Machine Learning, Machine Learning, 87–113 (1998)

Hierarchical Learning of Navigational Behaviors in an Autonomous Robot using a Predictive Sparse Distributed Memory*

RAJESH P. N. RAO rao@salk.edu
The Salk Institute, Sloan Center for Theoretical Neurobiology and Computational Neurobiology Laboratory, 10010 N. Torrey Pines Road, La Jolla, CA 92037, USA

OLAC FUENTES fuentes@jsbach.cic.ipn.mx
Centro de Investigación en Computación, Instituto Polítecnico Nacional, Mexico D.F. 07738, Mexico

Editors: Henry Hexmoor and Maja Matarić

Abstract. We describe a general framework for learning perception-based navigational behaviors in autonomous mobile robots. A hierarchical behavior-based decomposition of the control architecture is used to facilitate efficient modular learning. Lower level reactive behaviors such as collision detection and obstacle avoidance are learned using a stochastic hill-climbing method while higher level goal-directed navigation is achieved using a self-organizing sparse distributed memory. The memory is initially trained by teleoperating the robot on a small number of paths within a given domain of interest. During training, the vectors in the sensory space as well as the motor space are continually adapted using a form of competitive learning to yield basis vectors that efficiently span the sensorimotor space. After training, the robot navigates from arbitrary locations to a desired goal location using motor output vectors computed by a saliency-based weighted averaging scheme. The pervasive problem of perceptual aliasing in finite-order Markovian environments is handled by allowing both current as well as the set of immediately preceding perceptual inputs to predict the motor output vector for the current time instant. We describe experimental and simulation results obtained using a mobile robot equipped with bump sensors, photosensors and infrared receivers, navigating within an enclosed obstacle-ridden arena. The results indicate that the method performs successfully in a number of navigational tasks exhibiting varying degrees of perceptual aliasing.

Keywords: sensorimotor learning, autonomous navigation, stochastic hill-climbing, predictive sparse distributed memory, teleoperation, competitive learning, basis vectors, perceptual aliasing

1. Introduction

Goal-directed, collision-free navigation in unstructured environments is an extremely important component of the behavioral repertoire of autonomous mobile robots. Traditional sensorimotor controllers relied on fixed robot behaviors that were pre-wired by the control designer. Such an approach suffered from at least two problems: (a) the increasing complexity of the design process when scaling up from toy-problems to real world environments, and (b) the inherent inflexibility of pre-wired behaviors due to their inability to adapt to circumstances unforeseen at design time. The first problem was addressed by Brooks (1986), who proposed a *hierarchical behavior-based decomposition* of control architectures. Such behavior-based robot architectures have been shown to accomplish a wide

* This work was performed when the authors were affiliated with the Department of Computer Science, University of Rochester, Rochester, NY, USA.

variety of tasks in an efficient manner (Connell, 1990, Maes & Brooks, 1990). The second problem has been addressed by endowing robots with the ability to autonomously learn behaviors, either via direct experimentation and interaction with the environment or via remote teleoperation with the help of a human operator ("teaching-by-showing"). A number of learning algorithms have been utilized for this purpose including backpropagation/recurrent neural networks (Pomerleau, 1989, Pomerleau, 1991, Tani & Fukumura, 1994), self-organizing maps (Nehmzow & Smithers, 1991, Krose & Eecen, 1994), reinforcement learning (Mahadevan & Connell, 1991, Asada et al., 1994, Kaelbling, 1993b), evolutionary algorithms and genetic programming (Beer & Gallagher, 1992, Cliff et al., 1992, Lewis et al., 1992, Fuentes & Nelson, 1997), and hill-climbing routines (Pierce & Kuipers, 1991, Fuentes et al., 1995).

In the context of robot navigation, a popular approach has been the construction and use of global maps of the environment (Elfes, 1987). Such an approach, while being intuitively appealing, faces the difficult problem of robot localization *i.e.*, establishing reliable correspondences between noisy sensor readings and geometrical map information in order to estimate current map position. In addition, the global map inherits many of the undesirable properties of centralized representations that Brooks identifies in traditional robot controllers (see (Brooks, 1986) for more details). An alternative behavior-based approach to navigation was proposed by Mataric (Mataric, 1992) (see also (de Bourcier, 1993, Kuipers & Byun, 1988)). This method avoids the problems involved in creating and maintaining global representations by utilizing only local information as provided by landmarks along a navigational path. A potential problem with such landmark-based approaches is that incorrect landmark matches or changes in the landmarks themselves might result in catastrophic failure of the navigational system.

In this paper, we propose an adaptive behavior-based approach to autonomous mobile robot navigation. The approach is based on a hierarchical control architecture that uses a form of stochastic "hill-climbing" (Pierce & Kuipers, 1991, Fuentes et al., 1995) for learning lower level reactive behaviors and a predictive sparse distributed memory (Kanerva, 1988, Rao & Fuentes, 1996) for learning higher level behaviors pertaining to goal-directed navigation. A "teaching-by-showing" paradigm is adopted to initially train the sparse distributed memory for achieving a prespecified navigational behavior; such an approach drastically cuts down on the number of training trials needed as compared to learning by trial and error or by random exploration of the environment. The robot is teleoperated (via a remote joystick) on a small number of training paths within the given environment (in the robot simulation environment, this is done via a point-and-click user interface). During training, the vectors in the perception space as well as the motor space are adapted using a form of competitive learning that develops basis vectors that efficiently span the sensorimotor space encountered by the robot. After training, the robot navigates from arbitrary locations to a desired goal location based on motor output vectors computed by indexing into sensorimotor memory using present as well as past perceptions and by employing a saliency-based weighted averaging scheme for interpolating output vectors. By using past sensory inputs to provide the necessary context for disambiguating potentially similar perceptions, we alleviate the well-known problem of perceptual aliasing (Whitehead & Ballard, 1991, Chrisman, 1992, McCallum, 1996) in finite-order Markovian environments. We provide both experimental and simulation results for a mobile robot

equipped with bump sensors, photosensors and infrared receivers, navigating within an enclosed obstacle-ridden arena. The method is shown to perform successfully in a variety of navigational tasks involving varying degrees of perceptual aliasing.

2. Hierarchical Behavior-Based Control and Modular Learning

Traditionally, the task of developing a sensorimotor control architecture for a situated autonomous robot was left to the human programmer of the robot. Pre-wiring robot behaviors by hand however becomes increasingly complex for robots with large numbers of sensors and effectors, especially when they are performing sophisticated tasks that involve continuous interaction with the encompassing environment. For example, consider the task of learning to navigate between two points in an obstacle-filled environment. The task involves finding a mapping from an n-dimensional sensory space (outputs of various bump sensors, photosensors, infrared receivers, etc.) to a k-dimensional motor space (steering commands, speed, etc.) that optimizes the robot's performance in terms of, for instance, reaching the destination in a reasonable amount of time or on a minimally circuitous route. Since the number of different perceptions the robot may encounter grows exponentially with the number of sensors it possesses and the number of motor outputs available (2^n and 2^k in the binary case), the task of hardwiring behaviors becomes extremely cumbersome and error-prone in realistic and unstructured environments. The large dimensionality of the sensorimotor space also causes problems for approaches that involve learning monolithic control architectures for a given task. Learning tends to be substantially slow due to the need for extensive sampling of the data space in order to ensure accurate estimation.

An attractive way of circumventing this "curse of dimensionality" is to divide the given task into several layers of control such that the first layer consists of an elementary behavior (such as avoiding continuous contact with obstacles), with each subsequent layer building upon the performance obtained by the previous ones. If one chooses the partitions carefully, it may be possible to arrange for the first layer to use only a subset of the sensors available and for each subsequent layer to use a superset of the sensors used by the lower layers. This is essentially Brooks' well-known subsumption architecture (Brooks, 1986). Figure 1 illustrates this idea for a simple three-level hierarchical control architecture used in our experiments. The lower two levels implement the behaviors of collision detection (using bump sensors) and obstacle avoidance (using photosensors) while the higher level implements goal-driven navigation (using infrared receivers and photosensors) towards a specific goal location in the environment. Behaviors at all three hierarchical levels are learned by the robot as described in the following sections.

A hierarchical partitioning of the sensorimotor space facilitates *modular learning* by allowing the robot to learn the sensorimotor mapping corresponding to each layer independent of the other layers. Thus, different learning algorithms may be employed at the different hierarchical levels, as illustrated in this paper. In addition, a hierarchical partitioning greatly reduces the search space and often results in faster convergence of the learning routines, as observed in multigrid-based numerical optimization methods (Hackbusch, 1985) and in hierarchical reinforcement learning algorithms (Singh, 1992, Dayan & Hinton, 1993, Kaelbling, 1993a). Faster learning of the constituent

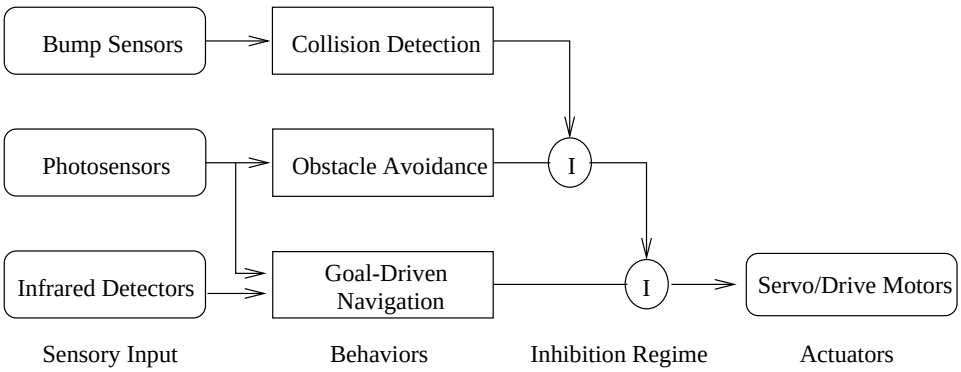

Figure 1. **Block Diagram of the Robot Control Architecture**. The lower level behaviors occur at the top of the diagram and can inhibit the higher level(s) if the current perception vector satisfies their criterion for execution. "I" denotes inhibition of the current behavior by a lower level behavior.

behaviors in turn permits faster learning of the overall behavior that was desired of the robot.

3. Learning Lower Level Behaviors Using Stochastic Hill-Climbing

To learn the lower level "reactive" behaviors such as collision detection and obstacle avoidance, we use a relatively simple stochastic "hill-climbing" technique that allows the robot to learn these behaviors via direct experimentation and interaction with the environment (see also (Pierce & Kuipers, 1991)). The algorithm is similar to some reinforcement learning techniques such as Q-learning (see (Kaelbling et al., 1996) for a review), and its use in learning robot navigation was first investigated in (Fuentes et al., 1995). We briefly review the essentials of the algorithm in this section.

Let $\mathcal{A}$ be the set of discrete-valued actions that the robot can perform, let $\mathcal{P}$ be the set of discrete-valued perceptions that the robot can obtain from its sensors. A policy is a mapping $m(p) : \mathcal{P} \rightarrow \mathcal{A}$ that defines the action $m(p) \in \mathcal{A}$ to be taken when confronted with the sensory stimulus $p \in \mathcal{P}$.

To every perception p we also assign a numeric value v_p measuring the desirability or "goodness" of the situations where that perception normally occurs. For example, a perceptual input indicating one or more depressed bump sensors has a low v, since it occurs in an undesirable situation (analogous to "pain" if interpreted liberally) where the robot has crashed into an obstacle, while the perception of having no bump sensors depressed will have a higher v, since it indicates that the robot is clear.

The task is to learn a policy m that will take the robot from "bad" to "good" perceptions and maintain it in good perceptions when they are found. The robot achieves this by computing a heuristic value $h(p)$ that measures how often, on average, the action taken in situation p has resulted in perceptions that are more desirable than p. For every *perception-action* pair in the current policy, we keep a heuristic value h and replace, via stochastic sampling of

the action space, those actions in the policy that are judged to be inadequate *i.e.*, for which h falls under a prespecified threshold.

The stochastic hill-climbing learning algorithm can be summarized as follows:

1. Randomly initialize m
2. Initialize heuristic value and occurrence counter
   $(\forall p \in \mathcal{P})h(p) = 0, n(p) = 0$
3. Repeat until the change in $h(p)$ is negligible for all $h(p)$:
   (A) Get perceptual input p from sensors
   (B) Perform action $m(p)$
   (C) Get resulting perceptual input r from sensors
   (D) Adjust heuristic value
   $h(p) = \frac{n(p)}{n(p)+1}h(p) + \frac{1}{n(p)+1}(\alpha(v_r - v_p) + \beta v_r)$
   (α and β are positive constants that determine the weight given to the relative change in "goodness" $(v_r - v_p)$ and the current goodness value v_r respectively).
   (E) Update occurrence counter
   $n(p) = n(p) + 1$
   (F) if $h(p) <$ *threshold(p)* replace $m(p)$ by a randomly chosen action $q \in \mathcal{A}$ and reinitialize $h(p)$ and $n(p)$

As a concrete example, in the case of the collision detection level of the control hierarchy in Figure 1, one may use $v_p = -1$ if p represents an undesirable perception where the robot has encountered an obstacle (*i.e.*, any of the bump sensors are depressed) and $v_p = 1$ otherwise. Experimental results obtained using this method are presented in Section 7.

4. Sparse Distributed Memory for Goal-Directed Navigation

The previous section described a method for learning low level reactive behaviors by storing in memory a set of perception-action pairs that best implement a particular behavior. Such a strategy can also be applied to the higher level task of memory-based navigation from an initial location to a prespecified goal location in the environment. For example, Nelson uses an associative memory in the form of a look-up table that associates a large set of perception vectors from different locations with a corresponding set of actions that allow navigation to a desired location (Nelson, 1991). However, such an approach suffers from at least three drawbacks: (1) the address space formed by the perception vectors is usually quite large and therefore, storing fixed reference vectors for every possible scenario becomes infeasible, (2) the training time increases drastically as the look-up table size increases, and (3) simple look-up table strategies relying on nearest-neighbor methods usually fail to generate the appropriate responses to novel situations. In this section, we address problems (1) and (2) by using only a *sparse* subset of the perceptual address space. This naturally leads to a memory known as sparse distributed memory (SDM) that was originally proposed by Kanerva (Kanerva, 1988). We address problem (3) of generalization in novel scenarios in Section 5 where we propose a modified form of SDM that uses competitive learning to adapt its sensorimotor space and radial interpolation to compute motor output vectors.

4.1. Kanerva's Model

Sparse distributed memory (SDM) (Figure 2) was first proposed by Kanerva as a model of human long-term memory (Kanerva, 1988). It is based on the crucial observation that if concepts or objects of interest are represented by high-dimensional vectors, they can benefit from the very favorable matching properties caused by the inherent tendency toward orthogonality in high-dimensional spaces.

Kanerva's SDM can be regarded as a generalized random-access memory wherein the memory addresses and data words come from high-dimensional vector spaces. As in a conventional random-access memory, there exists an array of storage locations, each identified by a number (the address of the location) with associated data being stored in these locations. However, unlike conventional random-access memories which are usually concerned with addresses only about 20 bits long (memory size = 2^{20} locations) and data words only about 32 bits long, SDM is designed to work with address and data vectors with much larger dimensions. Due to the astronomical size of the vector space spanned by the address vectors, it is physically impossible to build a memory containing every possible location of this space. However, it is also unnecessary since only a subset of the locations will ever be used in any application. This provides the primary motivation for Kanerva's model: *only a sparse subset of the address space is used for identifying data locations and input addresses are not required to match stored addresses exactly but to only lie within a specified distance of an address to activate that address*.

The basic operation of SDM as proposed by Kanerva can be summarized as follows:

1. **Initialization**: The physical locations in SDM correspond to the rows of an $m \times k$ contents matrix $\mathbf{C}$ (initially filled with zeroes) in which bipolar data vectors $\in \{-1, 1\}^k$ are to be stored (see Figure 2 (a)). Pick m unique addresses (p-element binary vectors $\mathbf{r}_i$) uniformly at random for each of these locations (these addresses are represented by the matrix $\mathbf{A}$ in Figure 2 (a)).

2. **Data Storage**: Given a p-element binary address vector $\mathbf{r}$ and a k-element bipolar data vector $\mathbf{d}$ for storage, select all storage locations whose addresses lie within a Hamming distance of D from $\mathbf{a}$ (these activated locations are given by the *select* vector $\mathbf{s}$ in Figure 2 (a)). *Add* the data vector $\mathbf{d}$ to the previous contents of each of the selected row vectors of $\mathbf{C}$. Note that this is different from a conventional memory where addresses need to exactly match and previous contents are overwritten with new data.

3. **Data Retrieval**: Given a p-element binary address vector $\mathbf{r}$, select all storage locations whose addresses lie within a Hamming distance of D from $\mathbf{r}$ (these locations are again given by the vector $\mathbf{s}$). Add the values of these selected locations in parallel (vector addition) to yield a sum vector $\mathbf{S}$ containing the k sums. Threshold these k sums at 0 to obtain the data vector $\mathbf{d}'$ *i.e.*, $d_i = 1$ if $s_i > 0$ and $d_i = -1$ otherwise.

Note that the addition step in (2) above is essentially a *Hebbian learning rule*. The statistically reconstructed data vector $\mathbf{d}'$ should be the same as the original data vector provided the *capacity* of the SDM (Kanerva, 1993) has not been exceeded. The intuitive reason for this is as follows: When storing a data vector $\mathbf{d}$ using a p-dimensional address vector $\mathbf{r}$, each of the selected locations receives one copy of the data. During retrieval with an address

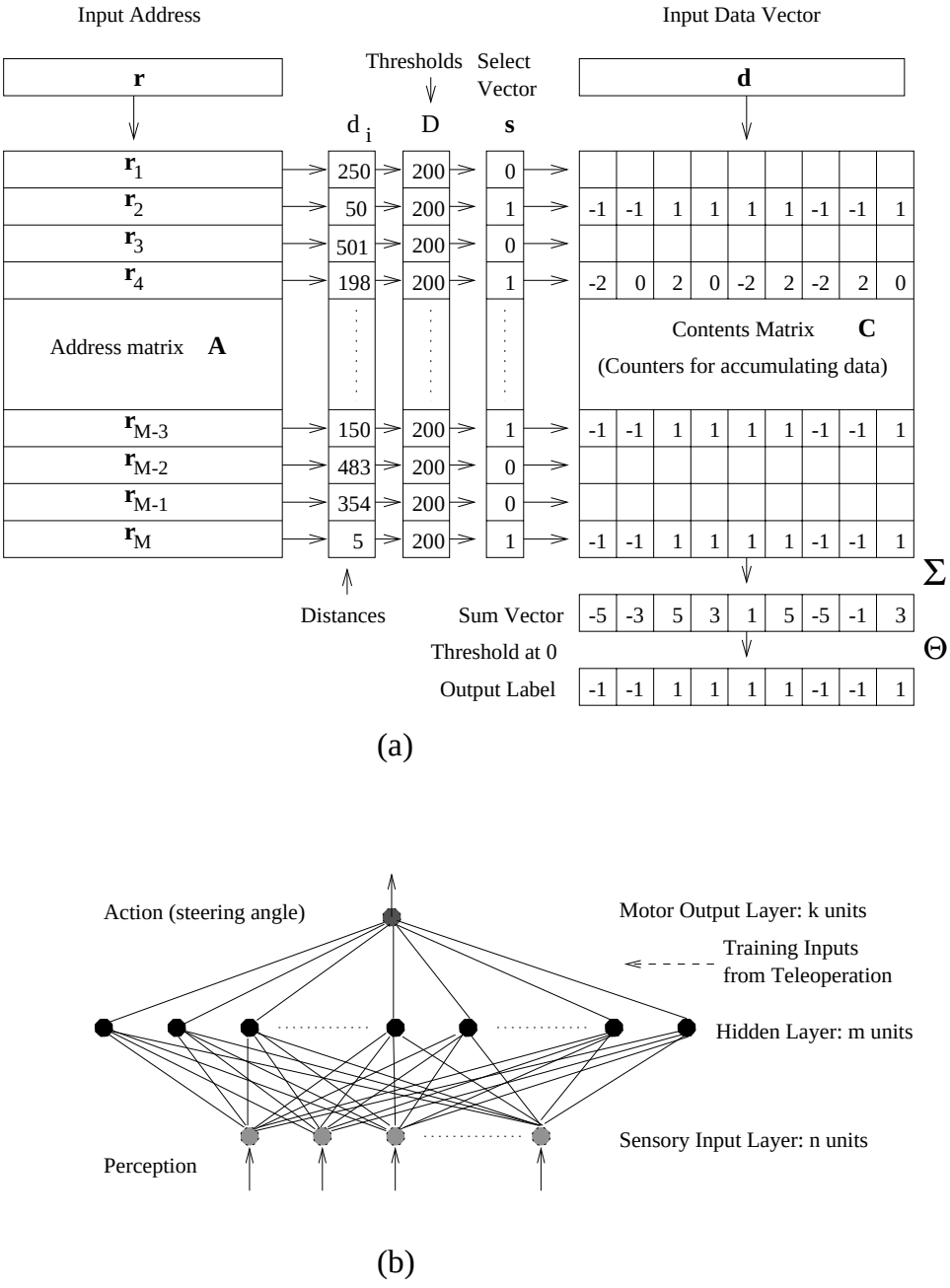

Figure 2. **Kanerva's Sparse Distributed Memory (SDM)**. (a) shows a schematic depiction of the model as proposed by Kanerva for storage of binary (bipolar) vectors. (b) shows a realization of the memory in the form of a three-layered fully connected network. The labels describe how the network can be used for perception-based navigation after suitable modifications (see Section 5).

close to $\mathbf{r}$, say $\mathbf{r}'$, *most* of the locations that were selected with $\mathbf{r}$ are also selected with $\mathbf{r}'$. Thus, the sum vector contains most of the copies of $\mathbf{d}$, plus copies of other different words; however, due to the orthogonality of the address space for large p, these extraneous copies are much fewer than the number of copies of $\mathbf{d}$. This biases the sum vector in the direction of $\mathbf{d}$ and hence, $\mathbf{d}$ is output with high probability. A more rigorous argument based on signal-to-noise ratio analysis can be found in (Kanerva, 1993).

4.2. Limitations of Kanerva's Model

The model of sparse distributed memory as originally proposed by Kanerva has several weaknesses that prevent its direct use in memory-based navigation:

- Both the address and data vectors are required to be binary in the standard model of the SDM. Since most natural environments yield multivalued input patterns, we must either modify the indexing mechanisms of the model or recode all inputs into binary form. We chose the former option since the latter sacrifices the interpolation properties of the memory.
- The standard model of the SDM assumes a *uniform distribution* for the input address vectors whereas in most natural circumstances, the input address vectors tend to be clustered in many correlated groups distributed over a large portion of the multidimensional address space. Therefore, if addresses are picked randomly as suggested by Kanerva, a large number of locations will never be activated while a number of locations will be selected so often that their contents will resemble noise. We remedy this situation by allowing the input address space to *self-organize* according to the sensorimotor input distribution.
- The standard SDM uses a single fixed threshold D for activating address locations. While this simplifies the analysis of the memory considerably, it also results in poor performance since the actual values of the distances between an input address vector and the basis address vectors in the SDM are lost during quantization to 0 or 1. Our model uses *radial interpolation* functions that weight corresponding data vectors according to the address vector's closeness to the input vector.

5. A Self-Organizing SDM for Goal-Directed Navigation

In the following, we describe the operation of a modified form of SDM that is suitable for memory-based goal-directed navigation. The memory can be realized as a three-layer fully-connected feedforward network as shown in Figure 2 (b). Assume the memory contains m storage locations. The first layer consists of n units representing the input perception vector $\mathbf{p}$. The hidden layer consists of m units while the output layer is represented by k units. Let $\mathbf{w}_i$ ($1 \leq i \leq m$) represent the vector of weights between hidden unit i and the input layer, and let $\mathbf{m}_i$ represent the vector of weights between hidden unit i and the output layer. The memory accepts multivalued perception vectors $\mathbf{p}$ from an arbitrary distribution and stores an associated multivalued motor vector $\mathbf{m}$ in a distributed manner in the data space.

5.1. Initialization

Pick m unique addresses (n-dimensional vectors $\mathbf{p}_i$) at random for each of the locations. This corresponds to randomly initializing the input-hidden weight vectors $\mathbf{w}_i$ ($1 \leq i \leq m$).

5.2. Competitive Learning of Sensorimotor Basis Functions

Given an input perception vector $\mathbf{p}$ and an associated motor vector $\mathbf{m}$ during the training phase, we self-organize the address and data space using a *soft competitive learning rule* (Nowlan, 1990, Yair et al., 1992):

1. Calculate the Euclidean distances $d_j = \|\mathbf{w}_j^t - \mathbf{p}\|$ between $\mathbf{w}_j^t$ and the input perception vector $\mathbf{p}$.

2. Adapt all weight vectors (sensory address space vectors) according to:

$$\mathbf{w}_j^{t+1} \leftarrow \mathbf{w}_j^t + g_j(t) P_t(d_j)(\mathbf{p} - \mathbf{w}_j^t) \tag{1}$$

   where P_t is defined as:

$$P_t(d_j) = \frac{e^{-d_j^2/\lambda_j(t)}}{\sum_{i=1}^{m} e^{-d_i^2/\lambda_i(t)}} \tag{2}$$

   and g_j is given by $g_j(t) = 1/n_j(t)$ where the counter:

$$n_j(t+1) = n_j(t) + P_t(d_j) \tag{3}$$

   $P_t(d_j)$ can be interpreted as the probability of the prototype vector $\mathbf{w}_j$ winning the current competition for perception $\mathbf{p}$. Note that the probability vector $\mathbf{P}$ obtained by vectorizing the $P_t(d_j)$ for $1 \leq j \leq m$ is the equivalent of the *select* vector $\mathbf{s}$ in Kanerva's model (Figure 2 (a)).

   The "temperature" parameter $\lambda_j(t)$ is gradually decreased to a small value in a manner reminiscent of simulated annealing. This causes the learning algorithm to evolve from an initially soft form of competition with a large number of winners to the case where only a *sparse* number of winners exist for any given input vector. The soft competition in the beginning tunes the initially random prototype vectors towards the input sensory space (thereby preventing the occurrence of "dead" units which never get updated) while the later existence of sparse number of winners helps in fine tuning the prototype vectors to form a set of distributed *basis vectors* spanning the sensory input space.

3. Given a training motor input $\mathbf{m}$, adapt the prototype vectors stored in the motor space according to Equation 1 using the *same* values for d_j as in 1 above (*i.e.*, distance between the input perception vector and the sensory basis vectors $\mathbf{w}_j^t$):

$$\mathbf{m}_j^{t+1} \leftarrow \mathbf{m}_j^t + \beta_j(t) P_t(d_j)(\mathbf{m} - \mathbf{m}_j^t) \tag{4}$$

where $\beta_j(t)$ ($0 < \beta_j(t) < 1$) is a gain function. Note that $\beta_j(t)$ *does not necessarily* have to decrease with time (this reinforcement strength could be made to depend on other factors such as importance to the animate system as evaluated by other modalities or other internal value mechanisms).

5.3. Computing Motor Output Vectors

Following training, the memory *interpolates* between the stored motor basis vectors to produce a motor output vector $\mathbf{o}$ for a given perception vector $\mathbf{p}$, as follows:

1. Calculate the Euclidean distances d_j between $\mathbf{w}_j$ and the input perception vector $\mathbf{p}$.
2. Let m_{ji} ($1 \leq i \leq k$) denote the weight from hidden unit j to output unit i. Then, the ith component of the reconstructed output vector $\mathbf{o}$ (in other words, the output of the output unit i) is given by:

$$o_i = \sum_{j=1}^{m} P_t(d_j) m_{ji} \tag{5}$$

The saliency-based weighted averaging above is a form of normalized radial interpolation that is similar to the output operation of *radial basis function* (RBF) networks (Broomhead & Lowe, 1988, Poggio & Girosi, 1990): the closer the current perception is to a given sensory basis vector, the more "salient" that memory location and the more the weight assigned to the motor vector associated with that basis vector. The above scheme is inspired by recent neurophysiological evidence (McIlwain, 1991) that the *superior colliculus*, a multilayered neuronal structure in the brain stem that is known to play a crucial role in the generation of saccadic eye movements, in fact employs a weighted population averaging scheme to compute saccadic motor vectors. We refer the interested reader to an earlier paper (Rao & Ballard, 1995b) for a closely related method for visuomotor learning of saccadic eye movements in a robot head.

6. Predictive Sensorimotor Memory

The self-organizing SDM described in the preceding section has been shown to be useful for perceptual homing by an autonomous mobile robot (Rao & Fuentes, 1995). However, it is not hard to see that the above method is limited to only those navigational behaviors that can be modeled as (zeroth-order) *Markov* processes where the current motor output depends solely on the current perception and past inputs are treated as irrelevant for determining the current action. In general environments, however, a zeroth-order Markovian assumption is often inappropriate, causing methods that rely on such an assumption to suffer from the problem of *perceptual aliasing* or *hidden state* (Chrisman, 1992, Whitehead & Ballard, 1991, McCallum, 1996).

6.1. Perceptual Aliasing

Perceptual aliasing, a term coined in (Whitehead & Ballard, 1991), refers to the situation wherein two or more identical perceptual inputs require different responses from an autonomous system. A number of factors such as limited sensing capability, noisy sensory inputs, and restricted resolution in addition to inherent local ambiguity in typical environments contribute towards exacerbating perceptual aliasing. The effects of aliasing can be reduced to some extent by incorporating additional sensory information that suffice to disambiguate between any two given situations (Whitehead & Ballard, 1991, Wixson, 1991). However, these methods still rely only on current percepts and thus, are unable to overcome aliasing in environments that are not zeroth-order Markovian. An alternate approach that is tailored toward modeling finite-order Markovian sensorimotor processes is to base the current output on both the current as well as the sequence of past inputs within a certain time window whose size depends on the order of the Markov process to be modeled. In other words, the current percept not only predicts the action for the current time instant t but also actions for future time instants $t, t+1, \ldots, t+s$, where s is a parameter determining the amount of "short-term memory" available for storing predicted actions (or equivalently, the length of the perceptual history window). Its value can be either predetermined or adapted on-line to counteract the effects of aliasing. The motor action for the current time instant t is then determined by a weighted average of the actions recommended by the current as well as past perceptions upto time $t-s$. This solution shares some similarities with Kanerva's s-fold memory for storage and retrieval of sequences (Kanerva, 1988); however, all the weaknesses inherent in Kanerva's original model (Section 4.2) still apply to the s-fold memories, thus preventing their direct application in countering the aliasing problem.

6.2. Using Past Perceptions for Motor Prediction

The informal solution for perceptual aliasing sketched in the previous section can be formalized as follows. Let $\mathbf{p}_t$ be the current perception vector and let $\mathbf{m}$ be the current motor action vector. Figure 3 shows the predictive form of the sparse distributed memory. Note that the motor space now contains additional sets of connections between the hidden and output layer, each such set determining the action to be executed at a given time in the future. A further set of connections (see Figure 3 (b)), some of which involve time delays of different durations, appropriately combine the outputs of this intermediate layer to yield an estimate of current motor output. The operation of the predictive memory is as follows:

- **Training**: During training (Figure 3 (a)), the perception space is self-organized as given by Equation 1. The motor space is also self-organized as in Equation 4 but using the *current* motor vector as the training signal for *each* set of hidden to motor output unit connections, with the constraint that the probability weight vector $\mathbf{P}$ (see Section 5.2) for the set of connections determining the action for time $t+i$ is the one that was obtained by indexing into the sensory address space using the perception at time $t-i$. Such a training paradigm ensures that after training, the perception at time t generates predicted actions for time steps $t, t+1, \ldots, t+s$.

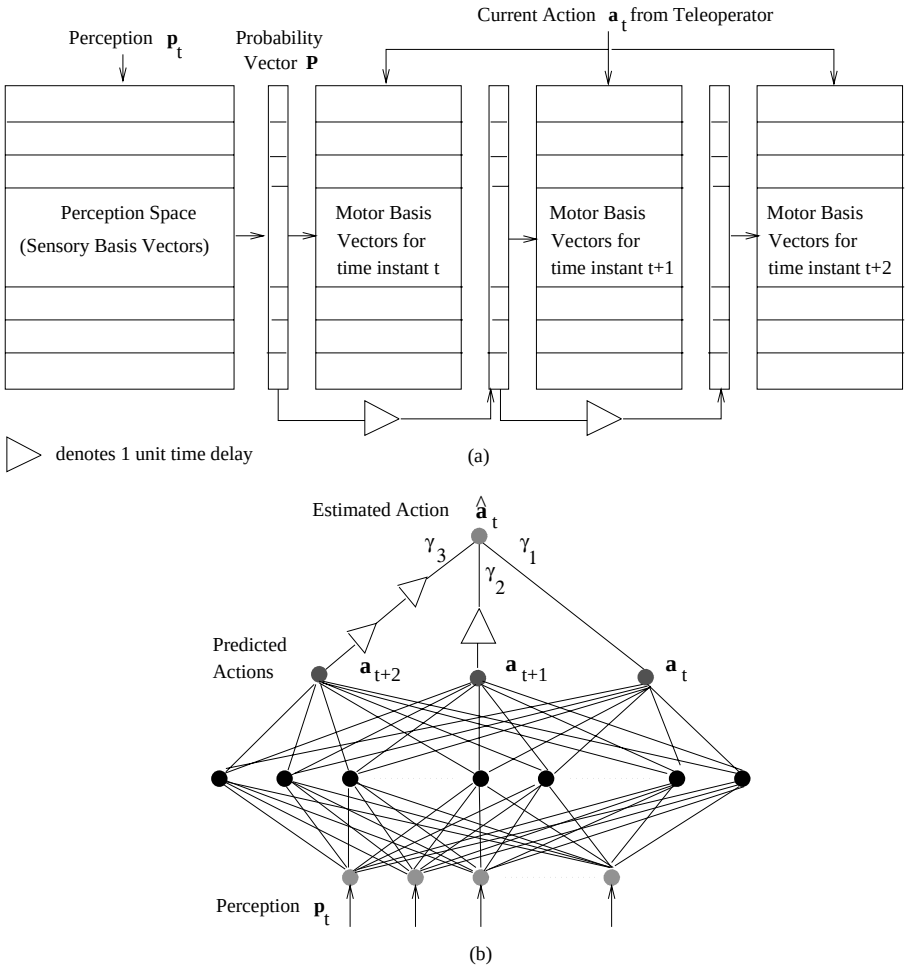

Figure 3. **Predictive Sparse Distributed Memory.** (a) shows the procedure of training the predictive memory. The current motor input $\mathbf{a}_t$ from the teleoperator is fed to each of the motor memories which are indexed using the probability vector $\mathbf{P}$ from the current perception (left), the previous one (middle), and the perception before the previous one (right). This allows the memory to generate predictions of motor output for the next two time steps in addition to the current one during the navigation phase. (b) shows the memory (in its network form) during the navigation phase. The current estimate of motor output $\hat{\mathbf{a}}_t$ is computed by averaging over the motor action vectors for the current time instant t predicted by current input, the previous input and the input before the previous one (this is achieved using zero, one and two delay units (triangle symbols) respectively).

- **Navigation**: During the autonomous navigation phase (Figure 3 (b)), the current perception yields a set of predicted actions $\mathbf{a}_t, \mathbf{a}_{t+1}, \ldots, \mathbf{a}_{t+s}$. Thus, at any given time instant t $(t \geq s)$, there exist s estimates $\mathbf{a}_t^1, \mathbf{a}_t^2, \ldots, \mathbf{a}_t^s$ of what the current action should be, based on both current and past sensory inputs. For example, in Figure 3 (b), $\mathbf{a}_t^1$ and $\mathbf{a}_t^2$ are the values $\mathbf{a}_{t+1}$ and $\mathbf{a}_{t+2}$ from one and two time steps ago respectively.

Destination Locations

Source Locations	One	Many
One	Simple Interlocation Navigation	Example: Foraging
Many	Example: Homing	General Navigation

Figure 4. **A Hierarchical Decomposition of Navigational Behaviors**. The classification is based on the number of possible source locations and the number of destinations allowed. The experiments investigated the robot's ability to acquire navigational behaviors shown in the three classes other than the one-many class.

We obtain the combined estimate of current motor output using the following weighted averaging method:

$$\hat{\mathbf{a}}_t = \sum_{i=1}^{s} \gamma_i(t)\mathbf{a}_t^i \tag{6}$$

where $\gamma_i(t)$ is the weight accorded to the predicted estimate i of the current action. For the experiments in this paper, the weights γ_i were hand-picked, but a better strategy that remains a subject of ongoing research is to estimate them on-line based on current and past perceptions.

7. Experimental Results

The methods proposed in the previous sections were tested in the context of a general classification of navigational behaviors as depicted in Figure 4. A custom-built three-wheeled mobile robot (Figure 5) was used. The robot's body was constructed using parts from a standard $Meccano/Erector$ set. The size of the robot was quite modest, measuring approximately 1 foot high, with length and width approximately 1 foot and one half of a foot respectively. The robot was equipped with three classes of sensors:

- **Bump Sensors**: Digital microswitches (with attached flexible metallic flaps) were used as bump sensors to indicate whether the robot was physically touching an obstacle. Five of these sensors, placed at different locations on the robot's body, were used for the *collision detection* behavior. The output of these sensors was either 1 (switch depressed) or 0 (switch not depressed).
- **Photosensors**: Six tilted photosensors (implemented via shielded photoresistors) were used for detecting obstacles lying ahead and also for measuring light intensity value due to the color of the floor during memory-based navigation. In addition, six horizontally-positioned photosensors were employed for measuring the amount of light from a light

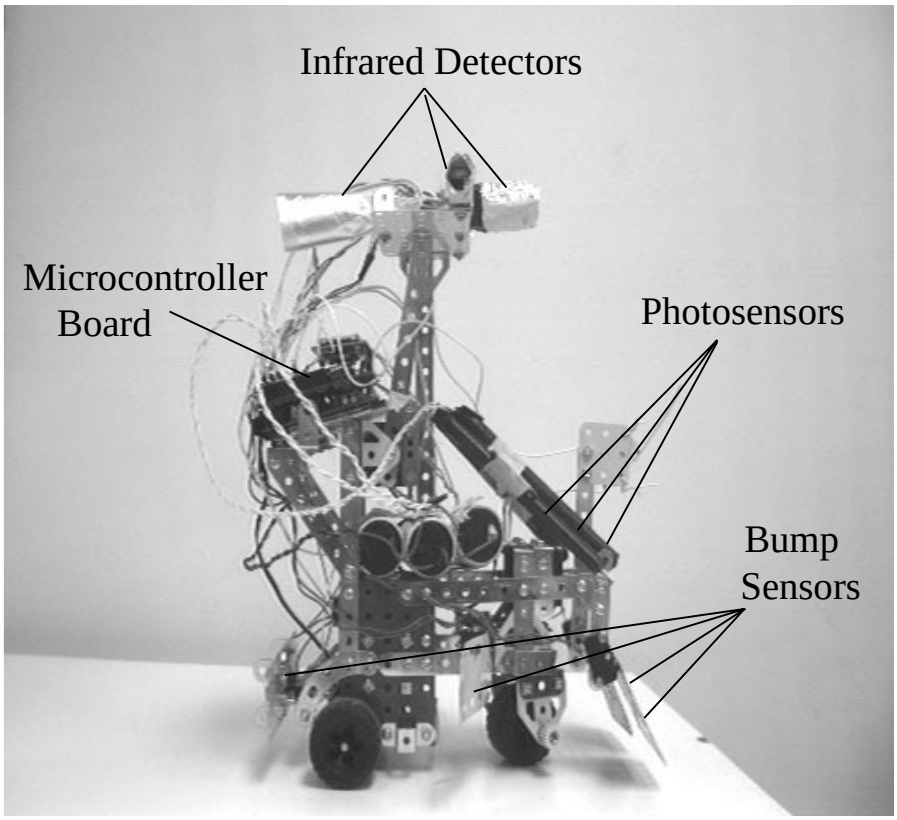

Figure 5. **The Mobile Robot used in the Experiments and Simulations.** The chassis of the three-wheeled robot was built from $Meccano/Erector$ parts. The robot carries its own power supply in the form of three rechargeable DC batteries (seen in the center) for driving the rear motor and a separate set of AA batteries for powering the Motorola 68HC11 microcontroller board and the front steering motor.

source in the environment. The light source was used to enhance the richness of the perception vector when learning to navigate using a sparse distributed memory. The outputs of the photosensors were integers in the range 0-255, the higher values denoting greater light intensity. These sensors were used for the *obstacle avoidance* behavior as well as for indexing into memory during memory-based navigational behaviors.

- **Infrared Detectors**: These sensors, when used in conjunction with infrared detection software, indicate the strength (along the line of sight of the sensor) of modulated infrared light being emitted from a source located a few feet away (see below). The output values of these sensors were again integers in the range 0-255. However, unlike the photosensors, the infrared response was generally quite non-linear and almost binary, alternating between very high and very low values depending on the orientation of the sensor with respect to the infrared source. Four of these sensors were used in

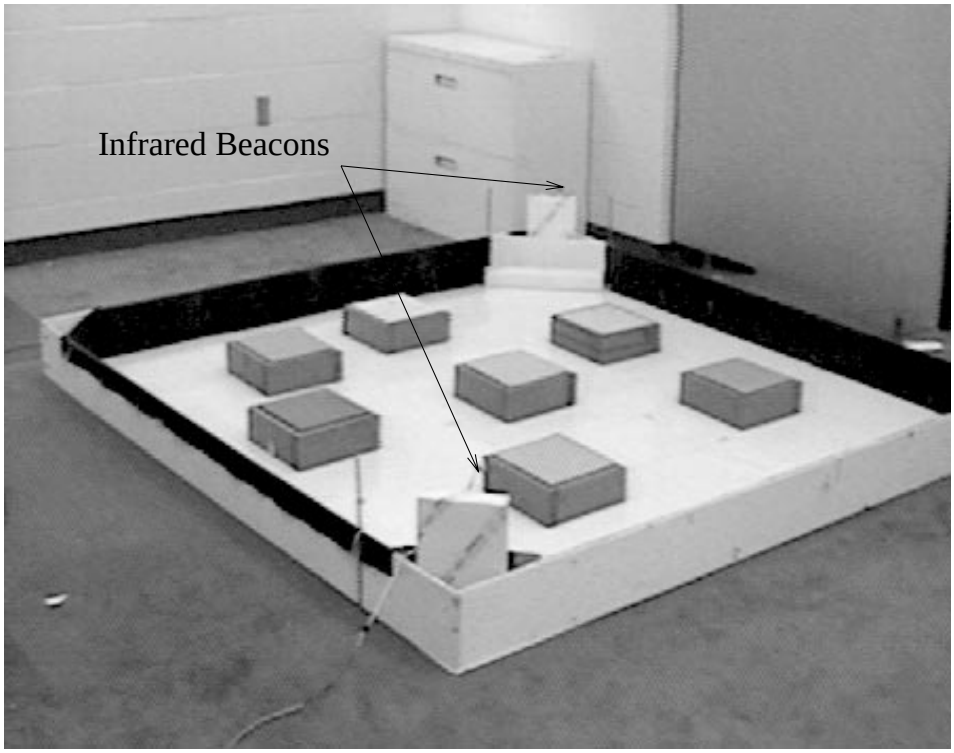

Figure 6. **The Environment of the Robot**. The robot was tested in an enclosed arena of size eight feet by eight feet containing a number of cuboidal wooden obstacles scattered around the arena. Two infrared beacons were placed on opposite corners of the arena as shown.

conjunction with the photosensors mentioned above for learning the memory-based navigational behaviors.

The above sensory repertoire was supplemented by two effectors consisting of a DC drive motor for the two rear wheels and a Futaba servo motor at the front for steering the robot. The maximum speed of the robot was approximately 1 foot per second. The robot executed turns by setting the servo motor (front wheel) direction to angles between -40 and $+40$ degrees with respect to the center of the servo (0 degrees denoted straight ahead). The robot was made completely autonomous with an on-board power supply of rechargeable DC batteries for driving the rear motor and a separate set of AA batteries for powering both the on-board Motorola 68HC11 microcontroller and the front servo motor. The software for the microcontroller was written and compiled off-line on a Sun SparcStation in the Interactive C language and the executable code was downloaded onto the microcontroller board after compilation.

The environment of the robot, shown in Figure 6, consisted of an eight-foot square arena containing a number of scattered cuboidal obstacles. Two infrared beacons were placed at

Table 1. Learned perception-to-action mapping for collision detection and recovery. See text for details (FL - front left, FR - front right, SL - side left, SR - side right, and B - back bump sensor).

Perception	Action
FL	Backward and Left
FR	Backward and Right
SL	Forward and Right
SR	Forward and Left
B	Forward and Random Turn
FL and SL	Backward and Left
FR and SR	Backward and Right
FL and FR	Backward and Random Turn

the near and far corners. The beacons transmitted a 40 kHz square wave modulated by a 100 or 125 Hz square wave depending on the corner. These beacons were used for cyclic navigation between the two corners for demonstrating the learned lower level behaviors during beacon following (see below; Figure 9).

The hierarchical control architecture of the robot is shown in Figure 1. The lowest level behavior of collision detection occurs at the top of the diagram and can inhibit both of the other behaviors when appropriate conditions are satisfied. Similarly, the obstacle avoidance behavior can inhibit the goal-driven navigational behavior if the prerequisite conditions for this behavior are satisfied by the current perception vector.

In the first set of experiments, we tested the stochastic hill-climbing strategy (Section 3) for learning the lower level collision detection and obstacle avoidance behaviors. After initialization with a random policy of perception-action vectors, the robot first proceeds to learn the task of detecting collisions with obstacles and taking the necessary actions for freeing itself from the current obstacle. Once the algorithm achieves adequate performance as specified by a preset criterion, the control architecture automatically switches behaviors and begins learning at the next level. For example, in the case of the collision detection behavior, when the frequency with which the robot collides with obstacles drops below a given threshold of tolerance, the algorithm proceeds to the task of learning the obstacle avoidance behavior. The goal of the avoidance behavior is to prevent collisions by mapping photosensor inputs to actions that will maneuver the robot away from obstacles, thereby avoiding collisions.

Figure 7 shows the performance of the robot while learning the collision detection behavior, where the goal of the robot is to detect collisions and take the necessary actions for freeing itself from the current obstacle. This learning occurred during random exploration of the environment. As seen in the figure, the number of collisions-per-cycle drops sharply until it reaches a stable value of approximately 0.25, beyond which point the robot was not able to improve. This is understandable given that the perceptions during the learning of this behavior are restricted to whether or not one or more bump sensors are depressed, and therefore collisions can only be detected after their occurrence. Table 1 shows the perception-to-action mapping (or "policy") learned for the obstacle detection behavior.

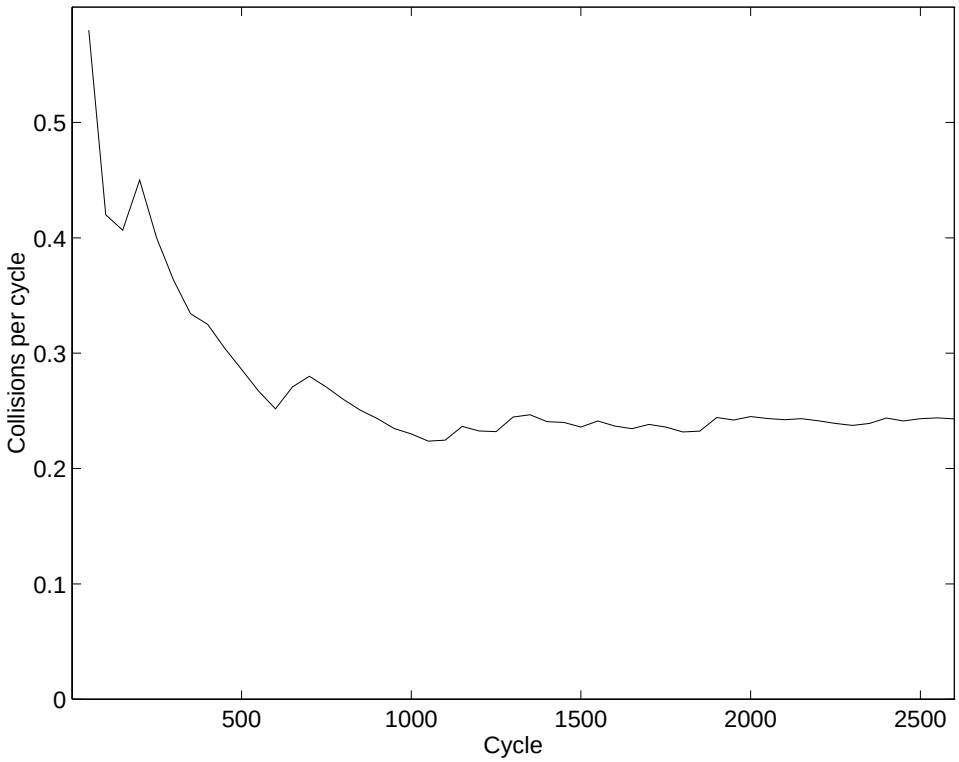

Figure 7. **Situated Learning of the Collision Detection Behavior using Stochastic Hill-Climbing.** The robot was initialized with a random policy and allowed to interact with its environment (Figure 6) to learn a policy for collision detection and recovery using stochastic hill-climbing. The plot shows the average number of collisions per cycle as a function of the number of cycles. Notice the reduction in the number of collisions over time indicating the formation of successively better policies for mapping the inputs from the bump sensors to steering commands for collision detection and recovery. The final learned policy is shown in Table 1.

The perception column indicates which of the five bump sensors is currently pressed - FL (front left), FR (front right), SL (side left), SR (side right), and B (back). The action column indicates which of six actions (indicating drive motor and steering wheel turn direction) to take for each perception. As can be seen, the robot appears to have learned the correct set of actions to take when its sensors indicate it has collided with an obstacle.

Once the robot has achieved a good level of performance in the collision detection behavior (as given by a predetermined threshold), it switches to the task of learning the next level behavior, in this case, obstacle avoidance using the photosensors. The learning curve for this behavior is shown in Figure 8. The number of collisions-per-cycle again drops sharply, starting this time with the final value from the first behavior, and eventually reaching a new minimum of about 0.03 collisions per cycle. The perception-to-action mapping that was learned is shown in Table 2. The perception column indicates which of the three front photosensors (L - left, M - middle, R - right) has detected an obstacle in front of the robot.

Table 2. Learned perception-to-action mapping for obstacle avoidance. See text for details (L - left, M - middle, R - right photosensor).

Perception	Action
L	Forward and Right
M	Forward and Right
R	Forward and Left
L and M	Forward and Right
M and R	Forward and Left
L and R	Forward
L, M and R	Forward and Random Turn

An obstacle was assumed to be detected if the analog sensor value (0-255) exceeded the threshold for the given sensor (thresholds of 180, 180, and 200 were used for the three sensors). The action column indicates which of four actions (indicating drive motor and steering wheel turn direction) to take for each perception. As can be seen from the table, the robot appears to have learned a policy that results in significantly fewer collisions than when using bump sensors alone. Note that in the case where the obstacle is detected in front by both the front left and right photosensors, the robot chooses to go forward and allow the lower-level collision detection behavior to handle the impending collision rather than going backwards. We attribute this behavior to the fact that in these cases, it may be too late to avoid the collision completely (the bump sensors become depressed before an avoidance action can be executed) and therefore it is sensible to engage the collision-detection behavior as soon as possible. Note that the finite turning radius of the robot and the rather cluttered environment of the arena together make it difficult to eliminate collisions completely.

Figure 9 depicts snapshots from a movie of the robot after it has learned both the lower level behaviors. These behaviors were used in conjunction with a higher-level beacon following behavior, where the goal was to navigate from a given location to an infrared beacon at a corner of the arena (this corresponds to the one-one navigation box in the classification of Figure 4). As the images demonstrate, the robot was able to successfully bootstrap from an initially random set of policies to learn the appropriate navigational behaviors for homing to the location of the goal beacon. The total time for learning the three behaviors was approximately 15 minutes, which is reasonable given that the learning occurred in a relatively noisy and cluttered real world environment.

A second set of experiments focused on the task of building upon the lower level reactive behaviors to achieve *memory-based navigation* using sparse distributed memory. The slow processing speed of the on-board microcontroller unfortunately limited its use in this endeavor and we therefore evaluated the feasibility of the memory-based algorithms by using a simulation (in MATLAB) of the robot and its environment (Figure 10). The simulated environment contains a single light source resting along one of the sides of the arena and an infrared beacon at one of the corners, designated as the "home" location. The floor of the simulated arena was colored in various shades of grey to provide a sufficiently rich set of visual inputs for perception-based navigation. The photosensor readings of the floor were one of the following 6 analog values depending on the color of the current region of the

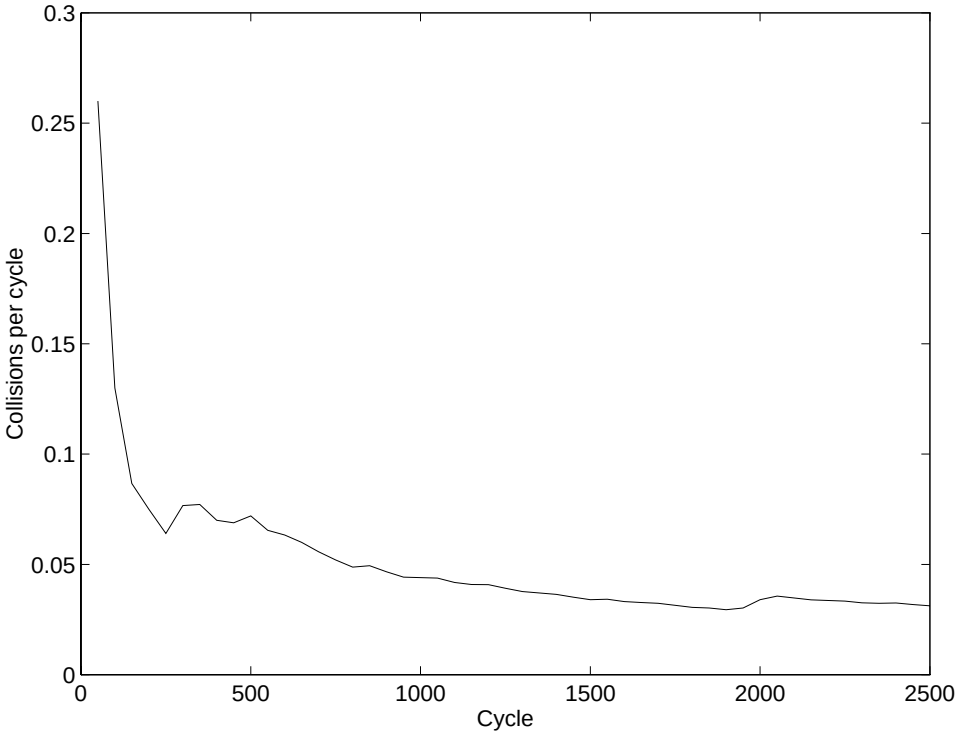

Figure 8. **Situated Learning of Obstacle Avoidance using Stochastic Hill-Climbing.** The average number of collisions per cycle is plotted as a function of the number of cycles. Once again, there is a reduction in the number of collisions over time indicating the acquisition of progressively better policies for mapping the photosensor inputs to steering commands for obstacle avoidance. The final learned policy is shown in Table 2.

floor: 0, 20, 40, 80, 100, and 120. The photosensor reading for an obstacle was assumed to be 255. Other robot perceptions pertaining to the infrared beacon on the corner and the light source on the side of the arena were computed using the simplifying assumption that light/infrared intensity at a particular orientation θ and at a distance r from the source is proportional to the solid angle subtended by the source (*i.e.*, $cos(\theta)/r^2$), assuming unit area for the robot sensors.

We first tested the simple non-predictive self-organizing SDM from Section 5 by training the simulated robot to home to a particular location from an arbitrary number of other locations in the robot arena (this corresponds to the many-one navigation box in the classification of Figure 4). The hierarchical control architecture of Figure 1 was used and it was assumed that the lower level behaviors of collision detection and obstacle avoidance had been previously learned as described above. The SDM parameters were set as follows: number of input units $n = 16$, number of hidden units $m = 300$, and number of output units $k = 1$. The 16 components of the input perception vector consisted of 4 infrared

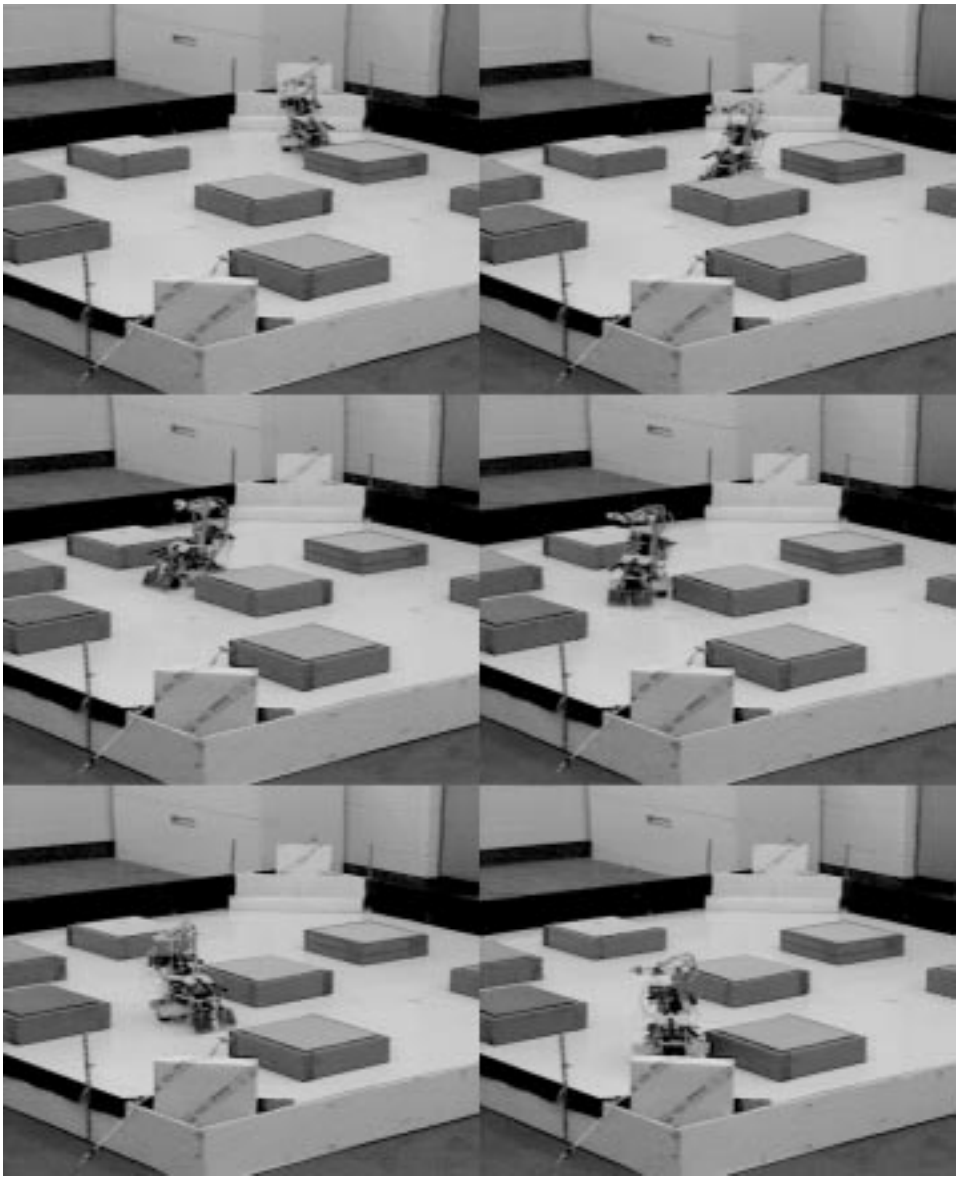

Figure 9. **An Example of Navigation after Learning**. The sequence of images (left to right, top to bottom) depict the navigational behavior of the robot after the lower level behaviors have been learned. These learned reactive behaviors were used in conjunction with a beacon following behavior (also learned via stochastic hill-climbing) to allow the robot to home to the infrared beacon in the nearest corner of the arena.

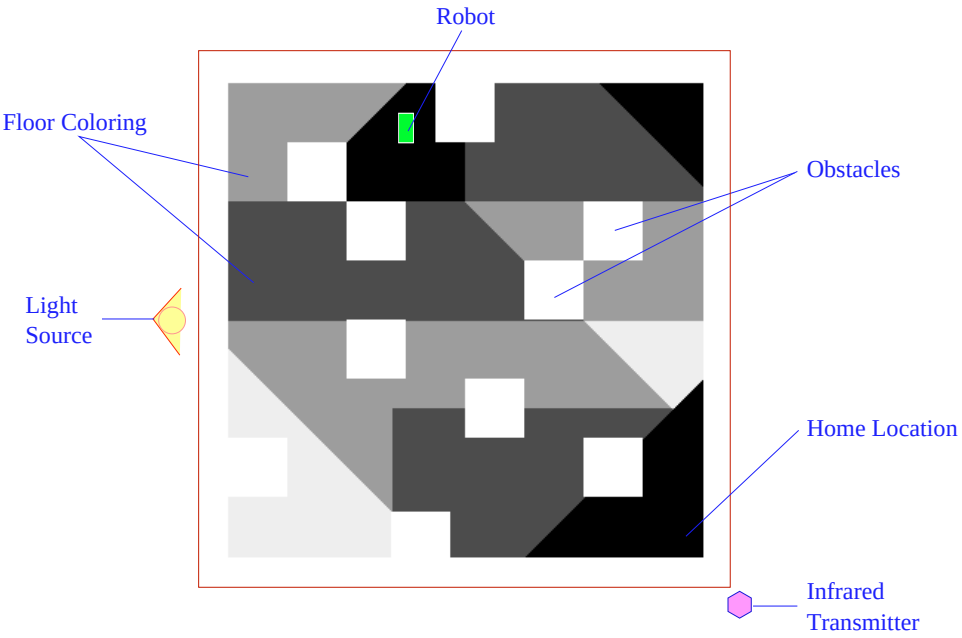

Figure 10. **Simulated Environment of the Robot**. The figure shows the model of the environment used in the simulation experiments. In addition to ambient lighting, a single light source was placed along one of the sides of the arena and an infrared beacon was assumed to be present at the home location. Floor color is depicted by shades of grey and obstacles are colored white. Perceptual input to the sparse distributed memory comprised of photosensor readings from the floor, from the light source, and infrared readings from the beacon, all taken at the current location.

inputs (located front, left, right, and back of the robot), 6 photosensor light source readings (one on the front, two on the left, two on the right, and one on the back) and 6 floor color readings from the remaining photosensors (located front left, front middle, front right, side left, side right, and back of the robot). The speed of the robot was assumed to be constant and the single output of the memory consisted of steering angle for the front wheel servo motor.

The memory was trained by "teleoperating" the simulated robot along several different paths to a single home location in the arena. Teleoperation was simulated using a point-and-click interface that allowed the operator to indicate the next position of the robot given a birds-eye view of the arena as in Figure 10. After training, the ability of the simulated robot to use its learned sensorimotor basis vectors for navigation was tested by placing the robot in different locations in the arena and allowing it to navigate autonomously. Figure 11 shows some typical examples of the robot's navigational behavior. As is evident, the robot appears to have learned an appropriate set of basis vectors during training for mapping its perceptions to the desired actions for navigating to the home position. Note that the existence of mild perceptual aliasing causes the robot to deviate on some occasions, thereby necessitating the use of the lower level collision detection and obstacle avoidance behaviors.

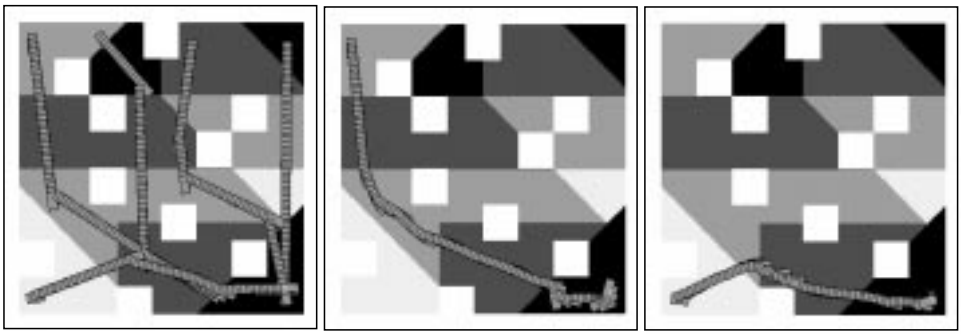

Figure 11. **Autonomous Homing using the non-predictive Sparse Distributed Memory** (a) The paths on which the memory was trained by "teleoperating" the simulated robot (shown as a grey rectangular box at successive moments in time). The home position is at the lower right corner. (b) and (c) depict the paths chosen by the robot when placed at two different positions within the arena. Note the slight deviations from the training paths caused by mild perceptual aliasing; the collision detection and obstacle avoidance behaviors are automatically invoked when the deviations cause encounters with the wall or the square obstacles.

In a subsequent set of experiments (corresponding to the many-many box in Figure 4), the robot was trained on a number of paths that *intersected* each other at different locations (Figure 12 (a)). Given a particular start location, the goal was to learn to navigate to a predetermined destination location. The current destination location was determined entirely by the starting location. What makes this task especially difficult is that at the points where the different paths intersect, local perceptions alone do not suffice to determine the next steering direction due to the many different actions that were executed during training from that perceptual state; some memory of past perceptions is required to disambiguate the current perceptual state. Under these conditions, the non-predictive memory usually fails to find the correct direction to continue in order to reach the pertinent goal destination since it relies solely on the current perception vector. On the other hand, as shown in Figure 12 (b), (c) and (d), the predictive SDM allows the robot to use the context of past perceptions to determine its current action, thereby guiding it to its appropriate destination in spite of the presence of varying degrees of perceptual aliasing. The parameters of the predictive SDM for this experiment were as follows: number of inputs $n = 16$, number of hidden units $m = 120$, number of outputs $k = 1$, and "short-term" memory size $s = 4$. The 16 components of the input perception vector were the same as in the previous experiment and the output was again the steering angle. Note that the number of hidden units is lesser than in the previous experiment, although we now have an extra unit for computing the estimated action $\hat{\mathbf{a}}_t$ and 4 additional units for computing the predicted actions $\mathbf{a}_{t+1}$, $\mathbf{a}_{t+2}$, $\mathbf{a}_{t+3}$ and $\mathbf{a}_{t+4}$ from the outputs of the hidden units (see Figure 3). Also note that the actions in the simulations were not noisy and therefore it is more likely in this case that the robot ends up at exactly one of the intersecting points in the training paths when placed at a given starting location. Testing the simulated robot under noisy sensorimotor conditions remains a subject of future research.

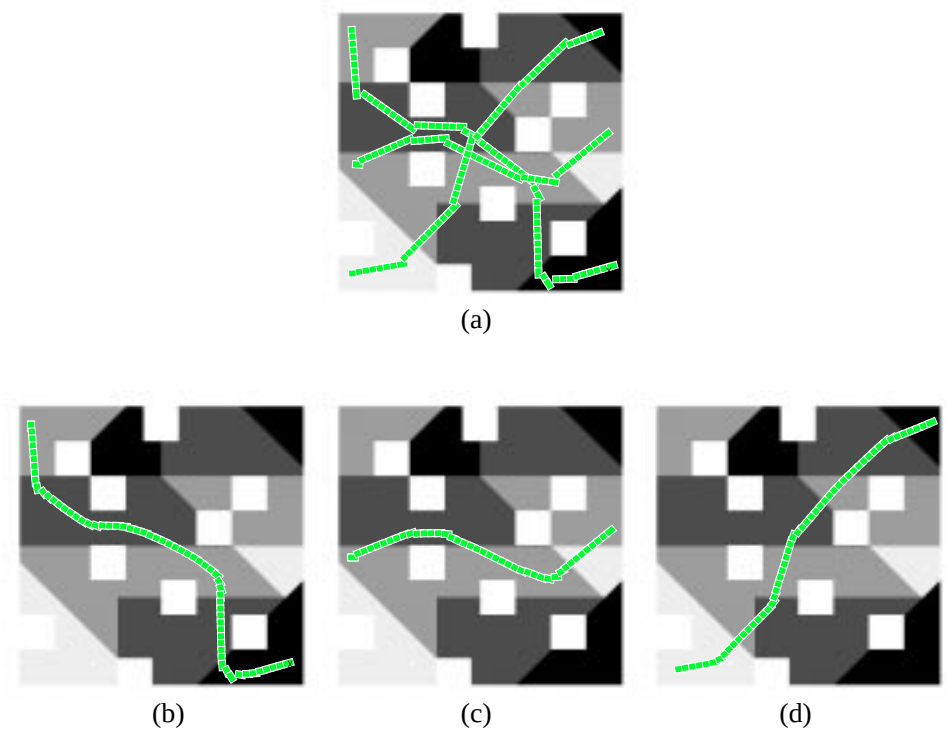

Figure 12. **Navigation using the Predictive Sparse Distributed Memory** (a) The paths on which the memory was trained by teleoperating the robot (grey rectangular box). Note the intersection of the training paths at various points which gives rise to perceptual aliasing. (b), (c) and (d) show that the predictive memory is able to circumvent aliasing effects and follow the path to its goal destination by using past sensory information as a contextual aid to disambiguating aliased perceptions.

8. Summary and Conclusions

We have shown that a hierarchical control architecture that combines low level reactive behaviors with a predictive sparse distributed memory for goal-directed navigation provides an efficient platform for learning useful navigational behaviors. The proposed approach enjoys several favorable properties:

- **Hierarchical Behavior-Based Task Decomposition and Modular Learning**: By dividing a task into several layers of control and partitioning the sensorimotor space in a hierarchical behavior-based manner, the problem of high dimensionality of the sensorimotor space is alleviated, thereby facilitating fast, independent and modular learning of behaviors.

- **Situated Learning of Reactive Behaviors**: Lower level reactive behaviors are learned on-line via direct experimentation and interaction with the environment using a simple stochastic hill-climbing algorithm. Such a method requires a minimal amount of mem-

ory space, involving the storage of only a small number of parameters for determining credit assignment, followed by a stochastic change in the perception-to-action mapping.

- **Sparse Memory**: In contrast to nearest-neighbor table look-up techniques, the memory-based method used for goal-directed navigation employs only a sparse number of memory locations and avoids the "curse of dimensionality" problem by intelligently sampling the high-dimensional sensorimotor space using competitive learning (see below).

- **Competitive Learning of Sensorimotor Basis Functions**: The contents of the sparse memory are self-organized using a form of competitive learning that can be related to maximum likelihood estimation (Nowlan, 1990). The learning rule allows the memory to autonomously form its own set of basis functions for parsimoniously describing the current sensorimotor space.

- **Distributed Storage**: Inputs to the memory are distributed across a number of locations, thereby inheriting the well-known advantages of distributed representations (Hinton et al., 1986) such as generalization to previously unknown inputs and resistance to faults in memory and internal noise.

- **Motor Prediction based on Past Perceptual History**: The problem of *perceptual aliasing* is alleviated by employing past perceptions to predict and influence current motor output. This extends the application of the method to finite-order Markovian sensorimotor environments and distinguishes our approach from previous neural network approaches based on training procedures such as back-propagation (Pomerleau, 1989).

- **Biological Similarities**: The structure of the memory bears some striking resemblances to the organization of the mammalian cerebellum (see (Kanerva, 1993, Rao & Fuentes, 1995) for more details) and therefore also to the cerebellar model of Marr (1969), Albus's CMAC (Albus, 1971), and various forms of radial basis function networks (Broomhead & Lowe, 1988, Poggio & Girosi, 1990). The possibility thus exists that biological structures and learning processes qualitatively similar to those proposed herein may underlie certain forms of goal-directed navigation and other sensorimotor activities in mammals.

The proposed method also has certain drawbacks that remain a subject of future research. For instance, although each behavior in the control architecture of the robot is learned autonomously, the present approach requires the appropriate set of behaviors for a given task to be specified a priori, along with a reasonable way to arbitrate among them. A better strategy would be to allow modularity to emerge autonomously within a control architecture (see, for example, (Nolfi, 1997)). Another problematic issue is the pre-determination of the length s of the perceptual history window (the amount of "short-term memory"). This depends on the Markovian order of the input environment, *i.e.*, the maximum number of past perceptions that the current perceptual state can depend on. In some cases, this may be experimentally estimated but in the general case, it may be desirable to adaptively increase the value of s as required (see (McCallum, 1996)). Alternately, following (Tani & Fukumura, 1994), one may use partially recurrent networks such as Jordan nets (Jordan, 1986) and related variants (Hertz et al., 1991) which do not require explicit specification of the memory parameter s. However, this approach suffers from the drawback

that past context is available only as a decaying weighted average of previous inputs which may not always suffice to disambiguate aliased environmental states.

Sparse distributed memories have previously been used for a wide variety of tasks such as object recognition (Rao & Ballard, 1995d), face recognition (Rao & Ballard, 1995c), speech recognition (Prager & Fallside, 1989), speech synthesis (Joglekar, 1989), and weather prediction (Rogers, 1990). The present work shows that in conjunction with learned lower level reactive behaviors such as collision detection and obstacle avoidance, such memories can also be used for learning useful navigational behaviors in autonomous mobile robots. A possible concern is whether our modifications to the standard SDM of Kanerva adversely affects its favorable properties. Fortunately, this is not the case for most of the modifications proposed. The favorable properties of high-dimensional vectors apply to both binary (as in the standard SDM) and real-valued vectors (see (Rao & Ballard, 1995a)). The self-organization of the address space to tailor it to the current input environment overcomes the problems of over- and under-utilization of memory caused by picking uniformly random fixed input addresses as in the standard SDM model. This helps to increase the capacity of the memory (Kanerva, 1993). Also, as already mentioned above, the specific competitive learning procedure in our approach is related to the statistically sound framework of maximum likelihood estimation.

Our ongoing efforts include extending the results obtained in the simulations to more complex real world environments, investigating the dynamics of credit assignment in the stochastic hill-climbing algorithm, and exploring the effects of on-line adaptation of some of the free parameters in the approach such as the length of the perceptual history window and the weights γ_i for combining predicted actions. A simultaneous effort involves the exploration of predictive recurrent networks (such as adaptive and hierarchical Kalman filter networks (Rao & Ballard, 1997)) for learning useful behaviors in autonomous robots.

Acknowledgments

We would like to thank the three anonymous reviewers for their comments and suggestions which contributed significantly to improving the clarity and quality of this paper. We would also like to thank Dana Ballard and Randal Nelson for several interesting discussions, Chris Brown for his encouragement and support for building the mobile robot, and Mike Van Wie for his help in the construction of the robot and for his involvement in some of the robot experiments. This work was supported by NSF research grant CDA-8822724, ARPA grant MDA972-92-J-1012 and ONR grant N00014-93-1-0221.

References

Albus, J. (1971). A theory of cerebellar functions. *Math. Bio.*, 10:25–61.

Asada, M., Uchibe, E., Noda, S., Tawaratsumida, S., & Hosoda, K. (1994). Coordination of multiple behaviors acquired by a vision-based reinforcement learning. In *Proceedings of the 1994 IEEE/RSJ International Conference on Intelligent Robots an Systems*, pages 917–924, Munich, Germany.

Beer, R. & Gallagher, J. (1992). Evolving dynamical neural networks for adaptive behavior. *Adaptive Behavior*, 1:91–122.

Brooks, R. A. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–22.

Broomhead, D. S. & Lowe, D. (1988). Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2:321–355.

Chrisman, L. (1992). Reinforcement learning with perceptual aliasing. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*.

Cliff, D., Husbands, P., & Harvey, I. (1992). Evolving visually guided robots. In Meyer, J. A., Roitblat, H., and Wilson, S., editors, *From Animals to Animats 2: Proceedings of the Second International Conference on the Simulation of Adaptive Behavior*, pages 374–383. Cambridge, MA: MIT Press.

Connell, J. H. (1990). *Minimalist mobile robotics : A colony-style architecture for an artificial creature*. Academic Press, Boston, MA.

Dayan, P. & Hinton, G. (1993). Feudal reinforcement learning. In Hanson, S., Cowan, J., and Giles, C., editors, *Advances in Neural Information Processing Systems 5*, pages 271–278. San Mateo, CA: Morgan Kaufmann.

de Bourcier, P. (1993). Animate navigation using visual landmarks. Cognitive Science Research Papers 277, University of Sussex at Brighton.

Elfes, A. (1987). Sonar-based real-world mapping and navigation. *IEEE Journal of Robotics and Automation*, 3:249–265.

Fuentes, O. & Nelson, R. C. (1997). Learning dextrous manipulation skills using the evolution strategy. In *Proceedings of the 1997 IEEE International Conference on Robotics and Automation*, Albuquerque, New Mexico.

Fuentes, O., Rao, R. P. N., & Wie, M. V. (1995). Hierarchical learning of reactive behaviors in an autonomous mobile robot. In *Proceedings of the IEEE International Conf. on Systems, Man, and Cybernetics*.

Hackbusch, W. (1985). *Multi-grid methods and applications*. Berlin: Springer-Verlag.

Hertz, J., Krogh, A., & Palmer, R. (1991). *Introduction to the Theory of Neural Computation*. Addison-Wesley Publishing Company, Inc., Redwood City, CA.

Hinton, G., McClelland, J., & Rumelhart, D. (1986). Distributed representations. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1. MIT Press, Cambridge, MA.

Joglekar, U. D. (1989). Learning to read aloud: A neural network approach using sparse distributed memory. Technical Report 89.27, Research Institute for Advanced Computer Science, NASA Ames Research Center.

Jordan, M. (1986). Attractor dynamics and parallelism in a connectionist sequential machine. In *Proceedings of the 1986 Cognitive Science Conference*.

Kaelbling, L. (1993a). Hierarchical reinforcement learning: Preliminary results. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 167–173. San Mateo, CA: Morgan Kaufmann.

Kaelbling, L. P. (1993b). *Learning in embedded systems*. MIT Press, Cambridge, MA.

Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285.

Kanerva, P. (1988). *Sparse Distributed Memory*. Cambridge, MA: Bradford Books.

Kanerva, P. (1993). Sparse distributed memory and related models. In Hassoun, M. H., editor, *Associative Neural Memories*, pages 50–76. New York: Oxford University Press.

Krose, B. J. A. & Eecen, M. (1994). A self-organizing representation of sensor space for mobile robot navigation. In *IEEE/RSJ/GI International Conference on Intelligent Robots an Systems*, pages 9–14.

Kuipers, B. J. & Byun, Y.-T. (1988). A robust, qualitiative approach to a spatial learning mobile robot. In *SPIE Cambridge Symposium on Optical and Optoelectronic Engineering: Advances in Intelligent Robotics Systems*.

Lewis, M. A., Fagg, A. H., & Solidum, A. (1992). Genetic programming approach to the construction of a neural network for control of a walking robot. In *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, Nice, France.

Maes, P. & Brooks, R. A. (1990). Learning to coordinate behaviors. In *Proceedings of AAAI-90*, pages 796–802.

Mahadevan, S. & Connell, J. (1991). Scaling reinforcement learning to robotics by exploiting the subsumption architecture. In *Proceedings of the Eighth International Workshop on Machine Learning*.

Marr, D. (1969). A theory of cerebellar cortex. *J. Physiol. (London)*, 202:437–470.

Mataric, M. (1992). Integration of representation into goal-driven behavior-based robot. *IEEE Transactions on Robotics and Automation*, 8(3):304–312.

McCallum, R. A. (1996). Hidden state and reinforcement learning with instance-based state identification. *IEEE Trans. on Systems, Man and Cybernetics*, 26(3):464–473.

McIlwain, J. T. (1991). Distributed spatial coding in the superior colliculus: A review. *Visual Neuroscience*, 6:3–13.

Nehmzow, U. & Smithers, T. (1991). Mapbuilding using self-organizing networks in "Really Useful Robots". In Meyer, J.-A. and Wilson, S. W., editors, *From Animals to Animats 1: Proceedings of the First International Conference on Simulation of Adaptive Behavior*, pages 152–159. Cambridge, MA: MIT Press.

Nelson, R. C. (1991). Visual homing using an associative memory. *Biological Cybernetics*, 65:281–291.

Nolfi, S. (1997). Using emergent modularity to develop control systems for mobile robots. *Adaptive Behavior*, 5:343–363.

Nowlan, S. (1990). Maximum likelihood competitive learning. In Touretzky, D., editor, *Advances in Neural Information Processing Systems 2*, pages 574–582. San Mateo, CA: Morgan Kaufmann.

Pierce, D. & Kuipers, B. (1991). Learning hill-climbing functions as a strategy for generating behaviors in a mobile robot. In *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior*, pages 327–336.

Poggio, T. & Girosi, F. (1990). Networks for approximation and learning. *Proc. IEEE*, 78:1481–1497.

Pomerleau, D. (1989). ALVINN: An autonomous land vehicle in a neural network. In Touretzky, D., editor, *Advances in Neural Information Processing Systems*, volume 1, pages 305–313. Morgan Kaufmann, San Mateo.

Pomerleau, D. (1991). Efficient training of artificial neural networks for autonomous navigation. *Neural Computation*, 3(1):88–97.

Prager, R. W. & Fallside, F. (1989). The modified Kanerva model for automatic speech recognition. *Computer Speech and Language*, 3(1):61–81.

Rao, R. P. N. & Ballard, D. H. (1995a). An active vision architecture based on iconic representations. *Artificial Intelligence (Special Issue on Vision)*, 78:461–505.

Rao, R. P. N. & Ballard, D. H. (1995b). Learning saccadic eye movements using multiscale spatial filters. In Tesauro, G., Touretzky, D. S., and Leen, T. K., editors, *Advances in Neural Information Processing Systems 7*, pages 893–900. Cambridge, MA: MIT Press.

Rao, R. P. N. & Ballard, D. H. (1995c). Natural basis functions and topographic memory for face recognition. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 10–17.

Rao, R. P. N. & Ballard, D. H. (1995d). Object indexing using an iconic sparse distributed memory. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 24–31.

Rao, R. P. N. & Ballard, D. H. (1997). Dynamic model of visual recognition predicts neural response properties in the visual cortex. *Neural Computation*, 9(4):721–763.

Rao, R. P. N. & Fuentes, O. (1995). Perceptual homing by an autonomous mobile robot using sparse self-organizing sensory-motor maps. In *Proceedings of World Congress on Neural Networks (WCNN)*, pages II380–II383.

Rao, R. P. N. & Fuentes, O. (1996). Learning navigational behaviors using a predictive sparse distributed memory. In *From Animals to Animats 4: Proceedings of the Fourth Int. Conf. on Simulation of Adaptive Behavior (SAB)*, pages 382–390.

Rogers, D. (1990). Predicting weather using a genetic memory: A combination of Kanerva's sparse distributed memory and Holland's genetic algorithms. In Touretzky, D. S., editor, *Advances in Neural Information Processing Systems 2*, pages 455–464. Morgan Kaufmann.

Singh, S. (1992). Transfer of learning by composing solutions of elemental sequential tasks. *Machine Learning*, 8:323–339.

Tani, J. & Fukumura, N. (1994). Learning goal-directed sensory-based navigation of a mobile robot. *Neural Networks*, 7(3):553–563.

Whitehead, S. & Ballard, D. (1991). Learning to perceive and act by trial and error. *Machine Learning*, 7(1):45–83.

Wixson, L. (1991). Scaling reinforcement learning techniques via modularity. In *Proceedings of the Eighth International Workshop on Machine Learning*, pages 368–372. Morgan Kaufmann.

Yair, E., Zeger, K., & Gersho, A. (1992). Competitive learning and soft competition for vector quantizer design. *IEEE Trans. Signal Processing*, 40(2):294–309.

Received September 1, 1997
Accepted December 30, 1997
Final Manuscript February 1, 1998