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

Jie Yang *†, Yangsheng Xu *

C.S. Chen †

*The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

†Department of Electrical Engineering
The University of Akron
Akron, Ohio 44325

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 an 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. Based on "the most likely performance" criterion, the best action sequence can be selected from all previously measured action data by modeling the skill as an 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. 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

Skill learning is important both to the theory of machine intelligence and to developing an intelligent robotic system in practice. The problem 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, however, is nonlinear, time-variant, and non-deterministic. Moreover, 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. Furthermore, a human possesses a variety of sensory organs such as eyes and ears, but a robot has limited sensors. The environment and sensing are subject to noises and uncertainty for a robot. These facts cause an additional difficulty and make it impossible to deal with the problem by general mathematical models or traditional AI methods.

In this paper, we propose to use hidden Markov model (HMM) in skill learning. The rationale can be appreciated from the discussion of the following three issues. 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. Therefore, *skill learning* discussed in this paper refers to 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. 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 issues, and have been treated separately. However, there is certainly a relationship between these two issues. Therefore, it is desirable to consider these two problems as a whole and approach it with the same framework.

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

In this paper, we propose HMM approach to skill learning, and solve the modeling and transferring skill problem with the same framework. For a given task, human performance which represents human skill and intention is encoded by a multi-dimensional hidden Markov model. Given the model structure based on the knowledge of the task, the model parameters can be optimally estimated from training data. The trained model represents the most likely human performance and can be used for selecting robot control input. We discuss the implementation of the proposed method to skill learning in teleoperation of a redundant space robot, Self-Mobile Space Manipulator. The method is valuable for intelligent reliable teleoperation.

1050-4729/93 $3.00

2 Problem Formulation

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. HMM has been successfully applied to speech recognition [1, 2, 3]. Recently it has been studied in force analysis and mobile robot path planning simulations [4, 5]. We formulate skill learning problem using HMM in this section. For the more detailed reference on theory, computation, and application of HMM, the readers are recommended to refer to [2]. In this paper we consider only a discrete HMM.

The skill we consider here is human strategy which can be measured by his actions (e.g., movement, force) to achieve a given task. In teleoperation human operator gives commands by a hand controller, and the robot end-effector executes the task. The desired trajectory of the end-effector given by an operator through a hand controller reflects the operator's skill to perform the task. For a given task, if we consider operator commands as the input and end-effector trajectory as the output, this system is in open loop, i.e., the skill to be learned here is open loop trajectory skill. A trajectory is a finite sequence of actions. The goal of the open loop trajectory skill learning is to model the most likely performance from the all recorded data and select one trajectory closest to the most likely performance. Here, we consider the trajectory in teleoperation as the observable stochastic process and the knowledge or strategy behind it as the underlying stochastic process.

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. The actual state at time t measured from observation 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

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

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

$$a_{ji} = P[q_{t+1} = S_j | q_t = S_i],\ 1 \le i,\ j \le N. \tag{2}$$

The state transition coefficients have the following properties:

$$a_{ji} \ge 0, \tag{3}$$

$$\sum_{i=1}^{N} a_{ji} = 1. \tag{4}$$

We now discuss the unit representation problem. we would like to use a unit that 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. Task, subtask, and other artificial units are candidates of the unit for skill learning. If a poor unit is selected, the HMM has an ability to absorb the suboptimal characteristics within the model parameters. If we use task as a unit and corresponding subtasks as states. 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 unit such as subtask and combine the subtask models into task model when needed. 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. Suppose the task has m subtasks (or transitions), we can use a n ($n \ge m$) state left-right HMM or Bakis model to describe the task as shown in Figure 1.

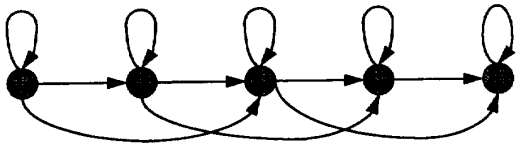

Figure 1: 5-state left-right HMM

The transition matrix in this case is

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 & \cdots & 0 \\ 0 & a_{22} & a_{23} & a_{24} & 0 & \vdots \\ \vdots & 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & a_{n-2,n-2} & a_{n-2,n-1} & a_{n-2,n} \\ \vdots & \ddots & \ddots & 0 & a_{n-1,n-1} & a_{n-1,n} \\ 0 & \cdots & \cdots & \cdots & 0 & a_{nn} \end{bmatrix} \tag{5}$$

Clearly this model has fewer parameters than that of ergodic or full connected HMMs. Furthermore, the initial state probabilities have the property

$$\pi_i = \begin{cases} 0, & i \ne 1 \\ 1, & i = 1 \end{cases} \tag{6}$$

And the state transition coefficients of state n are specified as

$$a_{nn} = 1,$$
$$a_{n,i} = 0, \quad i < n. \tag{7}$$

In order to train HMM, we need to record the all trajectories that we want the robot to learn and convert the trajectories into finite symbols. If we convert the continuous trajectory into p symbols by the certain signal processing techniques, B is a $n \times p$ matrix. HMM is trained by preprocessed data to find the optimal parameters in A and B to best present the all training data. The trained HMM represents the most most likely human performance. We then can evaluate the all trajectories and find one with the highest possibility to match the trained model.

3 Learning Skill through HMM

3.1 Multi-dimensional HMM

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, and thus 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. To obtain an HMM, we need to compute $P(O|\lambda)$ by an efficient algorithm known as the forward-backward algorithm [2]. For a multi-dimensional HMM, the forward-backward algorithm becomes:

Forward algorithm

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

$$\alpha_t(i) = P(O_1 O_2 \cdots O_t,\ S_t = i|\lambda) \tag{8}$$

This probability can be inductively computed as follows:

1. Initialization:

$$\alpha_1(i) = \pi_i b_i(O_1) \quad 1 \le i \le N. \tag{9}$$

2. Induction:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] \prod_{l=1}^{R} b_j(O_{t+1}(l)),$$
$$1 \le t \le T-1,\ 1 \le j \le N, \tag{10}$$

where R is the number of DOF.

3. Termination:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{11}$$

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

$$\beta_t(i) = P(O_{t+1} O_{t+2} \cdots O_T | S_t = i,\ \lambda) \tag{12}$$

This backward variable can be computed as follows:

Backward algorithm

1. Initialization:

$$\beta_T(i) = 1, \quad 1 \le i \le N. \tag{13}$$

2. Induction:

$$\beta_t(i) = \left[\sum_{j=1}^{N} a_{ij} \beta_{t+1}(j)\right] \prod_{l=1}^{R} b_j(O_{t+1}(l)),$$
$$t = T-1, T-2, \cdots, 1,\ 1 \le i \le N. \tag{14}$$

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 $P(O|\lambda)$.

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

$$\gamma_t(i,j) = P(S_t = i,\ S_{t+1} = j|O, \lambda)$$
$$= \frac{\alpha_t(i) a_{ij} \prod_{l=1}^{R} b_j(O_{t+1}(l)) \beta_{t+1}(j)}{P(O|\lambda)} \tag{15}$$

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

$$\gamma_t(i) = P(S_t = i|O, \lambda)$$
$$= \frac{\alpha_t(i)\beta_t(i)}{\sum_{k=1}^{N} \alpha_T(k)} \tag{16}$$

$\sum_{t=1}^{T-1} \gamma_t(i)$ 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:

$$\overline{\pi} = \gamma_1 \tag{17}$$

$$\overline{a}_{ij} = \frac{\sum_{t=1}^{T-1} \gamma_t(i,j)}{\sum_{t=1}^{T-1} \sum_{j} \gamma_t(i,j)}, \tag{18}$$

$$\overline{b}_j^{(i)}(k) = \frac{\sum_{t \in O_t(i) = v_k^{(i)}} \gamma_t(j)}{\sum_{t}^{T} \gamma_t(j)}, \tag{19}$$
$$i = 1, 2, \ldots R,$$
$$j = 1, 2, \ldots N,$$
$$k = 1, 2, \ldots M.$$

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

Equations (17) to (19) are the extension of Baum-Welch reestimation algorithm [6]. 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 is employed to obtain the probability $P(O|\lambda)$. By scoring all time sequences, we find the time sequence which best matches the trained HMM.

4 Skill Learning in Telerobotics

4.1 Programming System

In order to demonstrate the concept of HMM approach to skill learning based on the most likely criterion, we developed a programming system. 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, 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.

4.2 Teleoperation Task Description

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 [8, 9] (Figure 2). 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 tasks. 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

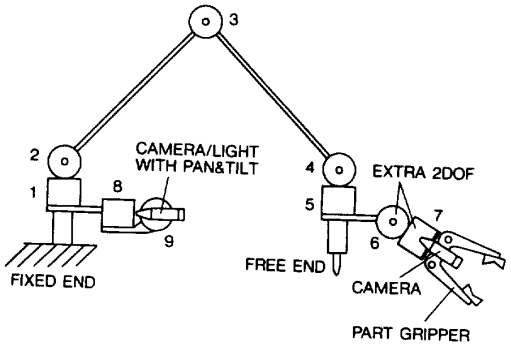

Figure 2: SM^2 configuration

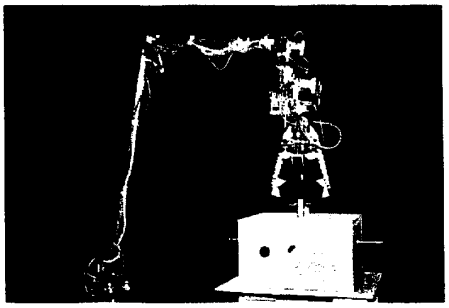

Figure 3: Photograph of SM^2 approaching an ORU mockup

(EVA). The skill learning is one of the most crucial issues for space robots.

The task we investigated in this paper is exchanging an Orbit Replaceable Unit (ORU) where an operator gives a control command by a hand controller and the space robot receives the command to move on the truss, 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). Gripper position (jaw opening) is measured with a potentiometer. Actuator current is measured to provide a rough indication of gripping force.

An ORU exchange task requires the robot to be capable of 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 want to model the operator's skill, or performance, so that the robot can learn the skill. Based on the skill model the robot

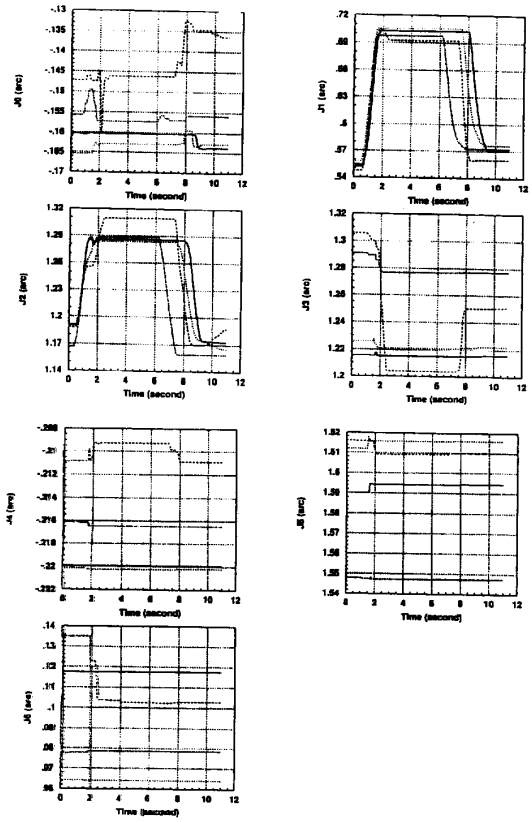

Figure 4: Four typical position trajectories of ORU exchanging in seven joints

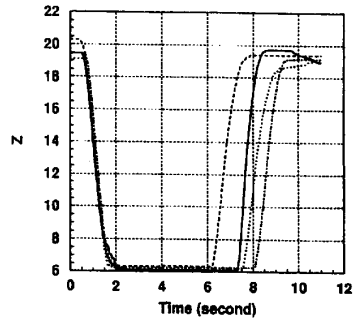

Figure 5: The corresponding four position trajectories of ORU exchanging in the Cartesian space (Z axis)

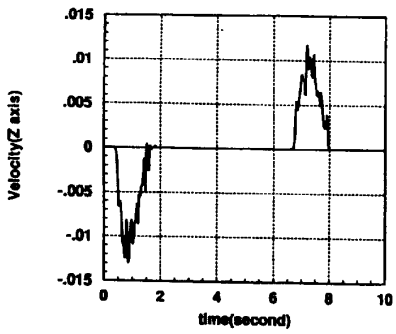

Figure 6: A velocity trajectory of ORU exchanging in the Cartesian space (Z axis)

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 an 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 robot end-effector movement is mainly in Z axis. The task can be viewed as a combination of the following subtasks: moving down; grasping and unscrewing ORU; moving up. We taught the robot by giving the command to the robot end-effector. We 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 4 and the corresponding position trajectories in Cartesian space (Z axis) are shown in Figure 5. 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 6 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.

We used a five state left-right HMM or Bakis model to model the task as shown in Figure 1. Let $n = 5$ we can obtain the form of transition matrix A, the initial state probabilities, and the state transition coefficients of state 5 from equations (5) to (7).

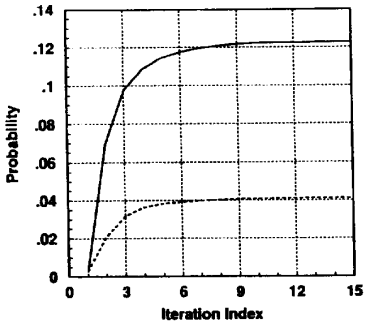

Figure 7: The forward scores for No. 60 and No. 95 trajectories (··· score for No. 60, —— score for No. 95)

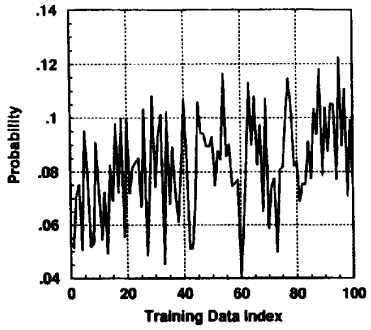

Figure 8: The forward scores for all position trajectories in Cartesian space

5 Experiments

5.1 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 (15)-(19), 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 [7] 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 7 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 8 where we can find that the No. 95 trajectory is the best and the No. 60 is the worst.

5.2 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 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. Fifteen iterations were run for the training processes. The Forward algorithm was used for scoring each trajectory. The final result of $P(O|\lambda)$ versus each trajectory is given in Figure 9 where we find that the No. 77 trajectory is the best and the No. 4 is the worst.

5.3 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.

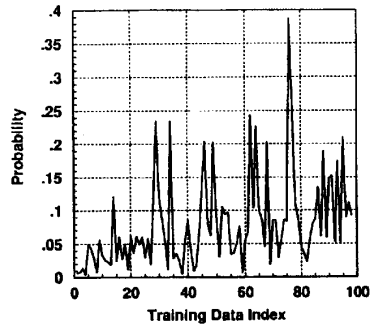

Figure 9: The forward scores for all position trajectories in joint space

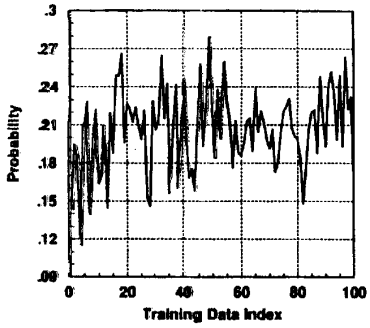

Figure 10: The forward scores for all velocity trajectories Cartesian space

The basic training technique and procedures are same as for the previous experiments. A typical velocity trajectory is shown in Figure 6. 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 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 10 where we find that the No. 49 trajectory is the best and the No. 4 is the worst.

6 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.

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 and uncovering the hidden process for different problems, the basic idea is applicable for feedback learning control which is one of our ongoing research topics in this direction.

References

[1] 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.

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

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

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.