Trajectory Planning of Robot Manipulators in Noisy Work Spaces Using Stochastic Automata

B. John Oommen, Nicte Andrade and S. Sitharam Iyengar

The International Journal of Robotics Research 1991 10: 135

DOI: 10.1177/027836499101000205

B. John Oommen
Nicte Andrade
School of Computer Science
Carleton University
Ottawa, Canada

S. Sitharam Iyengar
Department of Computer Science
Louisiana State University
Baton Rouge, Louisiana

Trajectory Planning of Robot Manipulators in Noisy Work Spaces Using Stochastic Automata

Abstract

We consider the problem of a robot manipulator operating in a noisy work space. The robot is assigned the task of moving from $\mathbf{P}_i$ to $\mathbf{P}_f$. Because $\mathbf{P}_i$ is its initial position, this position can be known fairly accurately. However, because $\mathbf{P}_f$ is usually obtained as a result of a sensing operation, possibly vision sensing, we assume that $\mathbf{P}_f$ is noisy. We propose a solution to achieve the motion that involves a new learning automaton, called the Discretized Linear Reward-Penalty (DL_{RP}) automaton. The strategy we propose does not involve the computation of any inverse kinematics. Alternatively, an automaton is positioned at each joint of the robot, and by processing repeated noisy observations of $\mathbf{P}_f$ the automata operate in parallel to control the motion of the manipulator.

1. Introduction

Robotics is probably one of the most fascinating and interesting areas of engineering and computer science. Not only is it an area of great importance economically, but as a research area, robotics encompasses such fields as kinematics, mechanics, computational geometry, controls, and language design.

One of the most interesting areas in robotics is the study of the problem of navigating a robot (or a manipulator) within a work space. When the robot has no obstacles to avoid and is operating in a noise-free work space, the problem is essentially a control problem. Solutions usually involve joint interpolated motions when the trajectory is not necessarily linear, and motions computed using recursive algorithms (such as Taylor's algorithm) if the path desired is linear. However, the problem is far more complex if the robot (or manipulator) has to plan its motion when there are obstacles in its work space or if the work space is noisy.

The initiator for the work in motion planning amidst obstacles was probably Udupa (1977). He introduced the concept of working in configuration space, and this was later studied extensively by many researchers (Lozano-Pérez and Wesley 1979; Brady et al. 1983; Lozano-Pérez 1983; Brooks and Lozano-Pérez 1985). Simultaneously, a variety of results concerning the theoretical issues of motion planning were presented by other researchers (Schwartz and Sharir 1983; Schwartz and Yap 1987) . The literature in this field is extensive, and we refer the reader to two comprehensive surveys of the results in the area: Whitesides (1985) and Schwartz and Yap (1987). We strongly recommend these surveys for a researcher who is just embarking on working in this field.

Whereas the problem of moving a single mobile robot in known terrains was almost completely investigated, the problem of navigating in unknown terrains was almost completely untouched until the past few years. The study of the problem of navigating mobile robots sparked a whole series of very interesting results. Learned paths were suggested by Iyengar and colleagues (Iyengar et al. 1985a), who also suggested using learned spatial graphs (Iyengar et al. 1985b) to compute the robot's path. Oommen et al. (1987) presented a formal approach to tackling the problem using learned visibility graphs. Although their solution was suitable only for a point robot, we believe that it is conceptually an ideal working model, inasmuch as the fact that the learned visibility graph actually converges to the actual visibility graph. Alternate solutions for practical robot navigation systems have been proposed that involve map making (Brady et al. 1983), environment learning (Chatilla 1982), terrain discretization

The International Journal of Robotics Research,
Vol. 10, No. 2, April 1991,

(Crowley 1985), and the use of multilevel planning (Giralt et al. 1979). Other papers involving mobile robots are those of Gouzenes (1984) and Chattergy (1985).

The subsequent question of much importance has been that of navigating (or moving) multiple objects. Schwartz and Sharir (1983) presented an analytic solution for the special case in which the objects to be moved were circular and the obstacles were polygonal. This problem is currently being studied by a number of researchers. Suffice it to say that the general problem of coordinating the motion of multiple independent objects was shown (Hopcroft et al. 1984) to be PSpace-Hard. From a practical viewpoint, Grossaman et al. (1985) of IBM considered the value of using multiple independent robot arms. Clearly this "value" depends on the criterion function used to evaluate the performance of the multiple robots. Based on the criteria investigated, they concluded that in both one and two dimensions there is "little merit" in having more than two arms.

Although the body of work done in the area of robotics and motion planning is extensive, the work concerning operating robots in real-life work spaces subject to noisy and inaccurate measurements is still in its infancy. Indeed, as Lozano-Pérez remarked in an opening address of the 1985 SIAM Conference on Robotics and Geometric Modeling (Kozlov 1985): "Nothing is ever where it is supposed to be—is the first law of Robotics." He went on to say that "one is lucky to find one paper" on the topics of planning error and planning sensing strategies. In a personal communication, Lozano-Pérez wrote, "Error is the central problem in robotics, but it often gets left behind in the problem formulation." This indeed is true.

In this article we will concentrate on the problem of controlling a robot manipulator in a noisy work space. If the robot is a *mobile* robot and the obstacles are fixed but unknown, a learning strategy can be used to learn the model of the world during (or alternatively prior to) the navigation process (Chatilla 1982; Iyengar et al. 1985a,b; Rao et al. 1985). However, even in this case, the problem has been only marginally studied when the observations obtained by the (vision or sonar) sensors are erroneous. However, when the robot is not mobile and the manipulator joints themselves have to be considered, the problem has a different flavor.

1.1. Problem Statement and Brief Survey of Solutions

In this article we consider the problem of a multilink robot manipulator operating in a noisy work space in which the joints of the robot can be prismatic and revolute. The robot is positioned at a configuration $\mathbf{P}_i$, which fully describes the position and orientation of its end effector. The robot is commanded to move to a configuration $\mathbf{P}_f$ inside the work space. $\mathbf{P}_f$ represents the desired ultimate position and orientation of the end effector. Because $\mathbf{P}_i$ is completely defined by the joint angles of the manipulator, it is not unrealistic to assume that it can be obtained to any desired degree of accuracy. However, because the accuracy of $\mathbf{P}_f$, the goal position, cannot be arbitrarily specified by the designer of the manipulator, it is conceivable that the robot may be asked to move to a noisy version of the configuration, $\mathbf{P}_f$. This is especially true if the latter configuration is obtained as a result of a sensing process —customarily a vision sensing process. The problem that we tackle in this article is indeed that of moving the robot from $\mathbf{P}_i$ to $\mathbf{P}_f$, where $\mathbf{P}_f$ is a fixed but unknown vector, which is unobservable. However, $\{\mathbf{Q}_f(n)\}$ is a sequence of observable points where

$$\mathbf{Q}_f(n) = \mathbf{P}_f + \eta$$

and η is an i.i.d. random vector. It is intended that the controller operates by processing $\mathbf{P}_i$ and $\{\mathbf{Q}_f(n)\}$.

Earlier, Azadivar (1987) studied the problem of incorporating the positional error associated with the individual joints of the robots in the motion planning. He did this by actually estimating success and failure parameters which fed into his optimization and feedback control loop. Arimoto and others (1985) studied mechanical and mechatronics systems with linear and nonlinear dynamics that may be operated repeatedly at low cost. Given a desired output y_d over a finite time duration [0,T] and an appropriate input u_0, these laws are formed by the following simple iterative processes:

1. $u_{k+1} = u_k + \Phi(y_d - y_k)$,
2. $u_{k+1} = u_k + \Gamma d/dt(y_d - y_k)$, and
3. $u_{k+1} = u_k + (\Phi + \Gamma d/dt)(y_d - y_k)$.

where u_k and u_{k+1} denote the kth and $(k+1)$st input values, Φ and Γ are positive-definite constant gain matrices, and y_k is the measured output at the kth time instant. Arimoto et al.'s work (1985) showed that law 1 with an appropriate gain matrix Φ is convergent in the sense that $y_k(t)$ approaches $y_{d(t)}$ as $k \to \infty$ in the meaning of the $L^2[0,T]$ norm if the objective system is linear and strictly positive. The same conclusion was proved when the system was subject to a linear time-invariant or time-varying mechanical system. In addition, a rough sketch of the convergence proof for the second and third learning control laws was presented for a class of linear and nonlinear dynamic systems.

Craig et al. (1986) presented an adaptive version of the computed-torque method for the control of manipulators with rigid links. The algorithm estimated

parameters online that appear in the nonlinear dynamic model of the manipulator (e.g., load and link mass parameters and friction parameters) and used the latest estimates in the computed torque servo. The authors proved the global convergence and stability of such a scheme in its nonlinear setting, as well as its asymptotic properties and derived the conditions for parameter convergence.

Koivo and Guo (1983) presented a new approach to the position and velocity control of a manipulator by using an adaptive controller of the self-tuning type for each joint. The complicated manipulator system was modeled by a set of time series difference equations. The parameters of the models were determined by online recursive algorithms that resulted from minimizing the sum of the squared errors. The adaptive controller of each joint was designed on the basis of the difference equation model and a chosen performance criterion function. The controller gains were then computed online using the model with the estimated values of system parameters. Later, Koivo (1986) studied the force-path control of a robotic manipulator in both the joint and the Cartesian coordinate systems. An autoregressive model with external excitation (ARX-model) was introduced for designing an adaptive controller with self-tuning. The controller minimized the conditional expectation of the sum of a quadratic position (velocity) error and a quadratic force error while satisfying the constraint of the ARX-model in which the estimated parameters were substituted for the unknown parameters. The basic approach was used to obtain an adaptive controller that operated on the variables that were expressed in terms of the joint coordinates. Another adaptive controller was similarly determined for the system variables expressed in terms of the Cartesian coordinates. The adaptive controller for force-path control in this work (Koivo 1986) has a form similar to that of a hybrid force/position controller, but the former has time-varying gains.

Going deeper into the traditional control systems concepts, Miller (1987) described a practical learning control system that is applicable to the control of complex robotic systems involving multiple feedback sensors and multiple command variables during both repetitive and nonrepetitive operations. In his controller, Miller used a general learning algorithm to learn to reproduce the relationship between the sensor outputs and the system command variables over particular regions of the state space of the system. The learned information was then utilized to predict the command signals required to produce desired changes in the sensor outputs. The learning controller required no a priori knowledge of the relationships between the sensor outputs and the command variables, thus facilitating the modification of the control system for specific applications.

The previous algorithms are just a few representative examples of solutions that are applicable to the problem we are studying. Indeed, any of the solutions found in traditional textbooks on optimal control and adaptive control can also be tailored to solve this particular problem.

In this article, we suggest a solution to the problem and consciously try to disengage ourselves from the traditional concepts of closed-loop feedback control theory. By this, we do not imply that the latter schemes are inferior. However, we aim to arrive at a solution that involves absolutely *no* estimation of parameters, absolutely *no* inverse kinematic computations, and *no* feedback control that is "hardware" oriented (i.e., that requires the tuning of servocontrollers, etc.). More importantly, apart from the scheme being computationally attractive, the solution is highly parallelizable.

The strategy that we propose to employ involves using learning automata. Learning automata have been extensively studied in the literature and have been used to model biologic mechanisms. They have also been used in pattern recognition, optimization, game playing, and more recently even in object partitioning. These automata interact with an environment, and based on the responses of a noisy environment they attempt to learn the optimal action offered by the environment.

We propose to use a learning automaton at every joint of the robot manipulator. This indeed involves merely maintaining a Finite State Machine at each joint, which dictates the motion that the particular joint has to make. Without using any other feedback arrangement except repeated noisy observations of $\mathbf{P}_f$, we propose to control the motion of the manipulator. Further, the control of the individual joints is achieved by having the automata operate *in parallel.* Finally, the feedback computations involved are of an elementary sort—they involve updating the states of the Finite State Machine.

In the next section we will discuss the elementary concepts of learning automata and discuss in detail the automaton that we use in this problem. We then discuss the use of the automata in the field of robotics.

2. Learning Automata

Learning automata have been extensively studied by researchers in the area of adaptive learning. The intention is to design a learning machine that interacts with an environment and dynamically learns the optimal action that the environment offers. The literature on

learning automata is extensive. We refer the reader to the work of Narendra and Thathachar (1974, 1989) and Lakshmivarahan (1981a). The last reference also discusses in fair detail some of the applications of learning automata, which include game playing, pattern recognition and hypothesis testing, priority assignment in a queueing system, and telephone routing. Other applications include the solution of stochastic geometric problems using learning automata (Oommen 1986a) and the partitioning of objects using various types of automata (Oommen and Ma 1987).

Broadly speaking, learning automata can be classified into two categories: Fixed Structure Stochastic Automata (FSSA), and Variable Structure Stochastic Automata (VSSA). An FSSA is one whose transition and output functions are time invariant. Examples of such automata are the Tsetlin, Krylov, and Krinsky automata (Tsetlin 1961, 1973). By far, most of the research in this area has involved the second category, namely VSSA. Automata in this category possess transition and output functions that evolve as the learning process proceeds. It can be shown that a VSSA is completely defined by a set of action probability updating functions (Varshavskii and Vorontsova 1963; Narendra and Thathachar 1974, 1989).

VSSA are implemented using a Random Number Generator (RNG). The automaton decides on the action to be chosen based on an action probability distribution. Nearly all the VSSA discussed in the literature permit probabilities that can take any value in the range [0, 1]. Hence the RNG must theroretically possess infinite accuracy. In practice, however, the probabilities are rounded off to a certain number of decimal places depending on the architecture of the machine that is used to implement the automaton.

To minimize the requirements on the RNG and to increase the speed of convergence of the VSSA the concept of discretizing the probability space was recently introduced in the literature (Oommen and Hansen 1984; Oommen and Christensen 1988). As in the continuous case, a discrete VSSA is defined using a probability updating function. However, as opposed to the functions used to define continuous VSSA, discrete VSSA utilize functions that can only assume a *finite* number of values. These values divide the interval [0, 1] into a finite number of subintervals. If the subintervals are all of equal length the VSSA is said to be linear. Using these functions discrete VSSA can be designed—the learning being performed by updating the action probabilities in discrete steps.

Learning automata can also be broadly classified in terms of their Markovian representations. Generally speaking, learning automata are either ergodic or possess absorbing barriers (Lakshmivarahan 1981; Narendra and Thathachar 1989). Automata in the former class converge with a distribution that is independent of the initial distribution of the action probabilities. This feature is desirable when interacting with a nonstationary environment, for the automaton does not "lock itself" into choosing any one action. However, if the environment is stationary an automaton with an absorbing barrier is preferred. Various absolutely expedient schemes that ideally interact with such environments have been proposed in the literature (Lakshmivarahan and Thathachar 1973; Lakshmivarahan 1981; Narendra and Thathachar 1989).

In this article we will present a discretized automaton, some of whose analytic properties have been introduced in the literature (Oommen and Christensen 1988). This machine is the Two-Action Discretized Linear Reward-Penalty (DL_{RP}) automaton. We shall state the principal property of the automaton, which is that the machine is ergodic and ϵ-optimal in certain restricted random environments. Indeed this is the only known symmetric linear reward-penalty automaton that is ϵ-optimal in *any* random environment. We will also consider the three-action Discretized Linear Reward-Penalty (DL_{RP}) automaton and show that it is expedient. We will then state the advantages of these automata over the traditional learning automata and proceed to propose their application to the particular robotics problem.

2.1. *Fundamentals and Learning Criteria*

The automaton considered in this article selects an action $a(n)$ at each instant 'n' from a finite action set $\{a_i | i = 1 \text{ to } R\}$. The selection is done on the basis of a probability distribution $\mathbf{p}(n)$, an $R \times 1$ vector where $\mathbf{p}(n) = [p_1(n), p_2(n), \ldots, p_R(n)]^T$ with

$$p_i(n) = Pr[a(n) = a_i],$$
$$\sum_{i=1}^{R} p_i(n) = 1 \qquad \text{for all } n. \tag{1}$$

The selected action serves as the input to the environment, which gives out a response $b(n)$ at time 'n'; $b(n)$ is an element of $B = \{0, 1\}$. The response '1' is said to be a 'penalty.' The environment penalizes the automaton with the penalty c_i, where

$$c_i = Pr[b(n) = 1 | a(n) = a_i] \qquad (i = 1 \text{ to } R). \tag{2}$$

Thus the environment characteristics are specified by the set of penalty probabilities $\{c_i\}$ ($i = 1$ to R). On the basis of the response $b(n)$ the action probability vector $\mathbf{p}(n)$ is updated and a new action chosen at $(n + 1)$. The reward probabilities are $1 - c_i$ for all i.

The $\{c_i\}$ are unknown initially, and it is desired that

as a result of interaction with the environment the automaton arrives at the action that evokes the minimum penalty response in an expected sense. It may be noted that if L is the action that obeys

$$c_L = \min_i (c_i), \tag{3}$$

then $p_L(n) = 1$, $p_i(n) = 0$ for $i \neq L$ achieves this result. Updating schemes for $\mathbf{p}(n)$ are to be chosen with this optimal solution in view.

With no a priori information, the automaton chooses the actions with equal probability. The expected penalty is thus initially M_0, the mean of the penalty probabilities. An automaton is said to learn *expediently* if, as time tends toward infinity, the expected penalty is less than M_0. We denote the expected penalty at time 'n' as $E[M(n)]$. The automaton is said to be *optimal* if $E[M(n)]$ equals the minimum penalty probability in the limit as time goes toward infinity. It is ϵ-optimal if in the limit $E[M(n)] < c_L + \epsilon$ for any arbitrary $\epsilon > 0$ by the suitable choice of some parameter of the automaton. Thus the limiting value of $E[M(n)]$ can be as close to c_L as desired.

3. The Discretized Linear Reward-Penalty (DL_{RP}) Automaton

3.1. The Two-Action DL_{RP} Automaton

The DL_{RP} automaton has $(N+1)$ states where N is an even integer. We refer to the set of states as $S = \{s_0, s_1, \ldots, s_N\}$. Associated with the state s_i is the probability i/N, and this represents the probability of the automaton choosing action a_1. Note that in this state the automaton chooses action a_2 with probability $(1 - i/N)$. Because any one of the action probabilities completely defines the vector of action probabilities, we will, with no loss of generality, consider $p_1(n)$.

The basic idea in the learning process is to make *discrete* changes in the action probabilities. By defining the transition map as a function from $S \times B$ to S, the changes in the action probabilities are indeed discrete. The transition map of the DL_{RP} automaton is specified by (4) below for $s(n) = s_k$, $1 \leq k \leq N-1$.

$$\begin{aligned} s(n+1) = s_{k+1} \quad & \text{if } a(n) = a_1 \text{ and } b(n) = 0, \\ & \text{or } a(n) = a_2 \text{ and } b(n) = 1 \\ = s_{k-1} \quad & \text{if } a(n) = a_1 \text{ and } b(n) = 1, \\ & \text{or } a(n) = a_2 \text{ and } b(n) = 0. \end{aligned} \tag{4}$$

Observe that (4) is valid only for the interior states. For the end states:

$$\begin{aligned} s(n+1) = s(n) \quad & \text{if } s(n) = s_0 \text{ or } s_N \text{ and } b(n) = 0 \\ = s_1 \quad & \text{if } s(n) = s_0 \text{ and } b(n) = 1 \\ = s_{N-1} \quad & \text{if } s(n) = s_N \text{ and } b(n) = 1. \end{aligned}$$

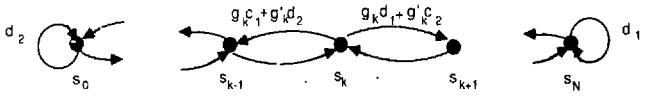

Fig. 1. Transition map of the DL_{RP} automaton.

Figure 1 shows the transition map of the automaton schematically.

Observe that if the machine is in state s_0, it has to choose a_2, and similarly if it is in s_N it has to choose a_1. Thus the change in action probabilities for $0 < p_1(n) < 1$ is:

$$\begin{aligned} p_1(n+1) = p_1(n) + 1/N \quad & \text{if } a_1 \text{ is chosen and } b(n) = 0 \\ & \text{or } a_2 \text{ is chosen and } b(n) = 1 \\ = p_1(n) - 1/N \quad & \text{if } a_1 \text{ is chosen and } b(n) = 1 \\ & \text{or } a_2 \text{ is chosen and } b(n) = 0. \end{aligned} \tag{5}$$

At the end states the following equality holds:

$$\begin{aligned} p_1(n+1) = p_1(n) \quad & \text{if } p_1(n) = 0 \text{ or } 1 \text{ and } b(n) = 0 \\ = 1/N \quad & \text{if } p_1(n) = 0 \text{ and } b(n) = 1 \\ = 1 - 1/N \quad & \text{if } p_1(n) = 1 \text{ and } b(n) = 1. \end{aligned}$$

The action probability updating rule warrants the name of the automaton. Note that if $c_1 < c_2$, the automaton has no absorbing barriers except in the degenerate cases when $c_1 = 0$ or $c_2 = 1$. The homogeneous Markov chain is defined by a stochastic matrix M whose arbitrary element $M_{i,j}$ is defined as:

$$\begin{aligned} M_{i,j} = Pr[s(n) = s_j | s(n-1) = s_i], \quad & \text{where} \\ M_{i,i-1} = g_i c_1 + g'_i (1 - c_2) \quad & \text{for } 1 \leq i \leq N, \\ M_{i,i+1} = g'_i c_2 + g_i (1 - c_1) \quad & \text{for } 0 \leq i \leq N-1, \\ M_{i,i} = 0 \quad & \text{for } 1 \leq i \leq N-1. \end{aligned} \tag{6}$$

In the above, $g_i = i/N$ and $g'_i = 1 - i/N$. All the other elements of M are zero. Furthermore, the boundary conditions for the Markov chain are specified by

$$M_{0,0} = (1 - c_2) \quad \text{and} \quad M_{N,N} = (1 - c_1). \tag{7}$$

The chain consists of exactly one closed communicating class. Further, because it is aperiodic, the chain is ergodic and the limiting distribution is independent of the initial distribution (Ross 1980). Let $\boldsymbol{\pi}(n)$ be the state probability vector, where, for all n,

$$\begin{aligned} \boldsymbol{\pi}(n) &= [\pi_0(n), \pi_1(n), \ldots, \pi_N(n)]^T, \\ \pi_i(n) &= Pr[s(n) = s_i], \quad \text{and,} \\ & \sum_{i=0}^{N} \pi_i(n) = 1. \end{aligned} \tag{8}$$

Then the limiting value of $\boldsymbol{\pi}$ is given by the vector that satisfies

$$\mathbf{M}^T \boldsymbol{\pi} = \boldsymbol{\pi}. \tag{9}$$

Using (9) we now state the asymptotic properties of the DL_{RP} automaton. The proofs of the theorems are

quite involved, but because they do not contribute to the fundamental application of the automata in robot control, they are omitted for the sake of brevity and continuity. The proofs are found elsewhere (Oommen and Christensen 1988).

THEOREM 1: Let $\Delta = (c_1 + c_2 - 1)$. Then π_i, the ith component of the asymptotic probability vector, obeys the following difference equation for $1 \leq i \leq N$.

$$\pi_i = \frac{c_2 - \Delta\left(\frac{i-1}{N}\right)}{(1-c_2) + \Delta\frac{i}{N}}\pi_{i-1} \qquad i = 1, 2, \ldots, N.$$

Proof: The theorem is proved elsewhere (Oommen and Christensen 1988). □

THEOREM 2: The DL_{RP} automaton is ϵ-optimal whenever the minimum penalty probability is less than 0.5.

Proof: The involved proof of the theorem is found elsewhere (Oommen and Christensen 1988). □

Remarks. (i) The question of whether the DL_{RP} automaton was ϵ-optimal was left open (Oommen 1986b), but Oommen conjectured that the machine was ϵ-optimal in all environments. As is obvious from the above, the latter conjecture is false. The ramifications of the result to get automata that are ϵ-optimal in all random environments are also described by these authors (Oommen and Christensen 1988).

(ii) When Tsetlin first designed the Tsetlin automaton, $L_{2N,2}$ (or linear tactic), his automaton was the first (deterministic or stochastic) automaton that could be proven to possess learning properties. The automaton was shown to be ϵ-optimal in environments whenever the minimum penalty probability is less than 0.5. It is not inappropriate to mention that the DL_{RP} automaton is *not* a generalized version of the linear tactic but is distinct in both design and operation for the following reasons:

1. Whereas the $L_{2N,2}$ automaton is a FSSA, the DL_{RP} scheme is a VSSA. The analytic reasoning for this is also in the literature (Oommen 1986; Oommen and Christensen 1988b).
2. In the case of the $L_{2N,2}$ automaton, the action probability vector is a *deterministic* vector. In the case of the DL_{RP} scheme, $\mathbf{p}(n)$ is a random vector. Thus whenever $c_1 < 0.5$, whereas in the former case the *probability* $p_1(\infty) \to 1$ as $N \to \infty$, in the latter case the *expected* probability $E[p_1(\infty)] \to 1$ as $N \to \infty$. Thus all the advantages of VSSA over FSSA (such as that of possessing the capability of choosing different actions at almost all consecutive time instances) are found in the DL_{RP} scheme. Additionally, the expected penalty tends to the value of the minimum penalty probability whenever the latter quantity is less than 0.5.

Modified absorbing and ergodic versions of the DL_{RP} automaton that are ϵ-optimal in *all* random environments have been presented elsewhere (Oommen and Christensen 1988). They are not of direct relevance to our particular problem and are not described here.

The use of the two-action DL_{RP} automaton to achieve the manipulator control will be discussed later. Indeed, this automaton commands the joint that it controls to either go forward or go backward in a discretized joint space. A generalization of this motion requires the joint controller to go forward, go backward, or stay at its current location. In order to understand the last motion, we need to study the design and the properties of the Multi-Action DL_{RP} automaton. Subsequently, the use of these machines in motion planning will be presented.

3.2. *The Multi-Action DL_{RP} Automaton*

The R-Action DL_{RP} automaton operates in a discretized probability space that divides the probability space [0,1] into NR intervals when $N \geq 1$ and does not divide the probability space at all if $N = 0$. The action-probability vector $\mathbf{p}(n)$ is defined by a set of probabilities, $[p_1(n), p_2(n), \ldots, p_R(n)]^T$, where $p_i(n)$ is the probability with which the automaton chooses action a_i, and the sum of these probabilities is unity. For the ease of notation, as before, we omit the reference to the time instant n.

Apart from (15), we constrain each p_i such that it has to be a value on the discretized space. Let $\delta = 1/NR$. Then,

$$p_i \in \{0, \delta, 2\delta, \ldots, (NR-1)\delta, 1\}.$$

The probability updating rule specified by the DL_{RP} automaton in the case of a reward ($b(n) = 0$) is as follows for $j \neq i$:

$$\begin{aligned} p_i(n+1) &= \max(p_i(n) - \delta, 0) && \text{if } a(n) = a_j,\ b(n) = 0 \\ &= 1 - \sum_{j \neq 1} \max(p_j(n) - \delta, 0) && \text{if } a(n) = a_i,\ b(n) = 0 \end{aligned} \tag{10}$$

The philosophy behind (10) is quite straightforward. If a_j is chosen and the automaton is rewarded, then each of the other p_js is decremented by δ if it is positive. These decrements are then added to $p_i(n)$.

To describe the case of a penalty response, we define a function RandVect, whose input is an integer $k <$

$R-1$, and j the index of an action. The output is a random subset $H_k(j)$ of k indices from the set $\{1,2, \ldots, R\} - \{j\}$. Using this notation, we define the updating rule as below:

$$\begin{aligned} p_i(n+1) &= \max(p_i(n) - (R-1)\delta, 0) && \text{if } a(n) = a_i,\ b(n) = 1 \\ &= p_i(n) + \delta && \text{if } a(n) = a_j,\ b(n) = 1,\ p_j(n) = k\delta, \text{ where } k \geq (R-1) \\ &= p_i(n) + \delta && \text{if } a(n) = a_j,\ b(n) = 1,\ p_j(n) = k\delta, \text{ where } k < R-1 \text{ and } i \in H_k(j). \end{aligned} \tag{11}$$

The philosophy motivating (11) is also quite straightforward. If a_i is chosen and the automaton is penalized, $p_i(n)$ is decremented by $(R-1)\delta$ if $p_i(n) \geq (R-1)\delta$, and this decrement is added to other actions, distributing the probability mass of δ for each action probability. However, if the action probability of the action chosen has only a value of $k\delta$, where $k < R-1$, this probability is decremented to zero, and k possibilities out of the $(R-1)$ remaining action probabilities are randomly chosen and each incremented by δ.

We now prove the asymptotic properties of the R-state DL_{RP} automaton. Unlike the results of Theorems 1 and 2, because the automaton described above is completely new to the literature, we believe that the proof of its asymptotic properties is fundamental to the results of this paper. The proof is thus presented below.

THEOREM 3: The R-state DL_{RP} automaton is expedient.

Proof: Consider the R-state DL_{RP} automaton with $N = 0$. In this case, the vector $\mathbf{p}(n)$ is a unit vector $\mathbf{e}_i$, where $\mathbf{e}_i$ is the vector with zeros and a unity in the ith position. By definition, if $\mathbf{p}(n) = \mathbf{e}_i$, the action chosen must be a_i. Thus, using (10) and (11), the probability updating rule can be written in the vector form as (12) below for $j \neq i$.

$$\begin{aligned} \mathbf{p}(n+1) &= \mathbf{e}_i && \text{if } \mathbf{p}(n) = \mathbf{e}_i,\ b(n) = 0 \\ &= \mathbf{e}_j && \text{with prob. } 1/(R-1) \text{ if } \mathbf{p}(n) = \mathbf{e}_i,\ b(n) = 1. \end{aligned} \tag{12}$$

Consider the Markov chain $\aleph$ defined by the states $\{i | 1 \leq i \leq R\}$, where $\aleph = i$ if a_i is the *action* chosen by the automaton. If $\mathbf{A}$ is the stochastic matrix defining the chain it can be seen that by virtue of (12), that $\mathbf{A}$ matrix has the form:

$$\mathbf{A} = \begin{bmatrix} 1-c_1 & c_1/R-1 & c_1/R-1 & \ldots & c_1/R-1 \\ c_2/R-1 & 1-c_2 & c_2/R-1 & & c_2/R-1 \\ \cdot & & & & \cdot \\ \cdot & & & & \cdot \\ \cdot & & & & \cdot \\ c_R/R-1 & c_R/R-1 & c_R/R-1 & \ldots & 1-c_R \end{bmatrix} \tag{13}$$

Let $\boldsymbol{\pi} = [\pi_1, \pi_2, \ldots, \pi_R]^T$ be the limiting asymptotic probability vector of the ergodic Markov chain described by (13). Clearly, $\boldsymbol{\pi}$ is obtained by solving

$$\boldsymbol{\pi} = A^T \boldsymbol{\pi}.$$

$\boldsymbol{\pi}$ is thus the eigenvector of the eigenvalue of A, which is unity.

Solving $[I - A]^T \boldsymbol{\pi} = 0$ yields for the first row,

$$\{c_1\pi_1 - (1/(R-1)) \sum_{i=2}^{R} c_i\pi_i\} = 0,$$

whence $\pi_1 = J/Rc_1$, where J is a constant independent of 'i' and is equal to

$$J = \sum_{j=1}^{R} c_j\pi_i.$$

Similarly, $\pi_i = J/Rc_i$, where J is defined above. Because the vector $\boldsymbol{\pi}$ is a probability vector, J can be solved for to yield the value for π_i as

$$\pi_i = (1/c_i)/\sum_{j=1}^{R} (1/c_j).$$

Because π_i is the limiting probability of being in state i, which is indeed the limiting probability of the automaton choosing a_i, the limiting expected penalty is $M(\infty)$, where,

$$M(\infty) = \sum_{j=1}^{R} \pi_i c_i = R/\sum_{j=1}^{R} 1/c_j.$$

Indeed, $M(\infty)$ is the harmonic mean of $\{c_i\}$. The expedience is proved from the above by observing that the harmonic mean of a sequence of numbers is never greater than that arithmetic mean. □

Remarks. (i) It can be seen that the above special case of the R-Action DL_{RP} automaton obtained by setting $N = 0$, is itself a very general (stochastic) version of Tsetlin's $L_{R,R}$ automaton (Tsetlin 1961, 1973). Of course, when $N \geq 1$, the resemblence between the two families of automata disappears, because Tsetlin's automaton is an FSSA, and the DL_{RP} automaton is, by definition, a VSSA.

(ii) Throughout the rest of the article we will merely be considering the case when $R = 3$, primarily because the actions that we require a joint controller to take are those of going forward one step in the discretized joint space, going backward one step, and staying at the same joint angle. In the case when $R = 3$, the probability vector can be drawn graphically as a point in the plane inside of an equilateral triangle. The height of the triangle is unity, and the vertices represent the unit vectors (1,0,0), (0,1,0), and (0,0,1). A point interior to the triangle represents the vector

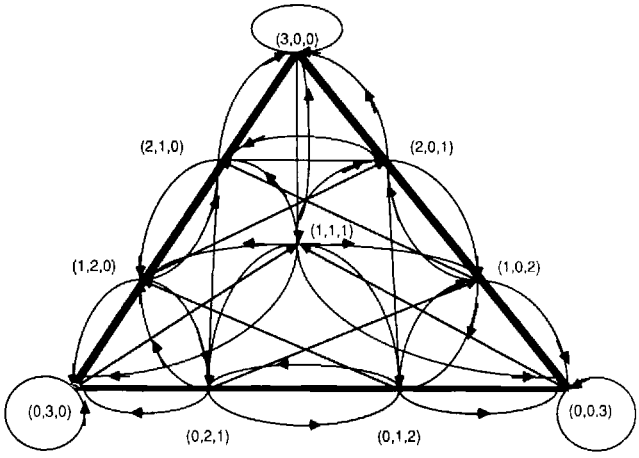

Fig. 2. The transition map of the DL_{RP} scheme for the case when R = 3 and N = 1.

$[p_1, p_2, p_3]^T$, where the quantity p_i is the length of the perpendicular from the point to the ith side of the triangle.

(iii) When $N = 1$, (i.e., when the probability space is divided into three equal intervals), there are 10 permissible probability vectors. These vectors are represented by the set of tuples (i, j, k) given below:

$$\{(i, j, k) | i + j + k = 3;\ i, j, k \geq 0\}.$$

Note that the action probability vector represented by the tuple (i, j, k) is the vector $[i/3, j/3, k/3]^T$. The physical representation of these tuples on the equilateral triangle described above and the transition matrix describing the Markov chain defined by the Three-Action DL_{RP} automaton are given in Figure 2 and Table 1, respectively.

The asympotic efficiency of the automaton can be studied by computing the stationary probabilities of **A** and computing the net average penalty. This has been done for various penalty probabilities $[c_1, c_2, c_3]$. In all cases the performance of the machine is superior to that of the corresponding machine obtained for $N = 0$.

We conjecture that the R-action DL_{RP} is ϵ-optimal whenever $c_{min} < 0.5$. The proof of the above conjecture will be quite involved. Indeed, it will require the solution of a stochastic tensor equation. However, simulation results seem to indicate that the conjecture is true.

4. Manipulator Control Using the DL_{RP} Automaton

The strategy by which we control the manipulator can now be proposed, because the theoretical framework has been laid.

Let us suppose we have a manipulator with K joints. These joints may be revolute or prismatic. The joint angles can be measured to yield the current position of the end effector $\mathbf{P}_i$, and this is assumed to be done quite accurately. The robot is continuously fed with a noisy version of the Cartesian coordinates of the desired goal position of the end effector $\mathbf{Q}_f(n)$. Unfortunately, $\mathbf{Q}_f(n)$ is noisy, and it represents the observable form of the *actual* goal position $\mathbf{P}_f$, the latter itself being unknown. Our intention is to control the manipulator so that it reaches arbitrarily close to $\mathbf{P}_f$. Furthermore, we intend to achieve the task adaptively.

The most straightforward method to achieve this "nonadaptively" is to take a large number of observations $\mathbf{Q}_f(n)$ of $\mathbf{P}_f$ and by computing the estimate of $\mathbf{P}_f$ from $\{\mathbf{Q}_f(n)\}$, any straightforward path planning strategy (for example, one using coordinated joint interpolated moves) can be utilized to move the end effector from $\mathbf{P}_i$ to this estimate of $\mathbf{P}_f$. We do not recommend this strategy for two reasons. First, this scheme is nonadaptive. Second, in a real environment, the process of obtaining a large number of observations of $\mathbf{P}_f$ can

Table 1. The Transition Matrix of the DL_{RP} Scheme for the Case When R = 3 and N = 1.*

	(300)	(030)	(003)	(210)	(120)	(201)	(102)	(021)	(012)	(111)
(300)	d1	**0**	0	0	0	0	0	0	0	c1
(030)	0	**d2**	0	0	0	0	0	0	0	c2
(003)	0	0	d3	0	0	0	0	0	0	c3
(210)	c2/6 + 2d1/3	0	0	0	d2/3	c2/6	0	2c1/3	0	0
(120)	0	c1/6 + 2d2/3	0	d1/3	0	2c2/3	0	c1/6	0	0
(201)	c3/6 + 2d1/3	0	0	c3/6	0	0	d3/3	0	2c1/3	0
(102)	0	0	c1/6 × 2d3/3	2c3/3	0	d1/3	0	0	c1/6	0
(021)	0	c3/6 + 2d2/3	0	0	c3/6	0	2c2/3	0	d3/3	0
(012)	0	0	c2/6 + 2d3/3	0	2c3/3	0	c2/6	d2/3	0	0
(111)	d1/3	d2/3	d3/3	c3/6	c3/6	c2/6	c2/6	c1/6	c1/6	0

* The transition map of the automaton is given in Figure 2.

be very time consuming, for it could involve processing as many digital images of the work space. Indeed, the robot will have to wait while all of these images are processed before it even starts its motion. Also, once it does start its physical motion, the time involved is again a large portion of the time that the user has to spend, because, as is well known, the mechanical motions are often the most time consuming. The ideal scenario is to plan the motion piecewise, and as the planned motion is executed mechanically, the computer plans the next segment of the motion. Indeed, this can be achieved (for example, in VAL II) by using motion commands suffixed by the symbol "!".

The strategy that we propose overcomes both of the above drawbacks. Let us assume that the Two-Action DL_{RP} automaton is the learning machine that is used. Every joint of the robot is equipped with such a machine, and each joint independently chooses to either go forward or go backward in the discretized joint space. The way by which this motion of going forward or backward one step is implemented is, of course, robot dependent.

The learning process of the manipulator is described as follows. Let $\psi(n)$ be a criterion function—for example, the Euclidian distance between $\mathbf{Q}_f(n)$ and $\mathbf{P}_i(n)$. Based on the actions stochastically chosen by the individual automata, the position of the end effector of the robot moves to $\mathbf{P}_i(n+1)$. The observation $\mathbf{Q}_f(n+1)$ is now obtained and the criterion function computed. If the latter function is less than it was in $\psi(n)$, each automaton is rewarded. Otherwise, the automata are penalized. Based on these responses the action probabilities are *locally* updated and the process repeated. When the criterion function is small enough, the process is terminated, and a fine motion planning strategy is invoked.

In the case when the Three-Action DL_{RP} automaton is the learning machine, each joint is equipped with such a machine, and each joint is commanded stochastically to go forward, stay where it is, or go backward at every time instant based on the action chosen by the automaton. Based on this decision, the position $\mathbf{P}_i(n+1)$ of the end effector is known, and using the updated observation $\mathbf{Q}_f(n+1)$ the criterion function is recomputed to analogously reward or penalize all the automata.

Our technique, apart from overcoming the drawbacks discussed above, has one major advantage. The strategy *does not* require the computation of any inverse kinematics and is thus extremely effective computationally. Indeed, inverse kinematic and inverse dynamic solutions can often have scores of parameters that are position, velocity, and acceleration dependent. By permitting the automata to perform stochastically and by repeatedly computing the straightforward criterion function, we have been able to avoid such tedious computations. Note also that the automata make their decisions and update their probability vectors independently, and hence they can be made to operate *in parallel,* thus reducing the actual physical time that elapses.

It must be noted that unlike most VSSA, maintaining the DL_{RP} automata involves incrementing (or decrementing) just one integer memory location per action, and furthermore the process of choosing an action involves invoking a random number generator exactly once per joint. This is quite appealing.

In both scenarios cited above, all the automata were either simultaneously penalized or simultaneously rewarded. If each automaton should get a distinct response, one has to consider how each automaton is performing, as opposed to seeing how the collective performance of the automata is. Thus in this case, a criterion function $\psi_i(n)$ can be computed for the ith joint, and the performance of this joint (in joint space) must be evaluated. However, this requires, at the very least, a linearized model of the inverse kinematics and can be more expensive than the scenarios discussed above. We will discuss the actual experimental significances of these issues in the next section.

5. Experimental Results

The technique that has been suggested in this article has been rigorously tested for a few simple two-dimensional robots. The first robot, R_1, is the familiar Horn's Robot (Brady et al. 1983), in which the robot operates in the plane and the two joints are revolute joints. The second robot, R_2, is the three-link generalization of Horn's Robot, in which the position and the orientation of the end effector can be controlled.

Various experiments were conducted in which $\mathbf{P}_i$ was specified and noisy versions of $\mathbf{P}_f$ were generated using noise that had a Gaussian (normal) distribution. In Figures 3 through 6, circles that represent perturbations of up to three standard deviations of the noise are presented to show the degree of noise introduced. Unfortunately, the circles appear elliptic because of the rectangular screens from which dumps of the snapshots were obtained for the actual final figures.

Automata with two actions and three actions were independently used to control the manipulator. In each case 100 experiments were performed, and the expected ensemble path of the robot end effector was traced. Also the expected decrease in the Euclidian distance from $\mathbf{P}_i(n)$ to final ideal goal position was also obtained. The graphs for a typical scenario are shown in Figure 3*A*,*B* for the case when $NR = 6$ and $R = 2$.

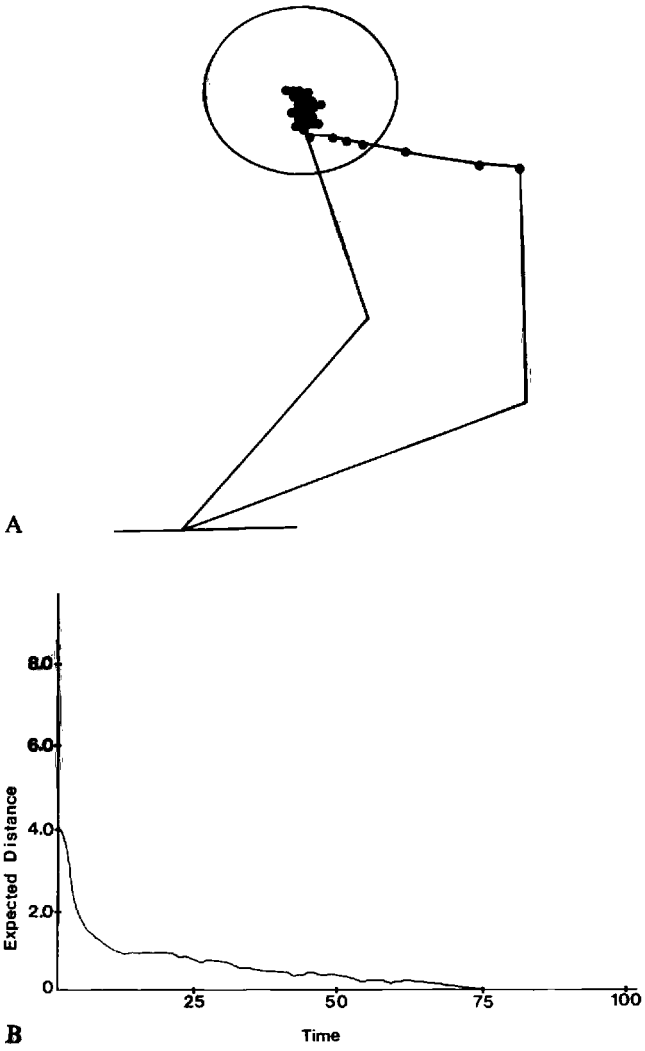

Fig. 3. (a) Expected path (given as a sequence of points) of the two-link Horn's Robot with links of length unity. The control is achieved using the DL_{RP} automaton with R = *2 and* NR = *6, and with a single response controlling both the automata. The standard deviation is 0.1 times the link length. The disk shows the three standard deviation range of noisy points. (b) Expected change in distance from the initial configuration to the final mean configuration for the set-up described in Fig. 3(a).*

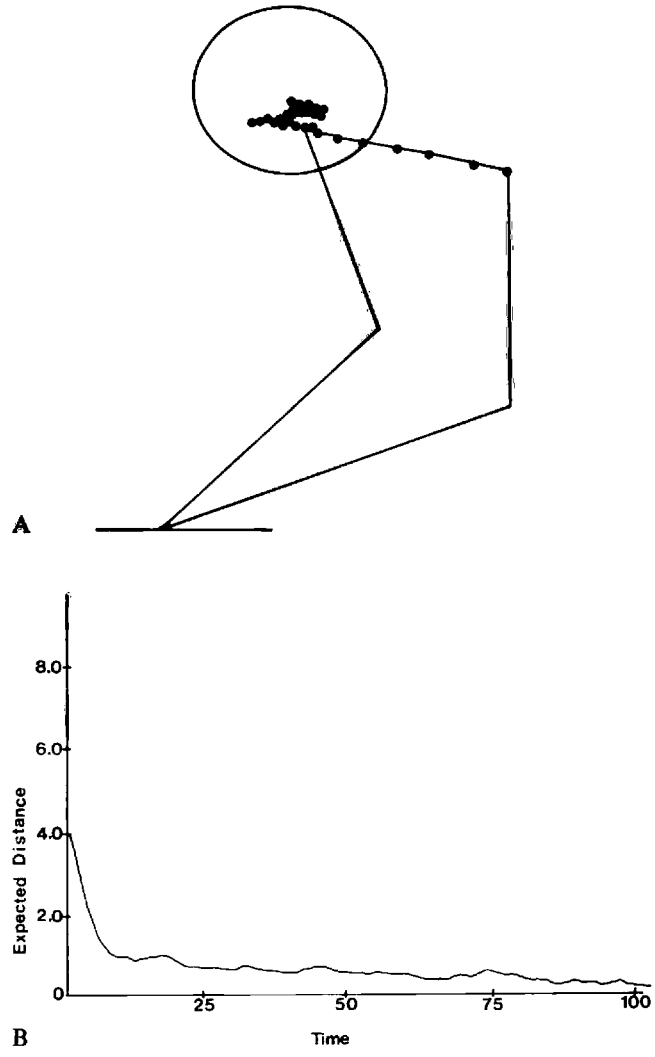

Fig. 4. (a) Expected path (given as a sequence of points) of the two-link Horn's Robot with links of length unity. The control is achieved using the DL_{RP} automaton with R = *3 and* NR = *9, and with a single response controlling both the automata. The standard deviation is 0.1 times the link length. The disk shows the three standard deviation range of noisy points. (b) Expected change in distance from the initial configuration to the final mean configuration for the set-up described in Fig. 4(a).*

The decrease in the Euclidian distance from $\mathbf{P}_f$ as a function of n is shown in Figure 3*B*. The number of iterations required to converge to a point within the three-standard deviation circle of the goal position in this case is approximately 60. Typically, the number of iterations decreases as the number of states of the machine increases, and in this case, the path planning is achieved in as few as 45 iterations for the case when $NR = 8$. Similar graphs are available for the case when $R = 3$. In Figure 4*A* we have shown the expected trajectory taken when $NR = 9$, and the decrease in the Euclidian distance is shown in Figure 4*B*. The power of the scheme is obvious, as the expected trajectory enters the three-standard deviation disk in only 75 iterations for the case when $NR = 12$.

In the case when each automaton had three actions ($R = 3$) and the automata received distinct responses,

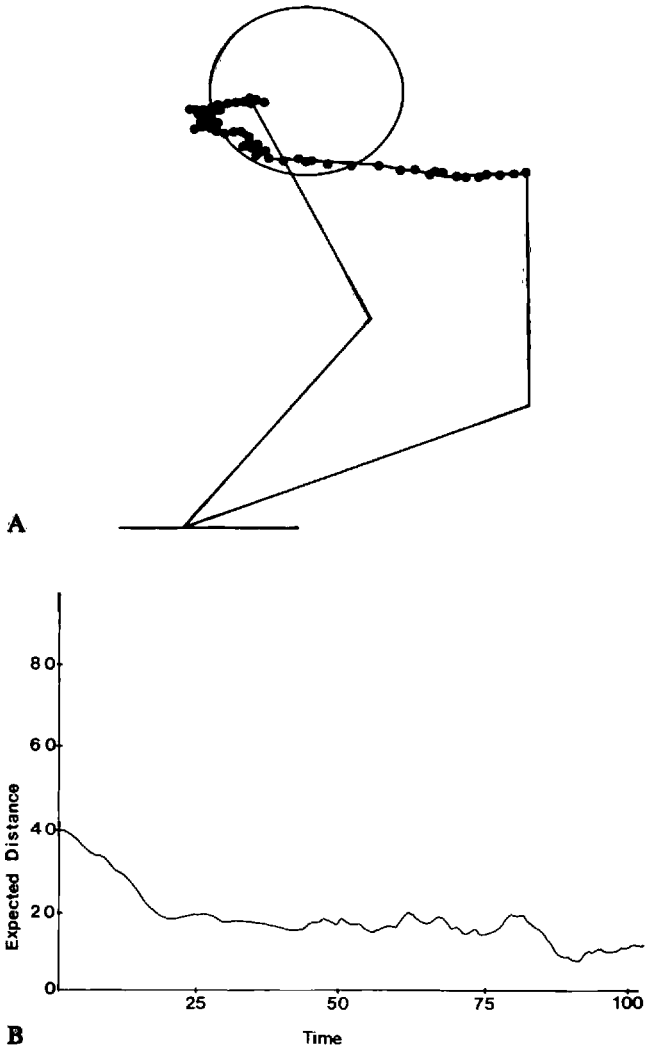

Fig. 5. (a) Expected path (given as a sequence of points) of the two-link Horn's Robot with links of length unity. The control is achieved using the DL_{RP} automaton with R = 3 and NR = 9, and with a separate response controlling each automaton. These responses are obtained using first-order approximations of the inverse kinematics of the robot. The standard deviation is 0.1 times the link length. The disk shows the three standard deviation range of noisy points. (b) Expected change in distance from the initial configuration to the final mean configuration for the set-up described in Fig. 5(a).

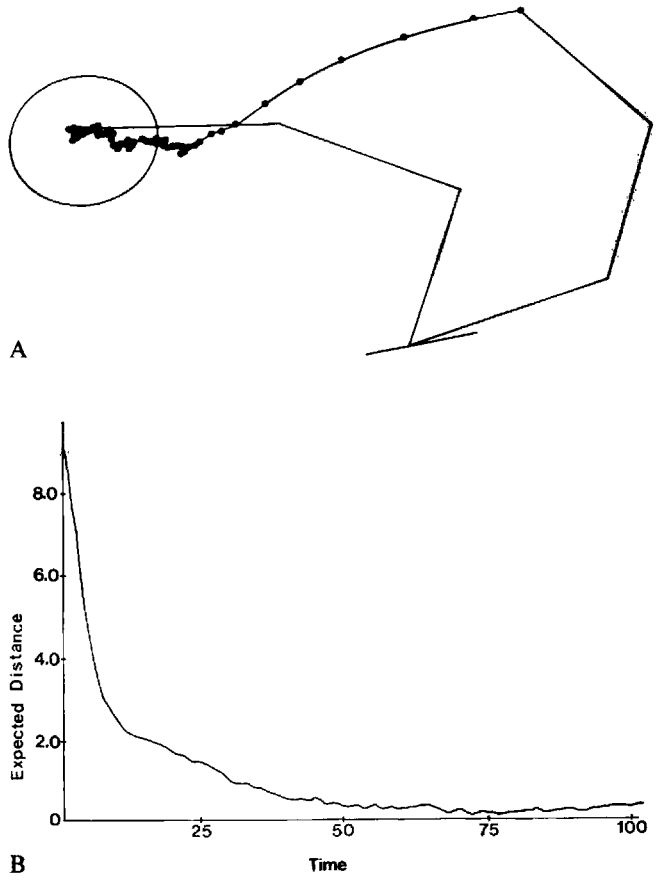

Fig. 6. (a) Expected path (given as a sequence of points) of the three-link Horn's Robot with links of length unity. The control is achieved using the DL_{RP} automaton with R = 3 and NR = 9, and with a single response controlling all the automata. The standard deviation is 0.1 times the link length. The disk shows the three standard deviation range of noisy points. (b) Expected change in distance from the initial configuration to the final mean configuration for the set-up described in Fig. 6(a).

the responses to the automata were obtained by computing the linearized versions of the inverse kinematics. These were obtained at any time instant by solving linear equations involving the "recent history" (i.e., last few Cartesian and joint space coordinates) of the end effector. The performance of the automaton is shown in Figure 5*A* for $NR = 9$. The corresponding decrease in the Euclidian distance from $\mathbf{P}_f$ as a function of n is shown in Figure 5*B*.

Amazingly enough, the different responses do not improve the collective learning capability of the automata, but actually degrade it. (cf. Figs. 4*B* and Figure 5*B*). This is probably because the parameters that characterize the environment (namely the set $\{c_i\}$) are not, in practice, time invariant. Thus they will change as the position and the orientation of the robot changes with regard to the ultimate goal position and orientation. This change in $\{c_i\}$ seems to be accentuated if the *individual* distances of the joints are measured (as opposed to the distance of only the end effector). However, the latter argument has not been justified, and we are unaware of an analytic technique

by which it can be explicitly clarified. It is merely a plausible argument.

Finally, to study the effect of the scheme for a more general robot, we have investigated its performance for the case of the three-link generalization of Horn's Robot. A variety of results have been obtained. Typically, the convergence is shown for the case when $NR = 9$ and $R = 3$ in Figure 6*A*, and the corresponding decrease in the Euclidian distance is shown in Figure 6*B*.

A question may be asked regarding the need for using the DL_{RP} automaton in this application. It has been well known that the updating function of a learning automaton must be dependent on the response it receives from the environment. For example, consider a continuous VSSA that completely ignores the penalty responses of the environment. Such an automaton is of the Reward-Inaction type, and it is well known that there are linear and nonlinear Reward-Inaction schemes that are both absolutely expedient and ϵ-optimal. Apart from the continuous schemes, indeed as shown in section IV of Oommen (1986b), even discretized ϵ-optimal schemes of the Reward-Inaction flavor do exist. However, although the linear symmetric Reward-Penalty schemes are at their best expedient (and definitely *not* absolutely expedient [Narendra and Thathachar 1989]), Reward-Penalty schemes need not be entirely rejected. In this article, we have shown that by discretizing the probability space the resulting symmetric automaton is indeed ϵ-optimal wherever $c_{\min} < 0.5$. Furthermore, apart from being ϵ-optimal, the automaton utilizes *all* the information provided by the environment and thus does not ignore any response as a Reward-Inaction automaton would.

The power of using the scheme suggested in this article is that the sensing mechanism used need not be too sophisticated. A primitive sensor working with the cooperation of such a learning strategy could indeed provide excellent results.

6. Conclusions and Open Problems

In this article we have considered the problem of controlling a manipulator arm in which $\mathbf{P}_i$ the initial position of the end effector is known and the goal position is noisily sensed. The robot is required to move from $\mathbf{P}_i$ to $\mathbf{P}_f$, but instead of having $\mathbf{P}_f$ specified, a series of noisy observations, $\{\mathbf{Q}_f(n)\}$, of $\mathbf{P}_f$ are available.

The solution we propose involves using learning automata. A new learning automaton, the Two-Action DL_{RP} automaton, has been introduced and proven to be ϵ-optimal wherever the minimum penalty probability is less than 0.5. The general R-action DL_{RP} automaton has been shown to be expedient but is conjectured to be ϵ-optimal in a similar environment. A DL_{RP} automaton is stationed at each joint of the robot, and these operate in parallel to control the individual joints of the robot.

Experimental results that demonstrate the power of the scheme for two simple two-dimensional robots have been presented.

We envisage the following future research possibilities:

1. We intend to actually study the power of the scheme for a real-life robot that has very primitive sensing operations.
2. Preliminary simulation results using various learning automata seem to suggest that this strategy could lead to fascinating dog-chase motion strategies that do not involve extensive inverse kinematic solutions. This avenue remains open.
3. We have also recently discovered two new families of discretized Reward-Penalty automata that are ϵ-optimal in all random environments. We hope to study the use of these automata to achieve more effective path planning.

Acknowledgments

The authors are grateful to Prof. I. Reichstein from Carleton University who proofread the final manuscript. Prof. Iyengar would like to thank Dr. Chuck Weisbin at the Oak Ridge National Laboratories for all his support in research activities. This work was partially supported by the Natural Sciences and Engineering Research Council of Canada. The work of Nicte Andrade was supported by the Fondation Yvon Boulanger, Montreal, Canada. A preliminary version of this paper was presented at the 1988 IEEE AI Applications Conference in San Diego, California, March 1988.

Nicte Andrade is no longer at Carleton University but can be contacted at the school's address.

References

Arimoto, S., et al. 1985. Learning control theory for dynamical systems. *Proc. 24th IEEE Conf. on Decision and Control,* pp. 1375–1379.

Arimoto, S., and Takegaki, M. 1981. An adaptive trajectory control of manipulators. *Int. J. Control* 34(2):219–230.

Azadivar, F. 1987. The effect of joint position errors of industrial robots on their performance in manufacturing operations. *IEEE J. Robot. Automat.* 3(2):109–114.

Brady, M., et al. 1983. *Robot Motion: Planning and Control.* Cambridge, Mass.: M.I.T. Press.

Brooks, R. A., and Lozano-Pérez, T. 1985. A subdivision algorithm in configuration space for findpath with rotation. *IEEE Trans. Sys. Man Cybernet.* 15(2):224–233.

Canny, J. F. 1984. Collision detection for moving polyhedra. AI memo 806. Artificial Intelligence Laboratory, M.I.T.

Chatilla, R. 1982 (Torsay, France). Path planning and environment learning in a mobile robot system. *Proc. European Conf. on Artificial Intelligence.*

Chattergy, R. 1985. Some heuristics for the navigation of a robot. *Int. J. Robot. Res.* 4(Spring):59–66.

Craig, J. J., Hsu, P., and Sastry, S. S. 1986 (San Francisco). Adaptive control of mechanical manipulators. *Proc. 1986 IEEE Robotics and Automation Conference,* pp. 190–195.

Crowley, J. L. 1985. Navigation of an intelligent mobile robot. *IEEE J. Robot. Automat.* 1(1):31–41.

Giralt, G., Sobek, R., and Chatilla, R. 1979 (Tokyo). A multilevel planning and navigation system for a mobile robot. *Proc. 6th Int. Joint Conf. Artificial Intelligence,* pp. 335–338.

Gouzenes, L. 1984. Strategies for solving collision-free trajectory problems for mobile and manipulator robots. *Int. J. Robot. Res.* 3(Winter):51–65.

Grossman, D. D., Evans, R. C., and Summers, P. D. 1985. The value of multiple independent robot arms. *Robot. Computer-Integrated Manufact.* 2:135–142.

Hopcroft, J. E., Schwartz, J. T., and Sharir, M. 1984. On the complexity of motion planning for multiple independent objects; PSPACE-hardness of the Waterhouseman's problem. *Int. J. Robot. Res.* 3:76–88.

Iyengar, S. S., et al. 1985a (Miami Beach). Learned navigation paths for a robot in unexplored terrain. *Proc. 2nd Conf. Artificial Intelligence Applications and Engineering of Knowledge Based Systems,* pp. 148–155.

Iyengar, S. S., et al. 1985b. Robot navigation algorithms using learned spatial graphs. ORNL technical report TM-9782. Oak Ridge National Laboratory, Oak Ridge, Tenn.

Kant, K., and Zucker, S. W. 1984. Trajectory planning in time-varying environments; 1:TPP = PPP + VPP. McGill University. Computer Vision and Robotics TR-84-7R. Laboratory, Montreal, Canada.

Koivo, A. J., and Guo, T. 1983. Adaptive linear controller for robotic manipulators. *IEEE Trans. Automatic Control* 28:162–170.

Koivo, A. J. 1986 (San Francisco). Force-position-velocity control with self-tuning for robotic manipulators. *Proc. IEEE Robotics and Automation Conference.* pp. 1563–1568.

Kozlov, A. 1985. Robotics and Model Share Innovations. SIAM News. 18(5). pp. 1, 11.

Lakshmivarahan, S. 1981a. *Learning algorithms: Theory and applications.* Springer-Verlag. New York.*

Lakshmivarahan, S. 1979. ϵ-Optimal learning algorithms—non-absorbing barrier type. Technical report EECS-7901. School of Electrical Engineering and Computing Sciences, University of Oklahoma, Norman, Oklahoma.*

Lakshmivarahan, S. 1981b. Two person decentralized team with incomplete information. *App. Math. Computation* 8:51–78.*

Lakshmivarahan, S., and Thathachar, M. A. L. 1973. Absolutely expedient algorithms for stochastic automata. *IEEE Trans. Sys. Man Cybernet.* 3:281–286.*

Lozano-Pérez, T. 1983. Spatial planning: A configuration space approach. *IEEE Trans. Computers* 32(2):108–120.

Lozano-Pérez, T., and Wesley, M. A. 1979. An algorithm for planning collision-free paths among polyhedral obstacles. *C. ACM* 22(10):560–570.

Miller III, W. T. 1987. Sensor-based control of robotic manipulators using a general learning algorithm. *IEEE J. Robot. Automat.* 3(2):157–165.

Narendra, K. S., and Thathachar, M. A. L. 1989. *Learning Automata.* Englewood Cliffs, N.J.: Prentice-Hall, 1989.*

Narendra, K. S., and Thathachar, M. A. L. 1974. Learning automata—a survey. *IEEE Trans. Sys. Man Cybernet.* 4:323–334.*

Narendra, K. S., and Thathachar, M. A. L. 1980. On the behaviour of a learning automaton in a changing environment with routing applications. *IEEE Trans. Sys. Man Cybernet.* 10:262–269.*

Narendra, K. S., Wright, E., and Mason, L. G. 1977. Applications of learning automata to telephone traffic routing. *IEEE Trans. Sys. Man Cybernet.* 7:785–792.*

Oommen, B. J., and Hansen, E. R. 1984. The Asymptotic optimality of discretized linear reward-inaction learning automata. *IEEE Trans. Sys. Man Cybernet.* 14(3):542–545.*

Oommen, B. J., and Thathachar, M. A. L. 1985. Multiaction learning automata possessing ergodicity of the mean. *Information Sciences* 35(3):183–198.*

Oommen, B. J. 1986b. Absorbing and ergodic discretized two-action learning automata. *IEEE Trans. Sys. Man Cybernet.* 16(2):282–296.*

Oommen, B. J. 1986a. A learning automaton solution to the stochastic minimum spanning cycle problem. *IEEE Trans. Sys. Man Cybernet.* 16(4):598–603.*

Oommen, B. J., et al. 1987. Robot navigation in unknown terrains using learned visibility graphs. Part I: The disjoint convex obstacle case. *IEEE J. of Robot. Automat.* 3(6):672–681.

Oommen, B. J., and Christensen, J. P. R. 1988. Epsilon-optimal discretized linear reward-penalty learning automata. *IEEE Trans. Sys. Man and Cybernetics.* 18(3):451–458.*

Oommen, B. J., and Ma, D. C. Y. 1987 (New Orleans). Fast object partitioning using stochastic learning automata. *Proc. of the 1987 ACM Int. Conf. on Research and Development in Information Retrieval,* pp. 111–112.*

Rao, S. V. N., et al. 1985. Concurrent algorithms for autonomous robot navigation in an unexplored terrain. Tech. Rep. 85-048, Dept. Computer Science, Louisiana State University, Baton Rouge.

Ross, S. M. 1980. *Introduction to Probability Models.* Academic Press. New York.*

Schwartz, J. T., and Sharir, M. 1983. On the piano movers' problem: III. Coordinating the motion of several independent bodies: The special case of circular bodies amidst polygonal barriers. *Int. J. Robot. Res.* 2:46–75.

* References with an asterisk are concerned with learning automata, those without are concerned with robot motion planning.

Schwartz, J. T. and Yap, C-K., Algorithmic and Geometric Aspects of Robotics, Lawrence Erlbaum Associates Publishers, 1987.

Thathachar, M. A. L., and Oommen, B. J. 1979. Discretized reward-inaction learning automata. *J. Cybernet. and Informat. Sci.* 1(Spring):24–29.*

Togai, M., and Yamano, O. 1986 (San Francisco). Learning control and its optimality. *Proc. IEEE Robotics and Automation Conference,* pp. 248–253.

Tsetlin, M. L. 1961. On the behavior of finite automata in random media. *Automat. Telemekh.* (USSR) 22:1345–1354.*

Tsetlin, M. L. 1973. *Automaton theory and the modelling of biological systems.* New York: Academic Press.

Tsypkin, Y. Z., and Poznyak, A. S. 1972. Finite learning automata. *Eng. Cybernet.* 10:478–490.*

Turchen, M. P., and Wong, A. K. C. 1985 (Miami Beach). Low level learning for a mobile robot: Environmental model acquisition. *Proc. 2nd IEEE Conf. Artificial Intelligence Applications,* pp. 156–161.

Udupa, S. M. 1977 (Cambridge, Mass.). Collision detection and avoidance in computer controlled manipulators. *Proc. 5th Int. Conf. Artificial Intelligence,* pp. 737–748.

Varshavskii, V. I., and Vorontsova, I. P. 1963. On the behaviour of stochastic automata with variable structure. *Automat. Telemekh.* (USSR) 24:327–333.*

Whitesides, S. 1985. Computational geometry and motion planning. In Toussaint, G. (ed.): *Computational Geometry.* North Holland.