existence of a preprocessing algorithm, its necessity, its complexity and its generality. The Fast Fourier Transformation (FFT) is a good tool to fulfill such a task. The Fourier transformation and its inverse establish a one-to-one mapping between the time domain and the frequency domain. FFT algorithms can be implemented efficiently. Furthermore, the Fourier transformation preserves information from the original signal, and ensures that important features are not lost as a result of FFT. Although the Fourier transform does not explicitly show the time localization of frequency components, the time localization can be presented by suitably pre-windowing the signal in time domain. Accordingly, STFT of a signal $x(t)$ is defined as [26]

$$STFT_x^\gamma(t, f) = \int_{t'} [x(t')\gamma^*(t' - t)]e^{-j2\pi f t'} dt' \qquad (23)$$

STFT at time t is the Fourier transform of the signal $x(t')$ multiplied by a shifted " analysis window" $\gamma^*(t' - t)$ centered around t. (All integrals are from $-\infty$ to ∞. The superscript * denotes complex conjugation.) Because multiplication by the relatively short window $\gamma^*(t' - t)$ effectively suppresses the signal outside a neighborhood around the analysis time point $t = t'$, the STFT is simply a "local spectrum" of the signal $x(t')$ around "analysis window" t. The windows can be overlapped to prevent loss of information.

VQ technique [27] can then be used to map a real vector onto a discrete symbol. A vector quantizer is completely decided by a codebook, which consists of a set of the fixed prototype vectors. A description of VQ process includes: (1) the distortion measure, and (2) the generation of certain number of prototype vectors. In the implementation, the squared error distortion measure is used, i.e.,

$$d(x, \hat{x}) = \|x, \hat{x}\| = \sum_{i=0}^{k-1} (x_i - \hat{x}_i)^2 \qquad (24)$$

The codebook is generated by VQ algorithm. One of the widely used algorithms is the LBG algorithm proposed by Linde, Buzo, and Gray [28]. The LBG algorithm iteratively splits the training data into 2, 4, ..., 2^m partitions with a centroid for each partition.

4.2   Scaling

Scaling is one of the most important issues in implementation. Recall that the forward and backward variables, $\alpha(i)$ and $\beta(i)$, consist of the sum of a large number of terms which are multiplications of elements of $\{a_{ij}\}$ and $\{b_j\}$. Since each a_{ij} and b_j is less than 1, $\alpha(i)$ and $\beta(i)$ will approach zero in an exponential fashion when the observation length T increases. For the multi-dimensional HMM, there are more multiplications of the output probabilities given the observation length. As a result, it is necessary to use re-scaling techniques. The most straightforward method is to multiply $\alpha(i)$ and $\beta(i)$ by a scaling factor so that they remain within the dynamic range of the computer for $1 \leq t \leq T$. Since the same scalings are done for $\alpha(i)$ and $\beta(i)$, the scaling factor will be canceled out at the end of the computation.

An alternative way to avoid underflow is to use a logarithmic representation for all the probabilities. This not only prevents underflow from happening, but also allows integers to

be used in the program representing the logarithmic values. If we represent probability P by $\log_b P$, more precision can be obtained by setting b closer to one. To multiply two numbers, we simply add their logarithms. To add two numbers, however, is more complicated. With a pre-computed table, we can implement the addition of two probabilities as one integer add, one subtract, two comparisons, and one table lookup [20].

4.3 Multiple Independent Sequences

The parameter estimation algorithm described in the previous section is only for one training datum., but it is easily generalized in our application where multiple training sequences are needed. Note that the Baum-Welch algorithm is nothing more than computing the frequencies of occurrence of various events. What we need to modify for multiple independent sequences is computing the frequencies of occurrence in each sequence separately and adding them together [20].

5 Skill Learning in Telerobotics

5.1 Self-Moble Space Manipulator (SM^2)

To evaluate the validity and effectiveness of the proposed scheme, we apply the previous learning scheme to the teleoperation control of the Self-Mobile Space Manipulator (SM^2), a space robot developed at the Robotics Institute of Carnegie Mellon University [29, 30]. The skill learning is one of the most crucial issues for space robots. As an add to understanding our experiments, we briefly introduce the system in this subsection.

SM^2 is a 7 DOF, 1/3-scale, laboratory version of a robot which was primarily designed to walk on the trusswork and other exterior surfaces of Space Station Freedom, and to perform manipulation tasks which are required for inspection, maintenance, and construction. Combining the mobility and manipulation functions in one body, SM^2, as a mobile manipulator, is capable of routine tasks such as inspection, parts transportation, object lighting, and simple assembly procedures. The system provides assistance to astronauts and greatly reduces the need for astronaut extra-vehicular activity (EVA).

The zero-gravity environment was established by the Gravity Compensation Systems. Two Gravity Compensation Systems, GCI and GCII, were developed for such a purpose. GCI allows walking experiments where both ends of the robot can be moved. GCII restricts one end of the robot to remain fixed under the boom-pivot point, but provides a second support point for payloads. GCII provides a high bandwidth response for the fast and precise motion needed for manipulation and fast-stepping experiments.

In order to perform a realistic experiment in the laboratory, a 1/3-size laboratory robot was designed and build based on a hypothetical, full-size, self-contained robot to be used on the space station. To achieve a mobility function on the Space Station trusswork, a five joint robot walker with minimum size and complexity was first developed. The robot includes five rotational joints and two slender links, and *node-grippers* at each end that enable it to attach itself to threaded holes in the truss nodes of space station freedom or other regular structure. Walking is accomplished by alternate grasping and releasing of the nodes by the grippers, and swinging of the feet from one node to the next. During each walking step, one end of the robot releases from a node, swings 90 or 180 degrees to a desired node location, and re-attaches to that node. Using such steps with alternate feet, SM^2 can move to any node on the exterior of the truss.

To provide the manipulation function, we developed a 2-DOF hand attached to the 5-DOF walker. The robot provides a 7-DOF serial configuration for manipulation (Figure 2), while the *node gripper* remains available for walking. For a manipulation task, the part gripper defines the tip of the robot, while one of the node grippers at the other end acts as the robot's base. The part gripper is electrically actuated, and has V-jaws (each jaw having two orthogonal Vs) suitable for grasping cylindrical or spherical objects from 25 mm (1 in) to 100 mm (4 in) in diameter.

While autonomous control is reasonable for locomotion, teleoperation is an efficient and
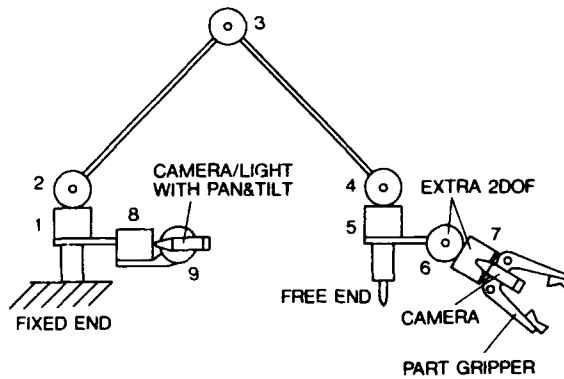
Figure 2: SM^2 configuration

feasible approach to the control of manipulation in space applications. We have developed a teleoperation interface with the Bird, a commercial, 6-DOF, free-flying hand controller which provides position and orientation of a radio transmitter (stationary) relative to a receiver (moving). The moving part of the Bird is mounted on a movable structure to improve the stability of the human arm during operation. In teleoperation, a human operator using experience or skill specifies the control input for robot motion in achieving certain goals associated to the given task by reviewing position and feeling force/torque. It would be desirable for a robot to learn human skill, or select the best control input according to certain criteria, for a repeatable task which can be recorded. This allows the robot to intelligently perform the task, to automatically reject noise through hand controllers or any other teleoperation tools, and to correct the undesirable control input that an operator may generate.

5.2 Task and Experiment Description

The task we investigated in this paper is exchanging an Orbit Replaceable Unit (ORU) teleoperation where an operator or astronaut gives a control command by a hand controller and the space robot receives the command to move around, find the destination, and replace ORU. The jaws of SM^2 are compatible with the handle of the ORU mockup that we built, and the gripper contains a motorized hex driver between its jaws to drive the hold-down screw of the ORU mockup (see Figure 3 and Figure 4). Gripper position (jaw opening) is measured with a potentiometer. Actuator current is measured to provide a rough indication of gripping force.

The exchange of an ORU requires the robot to have the capacity for gross motion, precise manipulation and load transporting. This is a typical example of teleoperated robot control. Since this is done by teleoperation, the robot performance is greatly dependent on the operator's skill and knowledge for a given task. Sometimes the motion is efficient and smooth, whereas at other times, the motion is slow and awkward (sluggish). This is why we
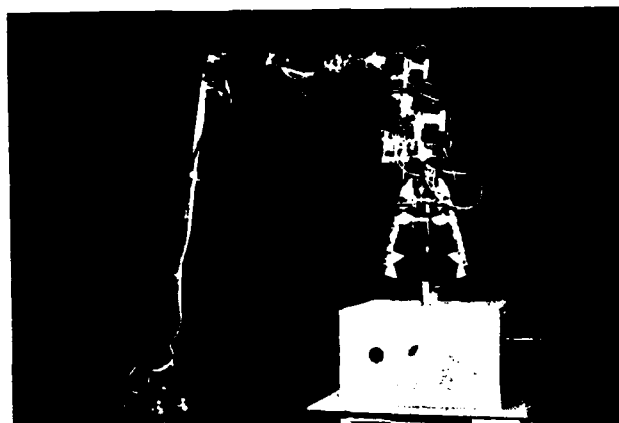
15

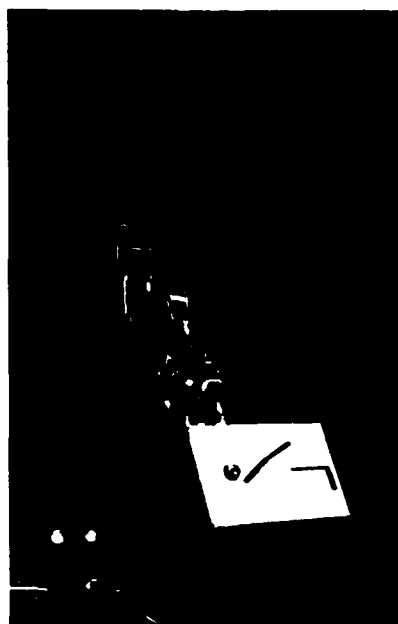Figure 3: Photograph of SM^2 approaching an ORU mockup

Figure 4: Photograph of SM^2 carrying an ORU mockup

want to model the operator's skill, or performance, so that the robot can learn the skill, and based on the skill model the robot is capable of providing smooth motion by correcting the action or the control command an operator mistakenly gives. The purpose of this research is to represent this type of knowledge by modeling the robot action measurement as a HMM, so as to allow the robot to learn human skill, to select the given commands from a human operator, and to provide an optimal control input for execution.

In the experiment, the movement of the robot is mainly in Z axis. The task can be viewed as following subtasks: moving down–grasping and unscrewing ORU–moving up. We taught the robot by giving the command for the part gripper motion, and recorded the data at 40 samples per second. A total of 100 operations were done by an operator and the corresponding trajectories of the position and velocity in both joint space and Cartesian space were collected. Four typical position trajectories of the task in 7 joints are shown in Figure 5 and the corresponding position trajectories in Cartesian space (Z axis) are shown in Figure 6. Since the mapping between Cartesian space and joint space is not unique, although the shapes of trajectories in Cartesian space are similar to one another, the shapes of trajectories in joint space are very different. Figure 7 shows a velocity trajectory which can be used to learn human skill in velocity domain.

We carried out three learning experiments in this study: (1) position trajectory learning in Cartesian space, (2) position trajectory learning in joint space, (3) velocity trajectory learning in Cartesian space. For the same experiments, the proposed approach is applied to learn operator skill from three different angles. Learning trajectory in Cartesian space is the most intuitive, considering the human control hand by viewing hand position with respect to destination in Cartesian space. Learning trajectory in joint space is also interesting. This study shows that the learning can be done in joint space. Moreover, it is more desirable to learn trajectory in joint space for a kinematically redundant robot to avoid the one-to-many mapping. Learning velocity trajectory demonstrates that, in some cases, it may be more convenient or more meaningful to model the skill in velocity domain. The velocity trajectory learning might be very useful for a variety of autonomous systems, such as learning driving skill for an vehicle with no driver.

Since we consider the trajectory as observable symbols, and subtasks as states in this approach, the underlying state sequence associated with the model has the property that, as time increases, the state index increases (or stays the same), i.e., the states proceed from left to right. We used a five state left-right HMM or Bakis model to model the task as shown in Figure 8.

The transition matrix in this case is

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 & 0 \\ 0 & a_{22} & a_{23} & a_{24} & 0 \\ 0 & 0 & a_{33} & a_{34} & a_{35} \\ 0 & 0 & 0 & a_{44} & a_{45} \\ 0 & 0 & 0 & 0 & a_{55} \end{bmatrix}$$

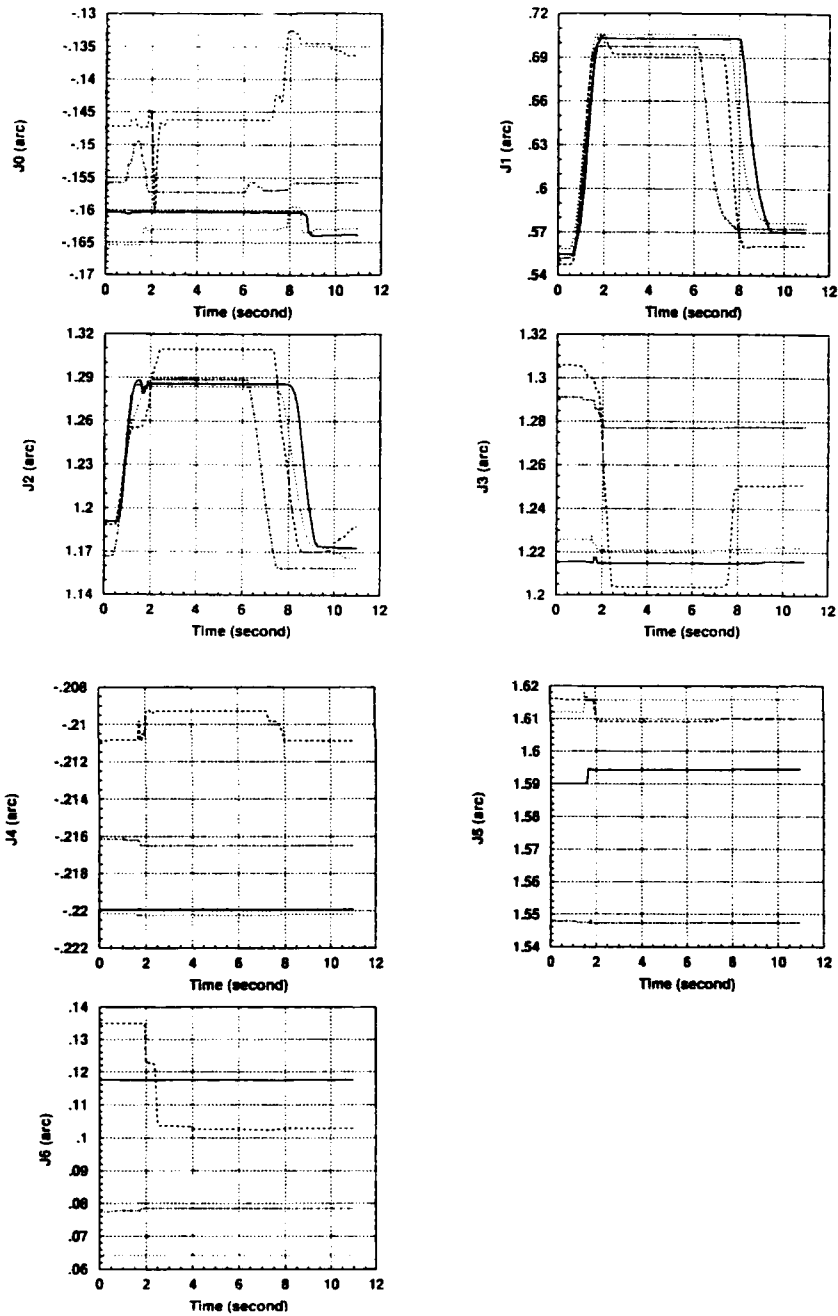Clearly this model has fewer parameters than that of ergodic or full connected HMMs.

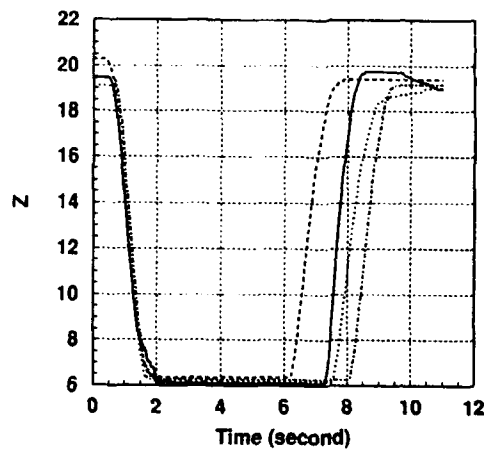Figure 5: Four typical position trajectories of ORU exchanging in seven joints

Figure 6: The corresponding position trajectories of ORU exchanging in the Cartesian space (Z axis)
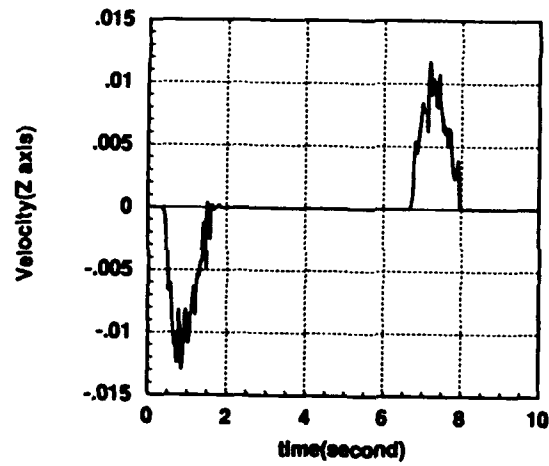

Figure 7: A velocity trajectory of ORU exchanging in the Cartesian space (Z axis)
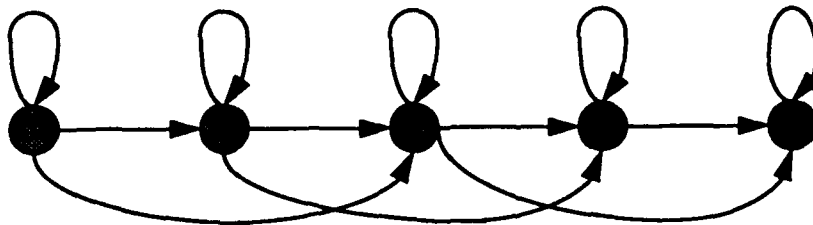

Figure 8: 5-state left-right HMM

Furthermore, the initial state probabilities have the property

$$\pi_i = \begin{cases} 0, & i \neq 1 \\ 1, & i = 1 \end{cases}$$

And t' e state transition coefficients of state 5 are specified as

$$a_{55} = 1,$$

$$a_{5i} = 0, \quad i < 5.$$

6 Trajectory Learning Experiments

6.1 Learning Position Trajectory in Cartesian Space

Because the motion of the part gripper for the action is mainly attributed by the motion in Z axis direction, HMM for the skill learning in Cartesian space was reduced to one-dimensional HMM, i.e., only one symbol is observable at each time t. This is a special case of multi-dimensional HMM. If we let $R = 1$ in equations (18)-(22), we can obtain the learning algorithms for one-dimensional HMM.

We employed FFT and VQ techniques for pre-processing the trajectories. The Hamming window was first used with a width of 400 ms (16 points) in every 200 ms. FFT analysis was then performed for every window. Finally, a set of 16-dimensional vectors was obtained from the amplitude of FFT coefficients. We trained the VQ codebook by those vectors and the VQ codebook was produced by LBG algorithm. The 256 vectors in the codebook were the symbols in the output probability distribution functions in our discrete HMM. An input vector corresponded to the set of 16 32-bit floating point FFT coefficient amplitudes, and was mapped onto an 8-bit index which represented one of 256 prototype vectors.

The observability matrix B is a 256×5 matrix with each column representing the observation probability distribution for one state. To initialize the model parameters, we first let output probabilities equal to $\frac{1}{256}$, where 256 is the VQ level. The transition probabilities were initialized by the uniformly distributed random number. With these initial parameters, the Forward-Backward algorithm was run recursively on the training data. The Baum-Welch algorithm was used iteratively to reestimate the parameters based on the forward and backward variables. After each iteration, the output probability distributions are smoothed using a floor between 0.0001 and 0.00001, and renormalized to meet stochastic constraints. Fifteen iterations were run for the training processes. The Forward algorithm was used to score each trajectory. Figure 9 shows the scores of the No. 60 and the No. 95 trajectories for each iteration. The score increase indicates that the model parameters are improved. The parameters converge after about 6 iterations. The final result of $P(O|\lambda)$ versus each trajectory is given in Figure 10 where we can find that the No. 95 trajectory is the best and the No. 60 is the worst.

6.2 Learning Position Trajectory in Joint Space

Trajectory learning also can be done in joint space by a multi-dimensional HMM. To model the skill in joint space for 7 DOF SM^2, a 7 dimensional HMM is employed to encode human skill. For a multi-dimensional HMM, the transition matrix A is the same as for a one-dimensional HMM. We pre-processed the trajectories in each joint in the same way that we did for the position trajectory learning in Cartesian space. We used a VQ codebook for each joint and the VQ codebooks were produced by LBG algorithm. Totally there were seven 256-vector codebooks generated by 7×5000 vectors. These sets of 256 vectors were the symbols in the output probability distribution functions in our discrete HMM. An input
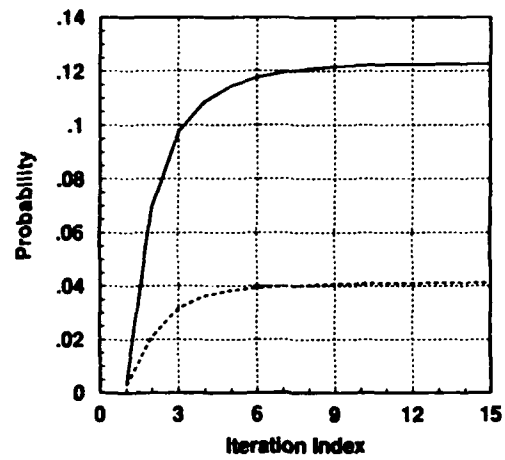
Figure 9: The forward scores for No. 60 and No. 95 trajectories ($\cdots$ score for No. 60, —— score for No. 95)
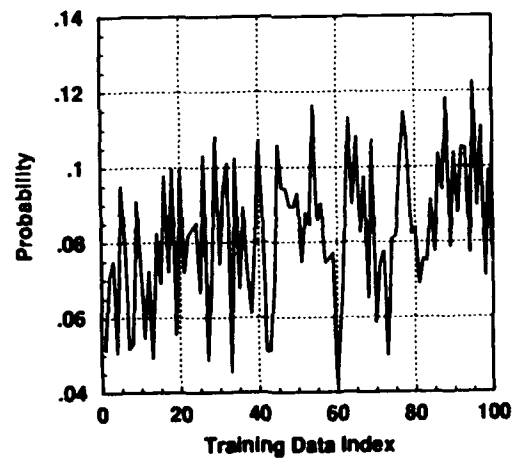

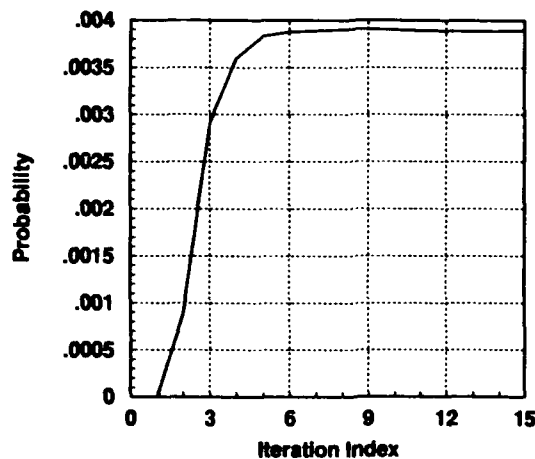Figure 10: The forward scores for all trajectories

Figure 11: The forward scores for No. 77 trajectory

vector corresponded to the set of 16 32-bit floating point FFT coefficient amplitudes, and was mapped onto an 8-bit index which represents one of 7 × 256 prototype vectors.

For output probabilities, we have seven 256 × 5 matrices with each column representing the observation probability distributions for one state. The output probabilities were initialized by $\frac{1}{256}$ and the transition probabilities were initialized by the uniform distributed random number. With these initial parameters, the extended Baum-Welch algorithm was used iteratively to reestimate the parameters according to the forward and backward variables. After each iteration, the output probability distributions are smoothed using a floor between 0.0001 and 0.00001, and renormalized to meet stochastic constraints. Fifteen iterations were run for the training processes. The Forward algorithm was used for scoring each trajectory. Figure 11 shows the scores of the No.77 trajectory for each iteration ($P(O|\lambda)$ versus iteration index). The score increase indicates that the improvement of the model parameters. The parameters converge after about 6 iterations. The final result of $P(O|\lambda)$ versus each trajectory is given in Figure 12 where we find that the No. 77 trajectory is the best and the No. 4 is the worst.

6.3   Learning Velocity Trajectory in Cartesian Space

The purpose of this experiment is to demonstrate that the HMM has the ability to learn the skills in different domains and the skills in different domains are not same for the same task. The velocity trajectory data in Cartesian space was used to train a 5 state one-dimensional HMM.

The basic training technique and procedures are same as for the previous experiments. A typical velocity trajectory is shown in Figure 7. After pre-processing the trajectories by the STFT and VQ techniques, we get a codebook and a series of symbols. The structure of A matrix is the same as the previous experiments. The B matrix is a 256 × 5 matrix with each column representing the observation probability distribution for one state. We initialized
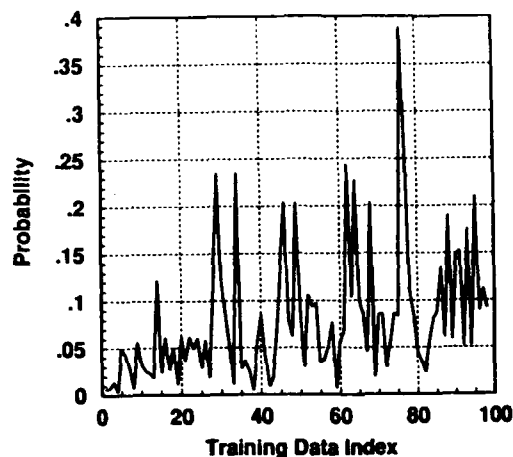
23

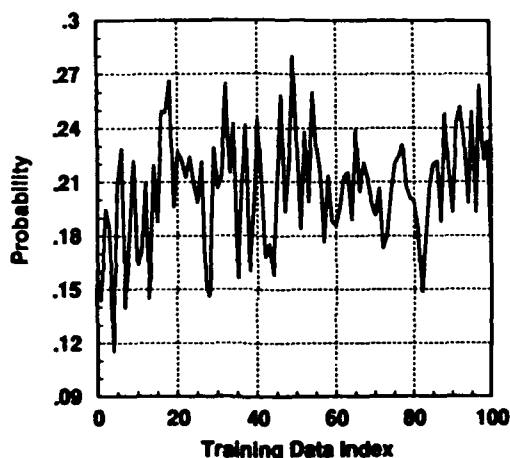Figure 12: The forward scores for all trajectories

Figure 13: The forward scores for all trajectories

output probabilities by $\frac{1}{256}$. The transition probabilities were initialized by the uniformly distributed random number. Twelve iterations were run for the training processes. The Forward algorithm was used to score each trajectory. The final result of $P(O|\lambda)$ versus each trajectory is given in Figure 13 where we find that the No. 49 trajectory is the best and the No. 4 is the worst.

6.4 Discussion

We have demonstrated that HMM is a feasible parametric model for skill learning in the different domains. Although we examined only the cases of the skill learning using position trajectory and velocity trajectory in this paper, the training data can be any sensory signal which reflects human skill. The experimental results showed that the skill model and the selection of the best performance in different domains could be different, even for the

24

same task and the same operator. This result implies that the skill learning depends upon sensation. Encoding human skill in joint space by the multi-dimensional HMM avoids the expensive computation of one-to many mapping for a kinematically redundant robot. Multi-dimensional HMM is also appropriate for fusion of different sensory signals with different physical meanings.

Theoretically, Baum-Welch algorithm gives only a local maximum of the likelihood function. If a poor initialization point is given, a poor local maximum may be reached. Particularly, if a probability is initialized to be zero, it will remain zero with every iteration. However, our experience showed that, for a discrete HMM, uniformly distributed initial points work well. The same observation was also given from other researches [20]. Another advantage of using discrete HMM is that the output distributions are automatically learned by the training process.

As mentioned in the previous section, the unit selection is an important issue in application of HMM. Furthermore, the structure and state selection are also important for effectiveness and efficiency of the learning procedure. In order to not lose information, we want to use a HMM which has complete structure and enough states. On the other hand, the complicated structure and extra states contribute extra parameters to be estimated. There is no standard for selecting structure and states. However, if the model is too complicated, we can eliminate the redundant states and paths. One way to get rid of extra parameters is decoding the trained model by the Viterbi algorithm [20] and deleting the states and paths unused or less used. Selecting a reasonable structure and number of states is the issue we are looking at currently.

Another issue is the real-time learning, i.e., dynamically updating the model to achieve the most likely performance. In real-time circumstance, we need to compute the frequencies of occurrence of the new data and add them to the model. The procedure is the same as that used to cope with multiple independent sequences. In this study, we have shown the fundamental theory and method that are needed and the preliminary experiments for real-time learning. However, various issues on real-time learning have not been discussed extensively. For example, what happens if the measured data fed in the learning process represents the poor skill, i.e., unskilled performance. Using the current method, the model will be updated to best match the performance of the operator, not to best represent the good skill. This is because we have a criterion to judge the model of the skill, but do not have a criterion to judge the skill itself. In other words, it is possible to become more unskilled in real-time learning. This is a common problem in other recognition fields such as speech recognition. One way to minimize the problem is to ensure the feeding data always represents the good performance. This again needs criterion to describe how good the skill is. We will look at this issue in the future.

7 Conclusion

In this paper we presented a novel method for human skill learning using HMM. HMM is a powerful parametric model and is feasible to characterize two stochastic processes – the measurable action process and immeasurable mental states – which are involved in the skill learning. We formulated the learning problem as a multi-dimensional HMM and developed a programming system which serve as a skill learning testbed for a variety of applications. Based on "the most likely performance" criterion, we can select the best action sequence out from all previously measured action data by modeling the skill as HMM. This selection process can be updated in real-time by feeding new action data and updating the HMM, and learning through this selection process.

As an example of the applications, we discussed the implementation of the proposed method in a teleoperation control space robot. An operator gave the control command by a hand controller for the Orbit Replaceable Unit exchanging task, and the robot learned the operation skill by selecting the sequence which represents the most likely performance of the operator. The skill was learned in Cartesian space, joint space, and velocity domain. The experimental results demonstrate the feasibility of the proposed method in learning human skill and teleoperation control. The learning is significant in eliminating sluggish motion and correcting the motion command that the operator mistakenly generates.

The method provides a feasible way to abstract human skill as a parametric model which is easily updated by new measurement. It will be found useful in various other applications, besides telerobotics, such as human action recognition in man-machine interface, coordination in anthropomorphic master robot control, feedback learning in the system with uncertainty and time-varying, and pilot skill learning for the unmanned helicopter. By selecting different units for the measured data in different problems, the basic idea is applicable for a variety of skill learning problems.

ACKNOWLEDGMENT

References

[1] S. Arimoto, S. Kawamura, and F. Miyazaki, "Bettering operation of robots by learning," *J. of Robot. Syst.*, vol. 1, No. 2, pp. 123-140, 1984.

[2] J.J. Craig, "Adaptive control of manipulators through repeated trials," Proc. American Control Conference, pp. 1566-1574, San Diego, June 6-8, 1984.

[3] S. Arimoto, "Learning control theory for robot motion," *Int. J. of Adaptive Control and Signal Processing*, vol. 4, pp. 543-564, 1990.

[4] K. Moore, M. Dahleh and S.P. Bhattacharyya, "Iterative learning control: a survey and new results," *J. of Robot. Syst.*, vol. 9, No. 5, pp. 563-594, 1992.

[5] E.W. Aboaf, "Task-level robot learning," *Master thesis*, MIT, August, 1988.

[6] E.W. Aboaf, C.G. Atkeson and D.J. Renkensmeyer, "Task-level robot learning," *Proc. 1988 IEEE Int. Conf. Robot. and Automat.*, pp. 1309-1310, 1988.

[7] E.W. Aboaf, S.M. Drucker and C.G. Atkeson, "Task-level robot learning: juggling a tennis ball more accurately," *Proc. 1989 IEEE Int. Conf. Robot. and Automat.*, pp. 1290-1295, 1989.

[8] C. Atkeson, "Using local weighted regression for robot learning", *Proc. 1991 IEEE Int. Conf. Robot. and Automat.*, pp. 958-963, 1991.

[9] M. Branicky, "Task-level learning: experiments and extensions," *Proc. 1991 IEEE Int. Conf. Robot. and Automat.*, pp. 266-271, 1991.

[10] H. Asada and B.-H. Yang, "Skill acquisition from human experts through pattern processing of teaching data," *Proc. 1989 IEEE Int.Conf. Robot. and Automat.*, pp. 1302-1307, 1989.

[11] H. Asada, "Teaching and learning of compliance using neural nets: representation and generation of nonlinear compliance," *Proc. 1990 IEEE Int.Conf. Robot. and Automat.*, pp. 1237-1244, 1990.

[12] H. Asada and S. Liu, "Transfer of human skill to neural net robot controller," *Proc. 1991 IEEE Int.Conf. Robot. and Automat.*, pp. 2442-2448, 1991.

[13] S. Liu and H. Asada, "Transferring manipulative skills to robots: representation and acquisition of tool manipulative skills using a process dynamics model," J. of Dynamic Syst., Measur., and Contr. vol. 114, pp. 220-228, June, 1992.

[14] B.-H. Yang and H. Asada, "Hybrid linguistic/numeric control of deburring robots," *Proc. 1992 IEEE Int.Conf. Robot. and Automat.*, pp. 1467-1474, 1992.

[15] Panos J. Antsaklis, Guest Editor, Special issue on neural networks in control systems, *IEEE Control System Magazine*, vol.10, No. 3, pp. 3-87, 1990.

[16] W.T. Miller, R. S. Sutton and P.I. Werbos, Editors, " Neural network for control," *MIT press*, 1990.

[17] Panos J. Antsaklis, Guest Editor, Special issue on neural networks in control systems, *IEEE Control System Magazine*, vol.12, No. 2, pp. 8-57, 1992.

[18] J.K. Baker, "The DRAGON system – an overview," *IEEE Trans. on ASSP*, vol. 23, No.1, pp.24-29, 1975.

[19] K.F. Lee, H.W. Hon and R. Reddy, "An overview of the SPHINX speech recognition system," *IEEE Trans. on ASSP*, vol. 38, No. 1, pp. 35-45, 1990.

[20] X.D. Huang, Ariki and M. A. Jack, "Hidden Markov models for speech recognition," *Edinburgh : Edinburgh University Press*, 1990.

[21] X.D. Huang, "Phoneme classification using semicontinuous hidden Markov models," *IEEE Trans. on ASSP*, vol. 40, No. 5, pp. 1062-1067, 1992.

[22] B. Hannaford, P. Lee, "Hidden Markov model analysis of force/ torque information in telemanipulation," *The International Journal of Robotics Research*, vol. 10, No. 5, pp.528-539, 1991.

[23] Q. Zhu, "Hidden Markov model for dynamic obstacle avoidance of mobile robot navigation," *IEEE Trans. on Robotics and Automation*, vol. 7, No. 3, pp.390-397, 1991.

[24] L.E. Baum, T. Petrie, G. Soules, and N.Weiss, " A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chins," *Ann. Math. Stat.*, vol. 41, No. 1, pp. 164-171, 1970.

[25] R.W. Schafer and L.R. Rabiner, "Digital representations of speech signals," *Proceedings of IEEE*, vol. 63, No. 4, pp. 662-677.

[26] F. Hlawstsch and G.F. Boudreaux-bartels, "Linear and quadratic time-frequency signal representations," *IEEE SP Magazine*, vol.9, No.2, 1992.

[27] R.M. Gray, "Vector quantization," *IEEE ASSP Magazine*, vol. 1, No.2, pp. 4-29, 1984.

[28] Y. Linde, A. Buzo, and R.M. Gray, " An algorithm for vector quantizer design," *IEEE Trans. on Commun.*, vol. COM-28 pp.84-95, 1980.

[29] Y. Xu, B. Brown, S. Aoki and T. Kanade, "Mobility and manipulation of a light-weight space robot", *Proceedings of IROS'92*, vol.1, pp. 11-19.

[30] Y. Xu, B. Brown, F. Friedman and T. Kanade, "Control system of self-mobile space manipulator, *Proceedings of IEEE Inter. Conf. on Robotics and Automation*, 1992.