From Neural Networks to the Brain: Autonomous Mental Development

Juyang Weng and Wey-Shiuan Hwang
Department of Computer Science and Engineering
Michigan State University
East Lansing, MI 48824
Email:weng@cse.msu.edu

Abstract

Artificial neural networks can model cortical local learning and signal processing, but they are not the brain, neither are many special purpose systems to which they contribute. Autonomous mental development models all or part of the brain (or the central nervous system) and how it develops and learns autonomously from infancy to adulthood. Like neural network research, such modeling aims to be biologically plausible. This paper discusses why autonomous development is necessary according to a concept called task muddiness. Then it introduces recent results for a series of research issues, including the new paradigm for autonomous development, mental architectures, developmental algorithm, a refined classification of types of machine learning, spatial complexity and time complexity. Finally, the paper presents some experimental results for applications, including: vision-guided navigation, object finding, object-based attention (eye-pan), and attention-guided pre-reaching, four tasks which infants learn to perform early but very perceptually challenging for robots.

Key words: cognitive development, autonomous learning, mental architecture, on-line learning, incremental learning, visual learning, working memory, long-term memory, self-organization, regression, autonomous navigation, attention selection, object recognition,

Corresponding author:
Juyang Weng
3115 Engineering Building
Embodied Intelligence Laboratory
Department of Computer Science and Engineering
Michigan State University
East Lansing, MI 48824-1027
Email: weng@cse.msu.edu
Phone: (517) 353-4388
Fax: (517) 353-4388

I. Introduction

A human being starts to develop from the time of conception. At that time, a single cell called a zygote is formed. In biology, the term *genotype* refers to all or part of the genetic constitution of an organism. The term *phenotype* refers to all or part of the visible properties of an organism that are produced through the interaction between the genotype and the environment. In the zygote, all the genetic constitution is called genome. At the conception of a new human life, a biological program starts to run. Let us call it the *developmental program*. The goal of an artificial developmental program is to simulate all or part of the functions of biological development.

CALL OUT: *The biological developmental program resides in the genome of the zygot.*

The biological developmental program handles two types of development, body development and mental development. The former is the development of everything in the body excluding the brain. The latter is the development of the brain (or the central nervous system CNS). Through the body development, a normal child grows in size and weight, along with many other physical changes. Through the mental development, a normal child develops a series of mental capabilities through interactions with the environment. Mental capabilities refer to all known brain capabilities, which include, but not limited to, perceptual, cognitive, behavioral and motivational capabilities. In this paper, the term *development* refers to mental development unless stated otherwise. The biological mental development takes place in concurrence with the body development and they are closely related. For example, if the eyes are not normally developed, the development of the visual capabilities is greatly affected. In the development of an artificial agent, the body can be designed and fixed, which helps to reduce the complexity of the autonomous mental development.

CALL OUT: *For the development of an artificial agent, the body can be designed and fixed, which helps to reduce the complexity of the autonomous mental development.*

Much progress has been made in neuroscience and developmental psychology. However, providing a detailed computational model of human mental development is still a holly grail. Computational models of mental development in the computational intelligence community can not only benefit from neuroscience and psychology, but also contribute to modeling and understanding how the brain works.

Past research in robot and machine learning has been informed by models of human development and learning to various degrees. BAIRN is a symbolic self-modifying information processing system in simulation for modeling a theory of cognitive development [36]. Also in simulation, Drescher [11] utilized the schema, a symbolic tripartite structure in the form of "context-action-result," to model a form of child sensory-motor learning in Piaget's constructivist theory of cognitive development. The explanation-based neural network

by Thrun [31] was motivated by "life-long learning." Some well-known animal learning models were implemented on robots, such as classical conditioning by Darwin IV [35] and instrumental conditioning by Skinnerbots [32]. The research effort at Michigan State University aims at engineering capable online developmental robots. It has been evolved four systems: Cresceptron (1991 - 1997) [45], SHOSLIF (1993 - 2000) [46], SAIL (1996 - present) [39] [48] [49] and Dav (1999 - present) [16]. The SAIL robot [39] and the Darwin V robot [1] are two robot prototypes that were developed independently around the same time for simulating cognitive development but with very different goals. Darwin V was designed to provide a concrete example of how the computational weights of neural circuits are determined by the behavioral and environmental interactions of an autonomous device. The goal of SAIL was to autonomously generate representations for scaling up to more complex capabilities in unconstrained, unknown human environments for potential engineering applications.

Recently, there has been a significant growth of research on computational modeling of mental development. As reviewed by the article in the Historic Perspective Column of this issue, the Workshop on Development and Learning (WDL) 2000 was the first multi-disciplinary gathering on computational modeling of mental development — both natural *and* artificial [50] [51]. Currently, there are at least two regularly scheduled gatherings on computational development, the International Conferences on Development and Learning and the International Workshops on Epigenetic Robotics, where much computational modeling work was been published in recent years. In the large volume of published computational modeling work of development, much work intends to study some related characteristics or component of development, which is of course useful for understanding development. However, the usefulness of component techniques has not been demonstrated fully until the techniques have been tested in a end-to-end developmental system capable of open-ended, task-nonspecific, autonomous mental development. A major benefit of developmental learning, compared to non-developmental machine learning, is the task non-specificity [50] in learning many simple to complex skills needed for carrying out any complex task. This paper emphasizes such developmental systems.

Section II discusses the necessity of autonomous mental development, the paradigm of biologically inspired autonomous mental development and some associated basic concepts. In Section III, we describe a series of mental architectures and an example. Section IV presents the computation requirements for autonomous development and some regression methods motivated by neural networks. Section V presents some experimental results for applications. To address the scalability, the space and time complexity is discussed in Section VI. Section VII provides some concluding remarks.

II. Autonomous Online Development

Why do we need autonomous mental development? Why is the traditional machine learning very limited in dealing with tasks that we think require intelligence? To understand these issues better, let us consider the concept of task muddiness.

A. Task muddiness

Despite the power of modern computers, whose basic principle known as the Turing Machine was first introduced in 1936 by Alan M. Turing [33], we have seen a paradoxical picture of artificial intelligence (AI): Computers have done very well in areas that are typically considered very difficult (by humans), such as playing chess games. But they have done poorly in areas that are commonly considered easy (by humans), such as visual and language understanding. For example, it is extremely difficult for a computer to understand a movie or, for that matter, to recognize the objects in a movie. What kind of tasks that requires intelligence?

The term "muddy" is often used in psychology to informally describe concepts that cannot be clearly defined or specified. Let us use it as a general term to measure how much intelligence is required to execute a task [43]. The composite muddiness of a task is a multiplicative product of many individual muddiness measures. There are many possible individual muddiness measures. Those individual muddiness measures are not necessarily mutually independent or at the same level of abstraction, since such a requirement is not practical nor necessary for describing the muddiness of a task. They fall into five categories: (1) external environment, (2) input, (3) internal environment, (4) output and (5) goal, as shown in Table I. The term "external" means external with respect to the brain and "internal" means internal to the brain.

The composite muddiness of a task can be considered as a product of all individual muddiness measures. In other words, a task is extremely muddy when all the five categories have a high measure. A chess playing task with symbolic input and output is a clean problem because it is low in categories (1) through (5). A symbolic language translation problem is low in (1), (2) and (4), moderate in (3) but high in (5). A vision-guided navigation task for natural human environment is high in (1), (2), (3) and (5), but moderate in (4). A human adult handles extremely muddy tasks that are high in all the five categories.

Howard Gardner [14] proposed the concept of "multiple intelligences," in the sense that human intelligence is displayed not only in logical-mathematical reasoning or emotional aspects, but also through other aspects such as bodily-kinesthetic and spatial skills. From the muddiness table Table I we have a more detailed appreciation what a human adult deals with even in a daily task, e.g., navigating or driving in a city environment. The composite muddiness of many tasks that a human or a machine can execute is proposed as a metric for measuring required intelligence.

TABLE I

A LIST OF MUDDINESS FACTORS FOR A TASK

Category	Factor	Clean ⟷	Muddy
External Env.	Awareness	Known	Unknown
	Complexity	Simple	Complex
	Controlledness	Controlled	Uncontrolled
	Variation	Fixed	Changing
	Foreseeability	Foreseeable	Nonforeseeable
Input	Rawness	Symbolic	Real sensor
	Size	Small	Large
	Background	None	Complex
	Variation	Simple	Complex
	Occlusion	None	Severe
	Activeness	Passive	Active
	Modality	Simple	Complex
	Multi-modality	Single	Multiple
Internal Env.	Size	Small	Large
	Representation	Given	Not given
	Observability	Observable	Unobservable
	Imposability	Imposable	Nonimposable
	Time coverage	Simple	Complex
Output	Terminalness	Low	High
	Size	Small	Large
	Modality	Simple	Complex
	Multimodality	Single	Multiple
Goal	Richness	Low	High
	Variability	Fixed	Variable
	Availability	Given	Unknown
	Telling-mode	Text	Multimodal
	Conveying-mode	Simple	Complex

CALL OUT: *The composite muddiness of many tasks that a human or a machine can execute is proposed as a metric for measuring required intelligence.*

It is clear that a human infant is not able to perform those muddy tasks that a human adult performs everyday. The process of mental development is necessary to develop such a wide array of mental skills. Much evidence in developmental psychology has demonstrated that not only a process of development is necessary for human intelligence, the environment of the development is also critical for normal development.
CALL OUT: *Much evidence in developmental psychology has demonstrated that not only a process of development is necessary for human intelligence, the environment of the development is also critical for normal development.*

B. Manual development

The traditional machine learning is as follows. Given a task T that the machine needs to perform, along with ecological conditions E_c under which the machine will run, a human being H constructs the machine agent A based on the ecological conditions E_c and the task T. After system fabrication, programming, machine learning and task testing, the machine is a function $A(0) = H(E_c, T)$ at the time of machine completion $t = 0$, where H denotes the human constructor as the very complex function that constructs the agent A. During the machine execution phase, $A(t)$ is run for $t > 0$ by interacting with the environment through its sensors and effectors, during which $A(t)$ may adapt and change with time t. Since $A(0)$ is completed after the task T is given, it is the human being who understands the task, chooses a representation, and maps the task to the representation.

For example, if the task is navigation in a city environment, the programmer decides that the concept of location is necessary and, thus, he assign a set of nodes, each representing a different location in the driving environment. Although the programmer does not necessarily rigidly specify which node representing which location, he decides the meaning of each node and he program rules for how to use these nodes (how to feed data into these nodes and how to use the value from the nodes) based on the programmed-in concept "location." Although such a representation framework can be general (e.g., Markov decision process or neural networks), the mapping between the task and an actual instance of the model is performed by a human and, thus, it requires a human to understand the task, not the machine. This process of task-specific human understanding and task-specific programming has proven to be a great hurdle to overcome.

This traditional manual development paradigm has created impressive man-made devices, from calculators to space shuttles. However, it has not been doing very well for constructing machines to perform muddy tasks. The main reasons include (1) the human designer cannot predict everything up-front (e.g., perception), (2) a hand-designed representation cannot account for major required information in changing environments (e.g.,

ceiling lights vs. wall posters).

In contrast, a human child can learn to perform new tasks that his ancestors have never learned, such as performing new gymnastic somersaults, reading about a new subject, or learning a new language. Thus, it is unlikely that the *task-specific* architecture and representation of these tasks are "hard wired" into the human genome. Many highly muddy tasks, such as driving, require so many simple to complex mental skills that task-specific programming is not very effective. The autonomous development paradigm is motivated by the above observation. Further, it also serves to computationally model human mental development.

C. Autonomous development

The paradigm of *autonomous development* for humans and machines is as follows. The tasks that the robot will learn to execute are unknown to the programmer during the programming time. Only given rough ecological conditions E_c (e.g., city environment or under sea), a human being H constructs a machine agent $A(t)$ based on the ecological conditions E_c: $A(0) = H(E_c)$, at the time of birth $t = 0$. After the birth, the machine agent $A(t)$ starts to interact with the environment (including its own body and humans) online in real-time through its sensors and effectors. Many muddy tasks T_1, T_2, ... T_n, ... that the robot ends up learning to perform are determined by human teachers, depending on the need at time $t > 0$ and on the performance of the agent $A(t)$ at that time.

CALL OUT: *When writing a developmental program for a developmental machine, the tasks that the machine will learn to execute are unknown to the programmer during the programming time.*

Biologically, the human zygote's genome $A(0)$ is the result of evolution, instead of being designed by any human programmer. As explained earlier, the human genome deals with not only mental development, but also body development. Therefore, the developmental approach can be employed by genetic algorithms. In fact, biological development is necessary within each generation to carry out evolution across generations.

We still use the navigation task as an example here. Since the navigation task is unknown, the concept "location" does not exist during the time of machine programming and, thus, no data structure is dedicated to this type of concept. A developmental program for autonomous development must generate vector clusters from sensory inputs as a (distributed) representation, but none of the clusters exclusively represent the concept of "location." It is the desired cognitive and behavioral capabilities that we are aiming at.

Biologically, this means that no hard-wired neurons are specified completely by the genome to handle a specific environment-dependent concept such as location. Wang & Merzenich [37] demonstrated that the skin areas from which a neuron in the somatosensory cortex of the adult monkey receives signals change significantly in weeks of training. Cells that have reached a cortical area have a genetically pre-positioned sensory modality (e.g., vision or touch), but what meaning it represents (e.g., which finger in the hand)

depends on experience. This is true not only in the young monkey, but also in the adult monkey.

In the above paradigm, the brain interacts with the environment through only the agent's sensors and effectors. In other words, the brain itself is closed during the development. In other words, the autonomous mental development should not require the human programmer to examine the internal representation and to directly alter the internal representation. Why should the brain be closed from the birth time? If the mother of a child must "see" and "understand" the internal representation of her child's brain as a necessary condition to teach her child successfully, the teaching duty is too hard for the mother. For robots, the purpose of the closed brain is to eliminate direct access to internal representation by humans after the task is known.

Some artificial cognitive frameworks (e.g., Soar [24] [25] and ACT-R [2]), and some statistical models (e.g., the Partially Observable Markov Decision Process (POMDP) [22]) are widely used as general frameworks for an agent. However, they are *computational frameworks*, not *developmental frameworks* because such a framework itself does not provide a mechanism about how an actual instance of representation can be autonomously generated directly from experience of sensors and effectors *without* human program-level intervention. For example, Soar, ACT-R and POMDP all require a human programmer to manually provide a concrete instance of representation from the supplied framework by (a) hand providing a *representation-to-task mapping* — the mapping between elements in the representation and the concepts of the given task (e.g., what task-specific concept a node represents) and (b) hand setting constraints to the parameters of the model (e.g., setting as many transition probabilities in POMDP permanently to zero according to the impossible transitions in the navigation environment, and make all the corners in the known environment share the same sub-network). Thus, the robot is still unable to generate representation autonomously from interactions with the environment. In contrast, a human child does not need his mother to establish *representation-to-task mapping*, since the mother does not know the internal brain representation of the child.

D. Innateness

The next issue is what should be programmed into such a program, or namely, what is "innate" for robots?

It is beyond the scope of this paper to fully discuss biological evidence about what might be innate. There are some inborn reflexes that are exhibited during infancy [9] [13]. For example, when the cheek is touched, the infant turns the head toward the touched side. This is known as the rooting reflex. These inborn behaviors do not need to be learned and are present early in life. These inborn reflexes are crucial for early survival but they can be overridden by later learned behaviors. However, they are not task-specific skills. For example, if a finger touches the infant's cheek instead of a nipple, his head still turns. Therefore, the rooting reflex facilitates nipple finding, not a task-specific skill only for the nipple-finding task.

Anything that is biologically innate (in the human genome) should be reasonable to be programmed into

such a developmental program. Therefore, it is not true that the robot developmental program starts from *tabula rasa*. The entire developmental program is "innate," along with some innate reflexes. The developmental program is not "bias-free" either.

CALL OUT: *Anything that is biologically innate (in the human genome) should be reasonable to be programmed into such a developmental program. The entire developmental program is "innate," along with some innate reflexes.*

A biological brain is both a *signal processor* and a *processor developer* (autonomous online learner) at the same time. In an autonomously developing robot, the *processor* and the *developer* are conceptually different but are accomplished by the same developmental program.

III. Mental architectures

To avoid a common misconception, it is necessary to discuss how the brain represents sensory world before we discuss mental architectures.

A. Representation

A central characteristic of open-ended development is to avoid a human hand-programmed world- or object-representation (e.g., a monolithic 3D map) for better environmental adaptation in learning. This characteristic is familiar to the neural network community. In the robotics community, this characteristic is supported by what is now called the behavior-based approach [6] [3]; while in the computer vision community it is consistent with what is called the appearance-based approach [21][34]. When the appearance-based approach for computer vision is used for mobile robots, these two approaches are combined [46] [8].

Biologically, the brain runs by sending signals of discrete spikes. To simulate by digital computers, we model brain samples in discrete time.

The brain has a finite memory. Therefore, it is conceptually useful to model the brain as an adaptive finite state machine. By adaptive, we mean that the number of states of the brain and the states are all adaptive. We will use the term "observation driven" to capture the nature of adaptation.

B. Markov decision process

To facilitate understanding, we start with the Markov decision process (MDP), which is a framework originated from mathematics and statistics and has been studied and used widely in the machine learning community [28], [20], [30]. The Markov decision process (MDP) has been used to model a part of the world that the agent is interested in.

Suppose $S = \{1, 2, ..., n\}$ is a set of n predefined symbolic states that is used to model a part of the world.

The state s_t at time t is a random variable taking one of the values in S. Its prior probability distribution is $P(s_0)$. The action a_t is the action of the agent at time t. Let H_t be the random history from time $t = 0$ up to time $t - 1$:

$$H_t = (s_{t-1}, s_{t-2}, ..., s_0, a_{t-1}, a_{t-2}, ..., a_0).$$

The process is called the k-th order MDP [28], [20], [30] if its conditional state transitional probability $P(s_t \mid H_t)$ satisfies

$$P(s_t \mid H_t) = P(s_t \mid l_t)$$

where l_t is the short last k frames of the history

$$l_t = (s_{t-1}, s_{t-2}, ..., s_{t-k}, a_{t-1}, a_{t-2}, ..., a_{t-k}).$$

MDP assumes that the state is directly observable (given). In many applications, the state of the world is not directly observable by the agent, or is partially observable (with noise). A partially observable MDP or POMDP [20], [30] (or HMM [29]) is one whose state s_t is not totally observable to the agent. Instead, there is an observation x_t at time t that depends on the state s_t by an observation probability $P(x_t \mid s_t)$.

MDP and POMDP have been widely used to model the part of the world that is of interest. Since it is a human designed model of the world, it needs a human programmer to establish the correspondence between the task that the robot learns about and the hand-crafted internal representation, e.g., the meaning of each node of the MDP and the meaning related to the task (e.g., a corner that the node represents). Therefore, MDP and POMDP are not suited for modeling the mental process in the central nervous system (CNS) and especially not suited for modeling developmental architecture. Therefore, we consider the following *observation driven* Markov Decision Process.

C. Type-1: Observation driven Markov decision process

Let $\mathbf{x}_t \in \mathcal{X}$ and $\mathbf{p}_t \in \mathcal{P}$ be the random observations (context) and predicted (primed) action at time t, respectively. Let H_t be the random vector of the entire history:

$$H_t = (\mathbf{x}_t, \mathbf{x}_{t-1}, ..., \mathbf{x}_0, \mathbf{p}_{t-1}, ..., \mathbf{p}_0).$$

We call the process the k-th order Observation-driven MDP (ODMDP) [10], if its transitional probability $P(\mathbf{p}_t \mid H_t)$ satisfies

$$P(\mathbf{p}_t \mid H_t) = P(\mathbf{p}_t \mid \mathbf{l}_t) \tag{1}$$

where $\mathbf{l}_k$ is the last k observations:

$$\mathbf{l}_t = (\mathbf{x}_t, \mathbf{x}_{t-1}, ..., \mathbf{x}_{t-k}, \mathbf{p}_{t-1}, ..., \mathbf{p}_{t-k}) \tag{2}$$

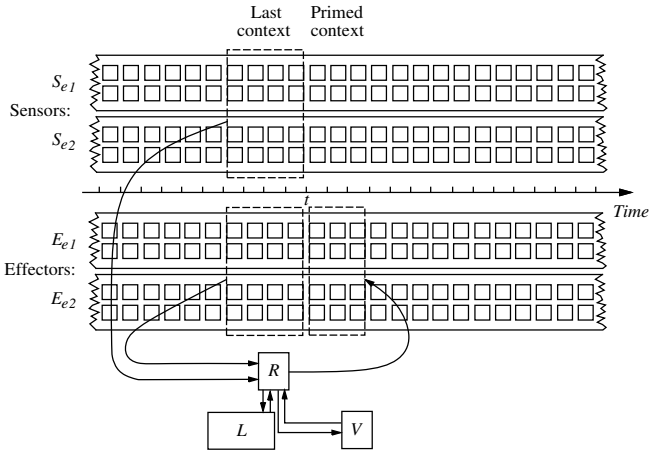

Fig. 1. Observation-driven Markov decision process: The architecture of a multi-sensor multi-effector online learning agent. In the temporal sensory and effector streams, each square denotes a receptor (e.g., pixel) or motor and vertically aligned squares form a time frame. The block marked with L is a set of context states (prototypes), which are clusters of all observed context vectors $\mathbf{l}_t$. V is the motivational (value) system.

as shown in Fig. 1.

This architecture is called Type-1 architecture. Note that numeric vectors in a high dimensional space are used as observation driven representation, instead of the hand-defined symbolic representation in MPD in the previous subsection.

The last k observations $\mathbf{l}_t$ are called the last context. The online processing problem using the architecture in Fig. 1 is then to estimate and realize the prediction $\mathbf{p}_t$ from the last context $\mathbf{l}_t$, where $\mathbf{p}_t$ is the desired action (vector) output at time t.

In practice, internally, multiple predictions for p_t are necessary to evaluate the value of each prediction. This is because multiple actions exist given the same context (e.g., left and right turns at a Y junction). Different actions lead to contexts of different values.

To denote multiple predictions, we add time variable explicitly to $\mathbf{l}_t$ and $\mathbf{p}_t$, to become $\mathbf{l}(t)$ and $\mathbf{p}(t)$, respectively. Thus, each context $\mathbf{l}(t)$ may correspond to multiple action possibilities $\mathbf{p}_1(t), ..., \mathbf{p}_k(t)$. This is accomplished by a mapping, simulating a major role of the association cortex:

$$\{\mathbf{p}_1(t), ..., \mathbf{p}_k(t)\} = R(\mathbf{l}(t)), \tag{3}$$

where $k \geq 0$ varies according to experience. Note that this k does not need to be the same as the context length k for the last context $\mathbf{l}(t)$. Thus, the mapping R is a mapping from the space of the last context $\mathcal{L}$ to the power set of the predicted outcome space $\mathcal{P}$, namely, $R : \mathcal{L} \mapsto 2^{\mathcal{P}}$.

R is developed incrementally through experience. For any $t > 0$ (after the machine "birth"), it is a total function since it is defined for all elements in $\mathcal{L}$, but it does not do well for most elements in $\mathcal{L}$ that are far

from those experienced. It is not an onto function since its range covers only to a very small part of $2^{\mathcal{P}}$.

Therefore, we need a value system that selects desirable contexts from multiple primed ones. The value system $V(t)$ takes a set of (e.g., k) contexts from the mapping R and selects a single action:

$$V(R(\mathbf{l}(t))) = V(\{\mathbf{p}_1(t), \mathbf{p}_2(t), ..., \mathbf{p}_k(t)\}) = \mathbf{p}_i(t) \tag{4}$$

where $1 \leq i \leq k$. In terms of mapping between input and output spaces, the value system is a mapping from the power set of $\mathcal{P}$ to the space of $\mathcal{P}$, that is, $V : 2^{\mathcal{P}} \mapsto \mathcal{P}$.

In summary, the two mappings, R and V, accomplish a composite mapping from the space of last contexts $\mathcal{L}$ to the space of primed action $\mathcal{P}$

$$\mathcal{L} \stackrel{R}{\mapsto} 2^{\mathcal{P}} \stackrel{V}{\mapsto} \mathcal{P}. \tag{5}$$

The mapping R is not static, since it is updated at every time instant t.

D. Simpler regressors

What the cortex does is regression, from the input vector space to the vector output space.

The simplest implementation of such a mental architecture is a multi-value lookup table. The static mapping of R is a large lookup table. It digitizes the input space $\mathcal{L}$ into c number of cells (artificial neurons). The prediction space $\mathcal{P}$ is digitized into m cells, $\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_m$, each corresponding to a candidate action. Each cell, represented by a vector $\mathbf{v} \in \mathcal{L}$ in the lookup table is linked to m candidate actions, $\mathbf{a}_i$, each being associated with an estimated value q_i. The simplest value system is to pick up the action $\mathbf{a}_j$ that is associated with the highest value $q_j = \max_{1 \leq i \leq m}\{q_i\}$.

A more efficient adaptation of R can use the Self-Organization Map (SOM) [23], Competitive Learning (CL) [15], Neural Gas [26], or Hierarchical Feature Map [27]. However, these unsupervised methods do not generalize well for high-dimensional inputs because of the lack of a capability to extract discriminant features. Supervised networks can also be used. Feed forward neural networks with incremental back-propagation learning do not have a systematically organized long-term memory, and thus, early samples will be forgotten in later training. Cascade-Correlation Learning Architecture [12] improves them by adding hidden units incrementally and fixing their weights to become permanent feature detectors in the network. Thus, it adds long-term memory. Major problems for them include the high-dimensional inputs and local minima. We will describe the Incremental Hierarchical Discriminant Regression (IHDR) after we present the regression requirements for autonomous mental development.

E. A simple value system

To complete the modeling of the Type-1 mental architecture, we need to model the value system V in Eq. (4). The value system is a selector of multiple predicted contexts according to estimated values.

The most basic innate value system in a new born is characterized by pleasure seeking and pain avoidance. Pleasure is denoted by a positive reward and pain is denoted by a negative reward. However, an immediate reward is typically delayed by a few milliseconds to a few seconds (e.g., from reaching a hot pot to feeling the pain).

Just negative and positive rewards are not sufficient for effective development. When punishment and rewards are absent, novelty is an important component of the value system. Habituation and sensitization in animal learning are examples of novelty in the value system. Typically, punishment (aversive stimuli) and rewards (appetitive stimuli) are temporally sparse and novelty is temporally dense. Huang & Weng [17] [18] formulated novelty is the failure of predicted context and integrated punishment, reward, and novelty in a developmental value system. In their model, sensed value at any time is a combination of weighted sum of aversive stimuli, appetitive stimuli and novelty. Aversive stimuli are counted as negative value, but appetitive stimuli and novelty are counted as positive value. The weights they contribute into the current sensed value are different: aversive stimuli highest, appetitive stimuli second highest and novelty the third.

It seems other non-sensor driven information, such as a desire to learn, is not a part of the developmental program. A desire to learn is a learned behavior, if learning is novel or receiving proper rewards from the environments.

Sensed value at time with short delays can be dealt with by the well-known Q-learning algorithm [38] which uses a time discount model to handle delayed sensed values. The major limitations in the application of Q-algorithm for autonomous development include (1) much delayed values should be avoided due to the time discount model favor small immediate values to large far values, (2) the predicted value of a new action must be well estimated before the next execution. During application experiments, we used rewards that immediately follow the corresponding actions, and the architecture keeps a first-in-first-out queue, called state-update-queue [41] to update the Q-values along the so called eligible trace (the last few visited states, i.e., quantized context states) multiple times before each state goes out of the state-update-queue.

Sensed values that are very much delayed cannot be effectively dealt with by the simple Q-learning algorithm alone. For example, in animal training, there is a significant delay between starting following the instruction of the human trainer to receiving a food reward because of a successful practice. An effective trainer designs a training schedule that involves many short-delayed rewards, rewarding every small step instead of just a single big reward at the end without any small rewards in between. Therefore, a well designed

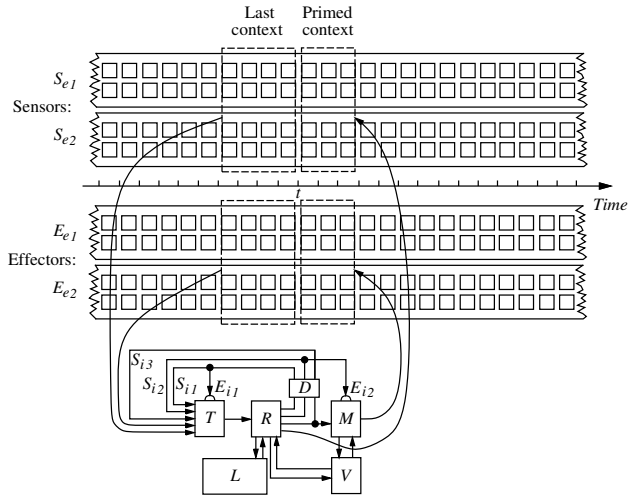

Fig. 2. Progressive additions of architecture components from Type-2 to Type-5. Type-2: adding attention selector T and its (internal) control input E_{i1}. Type-3: Adding motor mapping M and its (internal) control E_{i2}. Type-4: Adding internal controls S_{i1} and S_{i2} and the primed sensation S_{i3} to the entry port of perception T. The block marked with D is a delay module, which introduces a unit-time delay for the corresponding vector. Type-5: Developmental T, R, M and V.

training procedure is very important to train a developmental machine.

Other longer delays of rewards and more sophisticated value associated with the type of rewards need the mechanism of autonomous thinking, similar to the reasoning mechanism in [52].

F. Mental architectures Types 2 to 6

Other more sophisticated architectures are closer models of human mental architecture, but their autonomous learning is more complicated. The following presents mental architectures 2 through 6 which appeared in Weng 2004 [42] with more detail available in [44].

A Type-2 **Observation-driven Selective MDP** is a Type-1 architecture, with the addition of an attention selector T, more generally called sensory mapping a biological equivalent of early sensory processing as shown in Fig. 2. The control signal for attention selection is internal actions generated from the action end.

A Type-3 **Observation-driven Selective Rehearsable MDP** is a Type-2 mental architecture, with the addition of an action releaser M, a biological equivalent of pre-motor and motor cortex, as shown in Fig. 2. It enables the agent to internally rehearse an action without actually executing it, important for reasoning, planning and autonomous thinking.

A Type-4 **Observation-driven Self-Aware Self-Effecting (SASE) MDP** is a type-4 architecture but is further SASE. The term "self" here means the brain, instead of the body of the agent. A self-aware and self-effecting (SASE) agent has internal sensors S_i and internal effectors E_i for this internal (brain) environment, in addition to its external sensors S_e and external effectors E_e for its external environment (outside brain), as

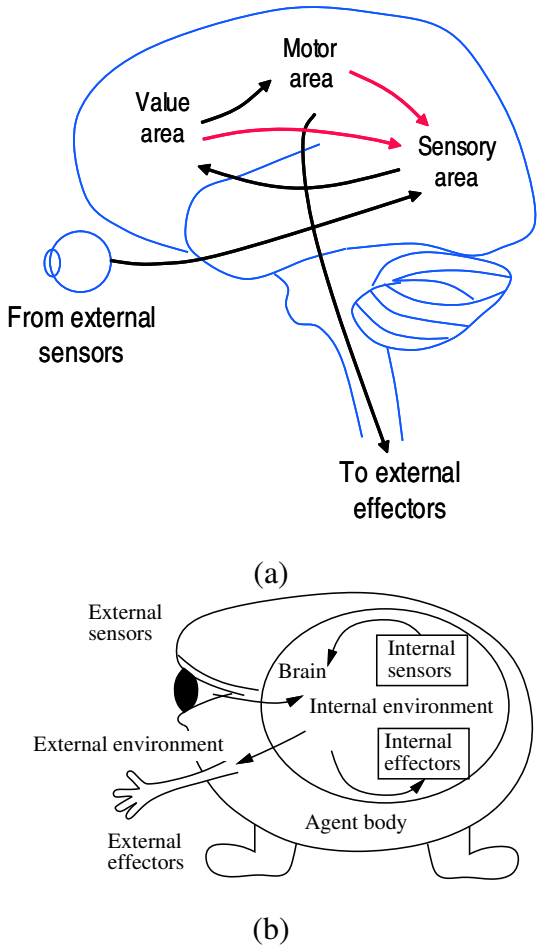

Fig. 3. (a) The brain is a SASE Agent. It interacts with not only the external environment but also its own internal (brain) environment: the representation of the brain itself. The afferent (out-going) projections from the motor area and the value area correspond to equivalent internal sensors. The efferent (receiving) projections from the motor area to the sensory area correspond to equivalent internal effectors. (b) The equivalent internal sensors and internal effectors of a SASE agent.

shown in Fig. 3. The regressor R takes signals from S_i and S_e and generates internal and external actions for E_i and E_e, respectively, as shown in Fig.2.

A Type-5 **Developmental Observation-driven SASE MDP** (DOSASE MDP) architecture is a Type-4 architecture but further it satisfies the following requirements:

1. During the programming time, the tasks that the agent will learn are unknown to the programmer.
2. The agent $A(t)$ starts to run at $t = 0$ under the guidance of its developmental program P_d. After the "birth," the brain of the agent is not accessible to humans.
3. Human teachers can only affect the agent $A(t)$ as a part of its environment through its sensors and effectors recursively. $A(t-1)$ is updated to $A(t)$, including T, R (and L), M, and V, as shown in Fig. 2.

A Type-6 Multi-level DOSASE MDP architecture is composed of several (sensorimotor) levels of Type-5 architecture. The primed contexts from a lower-level system are fed into the sensory input of the higher-level

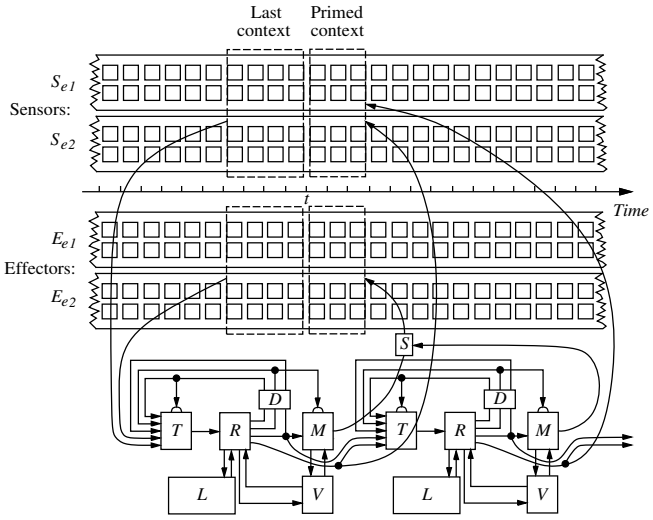

Fig. 4. The Type-6 architecture: multi-level DOSASE MDP.

system, as shown in Fig. 4.

In the above systematically introduced six types of architectures, the order at which new capabilities are added to the previous type is primarily a design choice. The order used here is motivated by a relatively large payoff in capability with a minimal addition to the architecture complexity. The higher the architecture type, typically the agent will able to generalize better given the same environment.

In summary, the major characteristics of mental architectures include: observation driven, selective, rehearsable, self-aware, self-effecting, multi-level and developmental. The above definition of Types 1 through 6 only gives a particular order in which these features are incorporated in an increasing complete sequence of mental architectures. Other combinations of these architecture features are also possible. For example, the developmental feature can be incorporated without adding all other features, like the SAIL-2 robot architecture explained later in the paper.

CALL OUT: *The major characteristics of mental architectures include: observation driven, selective, rehearsable, self-aware, self-effecting, multi-level and developmental.*

Of course, the added capabilities require training, but the mental skills from the training will be used for better generalization in future training, although there is no guarantee that the generalization will always be correct, a well known phenomena in child learning.

G. Learning types

In the machine learning literature, there have been widely accepted definitions of learning types, such as supervised, unsupervised, and reinforcement learning. However, these conventional definitions are too coarse to describe computational learning through autonomous development. For example, it is difficult to

identify any type of learning that is completely unsupervised. Further, the traditional classification of animal learning models, such as classical conditioning and instrumental conditioning, is not sufficient to address computational considerations of every time instant of learning. A definition of a refined classification of learning types is necessary.

We use a variable i to indicate *i*nternal task-specific representation imposed by human programmer (called *internal-state imposed* $i = 1$) or not (called *internal-state autonomous* $i = 0$).

We use e to denote autonomy of effector. If the concerned effector is directly guided by the human teacher or other teaching mechanisms for the desired action, we call the situation *action imposed* ($e = 1$). Otherwise, the learning is effector autonomous ($e = 0$).

We need to distinguish the channels of reward (e.g., sweet and pain sensors) that are available at the birth time, and other channels of reward that are not ready to be used as reward at the birth time (e.g., auditory input "good" or "bad") but implies a value after a certain amount of development. We define (inborn) biased sensors:

If the machine has a predefined preference pattern to the signals from a sensor at the birth time, this sensor is an *(inborn) biased sensor*. Otherwise, it is an *(inborn) unbiased sensor*.

In fact, all the sensors become biased gradually through postnatal experience — the development of the value system. For example, the image of a flower does not give a newborn baby much reward, but the same image becomes pleasant to look at (high value) after the baby has gown up.

We use the third variable b to denote whether a biased sensor is used. If any biased sensor is activated (sensed) during the learning, we called the situation *reinforcement* ($b = 1$). Otherwise, the learning is called *communicative* ($b = 0$).

Using these three key factors, any type of learning can be represented by a 3-tuple (i, e, b), which contains three components i, e, and b, each of which can be either represented by 0 or 1. Thus, there are a total of 8 different 3-tuples, representing a total of 8 different learning types. If we consider ieb as three binary bits of the type index number of learning type, we have 8 types of learning defined in Table II. We can also name each type. For example, Type 0 is state-autonomous, effector-autonomous, communicative learning. Type 7 is state-imposable, effector-imposed, reinforcement learning, but it has not been included in the traditional definition of either supervised learning or reinforcement learning. However, this learning is useful when teaching a positive or negative lesson through supervision.

CALL OUT: *Using three key features, state-imposed, effector-imposed and reinforcement, eight learning types are defined. This refined definition is necessary to understanding various modes of developmental and nondevelopmental learning.*

TABLE II

EIGHT TYPES OF LEARNING

Type (binary)	Internal state	Effector	Biased sensor
0 (000)	Autonomous	Autonomous	Communicative
1 (001)	Autonomous	Autonomous	Reinforcement
2 (010)	Autonomous	Imposed	Communicative
3 (011)	Autonomous	Imposed	Reinforcement
4 (100)	Imposable	Autonomous	Communicative
5 (101)	Imposable	Autonomous	Reinforcement
6 (110)	Imposable	Imposed	Communicative
7 (111)	Imposable	Imposed	Reinforcement

All learning types using a non-developmental learning method corresponding to Types 7 to 4, this is because the task-specific representation is at least partially handcrafted after the task is given. Autonomous mental development uses Types 0 to 3.

IV. A CORTICAL DEVELOPMENT ENGINE: IHDR

Let us examine some requirements for the regressor R for autonomous development.

A. Regression requirements

It is desirable for the regression engine R to have the following 7 properties, called 7 *regression requirements*. We discuss how the IHDR address each requirement.

1. The regression engine must deal with high-dimensional inputs with very complex correlation between components in the input vector (e.g., an image vector has over 5000 dimensions). Some input components are not related to output at all (e.g., the posters on the wall is not related to navigation). In other words, it must automatically derive the most relevant, discriminating features for superior generalization. IHDR automatically derive the most discriminant features subspaces in every node of the tree.

2. It must perform one-instance learning. An event represented by only one input sensory frame needs to be learned and recalled. The lookup-table regression satisfies this requirement. IHDR stores clusters as prototypes, but not all the learned samples. Thus, one-instance learning is realized by either a new prototype, or a similar prototype.

3. It must adapt to increasing complexity dynamically. It cannot have a fixed number of parameters, like a traditional neural network, since such a program must dynamically create system parameters to adapt to regions where increased complexity is needed due to, e.g., increased practice for some tasks. The lookup-

table regression satisfies this requirement, since it does not have internally designed parameters that restrict the degree of freedoms. IHDR dynamically grows subtrees to adapt to the increased complexity.

4. It must deal with the local minima problem. In methods that use traditional feed-forward neural networks with back-propagation learning, there is no guarantee to find the best match. The lookup table method satisfies this requirement since it exhaustively search for the nearest neighbor. IHDR uses a data-driven coarse-to-fine search tree to contain the local minima problem.

5. It must be incremental. The input must be discarded as soon as it is used for updating the memory. It is impossible to keep all the training samples during open-ended incremental development. The lookup table method keeps all the samples, thus, it is incremental but the space needed is too large to be practical. IHDR incrementally updates.

6. It must be able to retain most of the information of the long-term memory without catastrophic memory loss. However, it must also forget and neglect unrelated details for memory efficiency and generalization. With an artificial network with back-propagation learning, the effect of old samples will be lost if these samples do not appear later. The lookup table method keeps every learned sample, thus, it does not suffer from catastrophic memory loss. In IHDR, the long term memory stored as tree structure and micro-prototypes prevent catastrophic memory loss.

7. It must have a very low time complexity in computing and update so that the response time is within a fraction of second, even if the memory size has grown very large. Thus, any slow learning algorithm is not applicable here. The time complexity of the lookup table method is linear in the total size of the tree size, which increase linearly in time until it reaches the soft limit. Therefore, it is impossible for the lookup table method to be real-time when the memory size has grown the very large. Given a input vector $\mathbf{l}(t)$, the time complexity for IHDR to find a match and updating the IHDR tree with n leaf nodes is $O(d \log(n))$ where d is the constant dimension of $\mathbf{l}(t)$. Thus, when n is very large, the time complexity for update is still small (very slow logarithmic growth).

CALL OUT: *The seven regression requirements for cortical mapping and development are: high-dimension, one-instance learning, adaptation to complexity, freedom of local minima, incremental learning, long-term memory, and low time complexity.*

B. Incremental Hierarchical Discriminant Regression

Taking all of these requirements into account is very challenging. Existing regressors were not designed for all 7 requirements, including the Self-Organization Map (SOM) [23], Competitive Learning (CL) [15], Neural Gas [26], Hierarchical Feature Map [27], feed forward neural networks with incremental back-propagation learning, the Cascade-Correlation Learning Architecture [12] and the Support Vector Machines (SVM). For

example, the support vector machine SVM has problems with requirements 3 (because of batch learning) and 5 (because of batch learning). Of course, some are better than others in those 7 regression requirements.

These 7 requirements have been taken into account in the Incremental Hierarchical Discriminant Regression (IHDR) mapping engine as shown in Fig. 5.

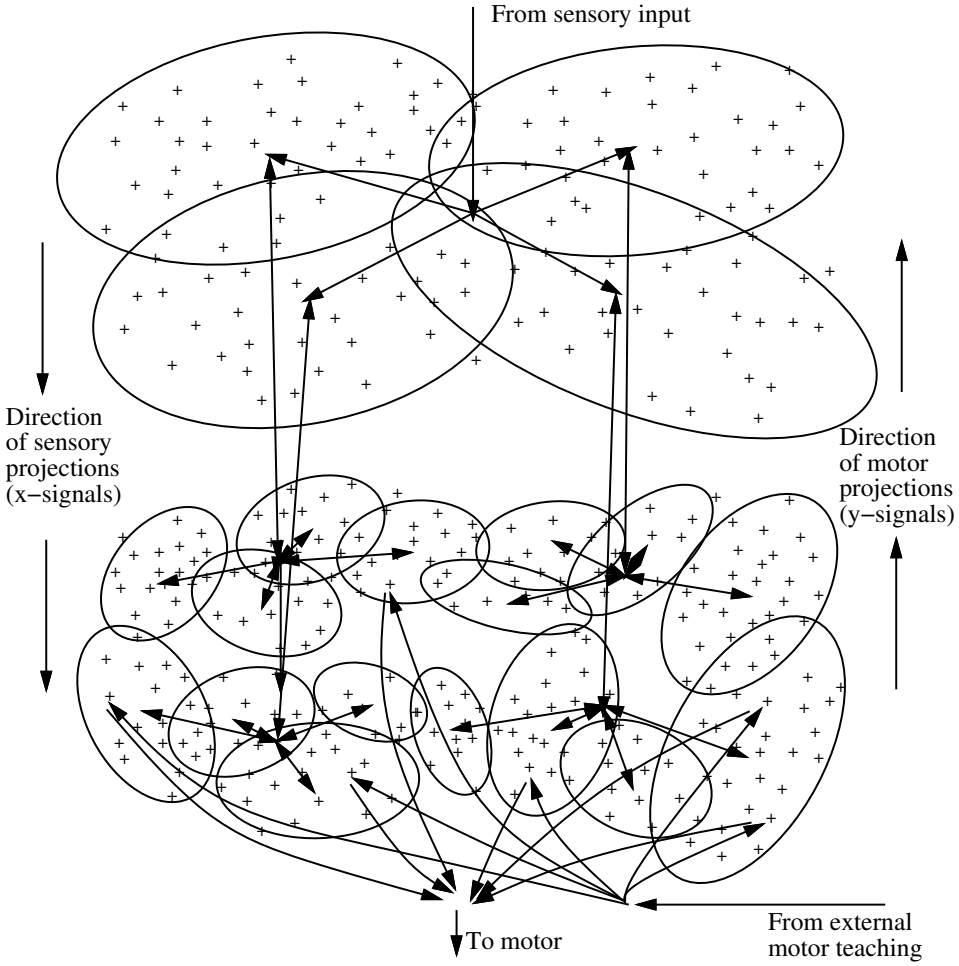

Fig. 5. Illustration of an IHDR tree incrementally developed (self-organized) from learning experience. The sensory space is represented by sensory input vectors, denoted by "+". The sensory space is repeatedly partitioned in a coarse-to-fine way. The upper level and lower level represent the same sensory space, but the lower level partitions the space finer than the upper level does. An ellipse indicates the space for which the neuron at the center is responsible. Each node (cortical region) has q feature detectors (neurons) ($q = 4$ in the figure), which collectively determine to which child node (cortical region) the current sensory input belongs (excites). An arrow indicates a possible path of the signal flow. In every leaf node, prototypes (marked by "+") are kept, each of which is associated with the desired motor output.

Given any input $\mathbf{x}$, IHDR uses a supervised learning to update the tree if the desired output $\mathbf{y}$ is available. Otherwise, it retrieves the last updated tree using the current input $\mathbf{x}$ from the root. The most discriminating subspaces at each level of the tree guide the search only along a single path (multiple paths corresponding to an option) down the tree until the best matched leaf node is reached. In this leaf node, the best matched

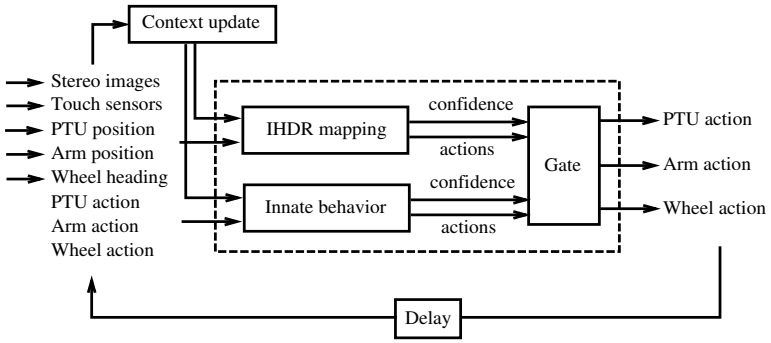

Fig. 6. A schematic illustration of the architecture of the SAIL-2 robot.

prototype $\mathbf{x}'$ is associated with multiple action clusters each with an estimated value. These action-value pairs are the corresponding output of the input $\mathbf{x}$.

IHDR is a model for cortical learning and cortical development. To deal with a high-dimensional input, IHDR automatically derive the most discriminating feature subspaces using virtual labels from the incremental clustering in the output space. To deal with one-instance learning, it spawn a new prototype in the best matched leaf node whenever the current input is sufficient different from the best-matched existing prototypes. To adapt to increasing complexity dynamically, IHDR has dynamic set of parameters, represented by the adaptive clusters in every internal node and the increasing clusters in the leaf nodes. To deal with the local minima problem, IHDR uses data-driven coarse-to-fine approximation of the input distribution. Therefore, there is an approximation precision, limited by the observations, but not the gross local minima problem in a global fitting method (e.g., batch propagation algorithm). To deal with incremental learning, the entire tree of IHDR is built incrementally by updating the related part of the tree using each input and output vector pair. IHDR has long-term memory as the tree structure, and the cluster parameters in the shallow nodes. IHDR has a very low time complexity $O(d \log(n))$ for computing response and update the tree from each input, where d is the input dimension and n is the number of nodes in the tree. More detailed description of IHDR is available in [19] [47].

C. SAIL-2 online learning architecture

Fig. 6 gives a schematic illustration of the implemented SAIL-2 architecture. The block marked by dashed lines realizes the mapping R followed by R in Eq. 5. The left incoming arrows denote the composition of the last context vector $\mathbf{l}(t)$. The right outgoing arrows denote the action output vector $\mathbf{p}(t)$.

Based our discussion of mental architecture, we can see that the SAIL-2 architecture is a developmental, observation driven, Markov decision process. It is not selective, rehearsable, or SASE (e.g., it does not self-sensed internal actions). Although it has two pathways of sensorimotor systems (IHDR mapping and

Innate behavior pathways), these two pathways do not feed the prediction into another. Therefore, it is not a multilevel architecture as defined for Type-6.

Although actual sensors and effectors are marked in Fig. 6, the SAIL-2 architecture is general. Any robot can use this architecture as long as the appropriate sensors and effectors are filled in.

It is important to note that every effector that the brain needs to actively control must have the corresponding sensor input in $x(t)$. For example, if the robot controls the wheel action, it must have a dedicated virtual sensor that senses the wheel action. Otherwise, its context vector does not contain the information about the wheel. When the sensory inputs are very similar around a corner, the wheel heading can tell whether the turn has started or not. In the architecture in Fig. 6, there are three effectors, pan-tilt units (PTU) for the eyes, the arm, and the wheels, each having corresponding sensors which tell, at the next time instant, what the robot does through a delay unit.

D. Innate and learned behaviors

An innate behavior is programmed prior to the "birth" of the robot. The current implemented built-in innate behavior is the motion detection and tracking mechanisms for vision. When an object is moving in the scene, the absolute difference of each pixel between two consecutive image frames records the amount of intensity change for the pixel. The absolute intensity changes of all the pixels constitute another image called the intensity-change image, which is directly mapped to the control of the PTU of each eye, also using the IHDR mapping technique. However, this mapping was generated by a "prenatal" offline learning process. In other words, this offline learning generates innate behaviors in robot "newborns." Our experience indicated that it is computationally much faster and more reliable to generate innate behaviors this way than explicitly finding the regions of moved objects through direct hand programming.

The online generated IHDR mapping and the innate behavior may generate PTU motion signals at the same time. The resolution of such a conflict is performed by the gate system. In the current implementation, the gate system performs subsumption [6]. Namely, the learned behavior takes higher priority. Only when the learned behavior does not produce actions, can the innate behavior be executed. A more resilient way of conducting subsumption is to use the confidence of each action source, as indicated in Fig. 6, but we did not find it was necessary in the experiments conducted. This confidence-based action arbitration may be needed when the number of tasks learned becomes large, but this subject is beyond the scope of the reported work.

V. EXPERIMENTS

The purpose of this section is to provide a concrete example of application. We first describe the SAIL robot which was used as the developmental platform of this example.

A. SAIL robot

Fig. 7. The SAIL robot.

The human-size SAIL robot, house built at Michigan State University, is shown in Fig. 7. It is equipped with two CCD color cameras, each with a wide-angle lens. Its "neck" can turn. Each of its two "eyes" is controlled by a fast pan-tilt unit. Its torso has 4 pressure sensors to sense push actions and force. It has 32 touch sensors on its arm, neck, head, and bumper to allow humans to teach it how to act by direct touch. A six-joint robot arm is the robot's manipulator. Its drive-base is driven by two independent drive wheels, one on each side. The base is adapted from a wheel-chair and, thus, the SAIL robot can operate both indoor and outdoor. The main computer is a high-end dual-processor, dual-bus Pentium II 333Mhz PC workstation with 512MB RAM and an internal 27GB three-drive disk array for real-time sensory information processing, real-time memory recall and update as well as real-time effector controls. This platform was used to test the SAIL-2 program.

At the "birth" time of the SAIL robot, its program starts to run. This algorithm runs in real-time, through the "life span" of the robot. The robot learns while performing simultaneously. The innate reflexive behaviors enable it to explore the environment while improving its skills. Human trainers educate the robot by interacting with it, very much like how human parents interact with their infants, letting it look around, demonstrating how to reach objects, teaching commands with the required responses, delivering reward or punishment (pressing "good" or "bad" buttons on the robot), etc.

The SAIL robot has been trained for four tasks, (1) vision-guided navigation using supervised and reinforcement teaching methods, (2) object-finding, (3), object-based attention (eye-pan), and (4) attention-guide arm pre-reaching. The first task is to test the power and limit of perception-based behavior learning, with an emphasis on the complexity of perceptual capability. The tasks (2) - (4) were trained in the same setting, but different from that of task (1), with an emphasis on demonstrating earlier skills serving as "scaffold" for the later ones: Recognizing object guides turning the eyes toward the object, which in turn provides the context

for arm pre-reaching. Within tasks (2) to (4), it is much harder to learn the later tasks without accomplishing the earlier ones.

B. Task 1: Vision-guided autonomous navigation

At the birth time, the "baby" robot does not "know" how to move around. We took it out of the laboratory for a "walk."

At first we taught the robot using a supervised learning technique: pushing it around the corridors of MSU's Engineering Building, shown in Fig. 8, as it learns online. The human teacher pushes the pressure

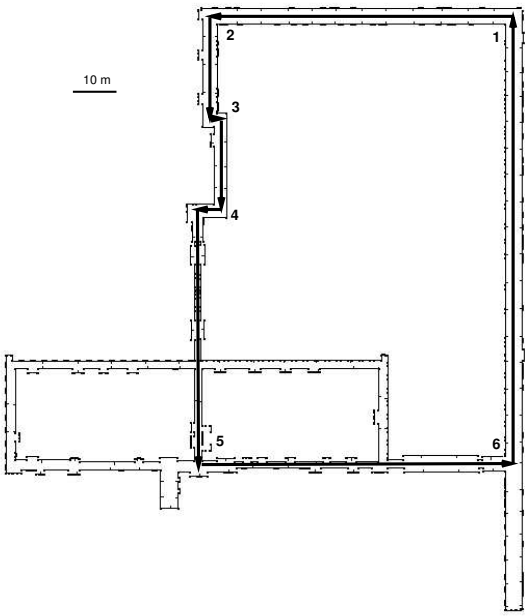

Fig. 8. The long navigation path along the corridors of the Engineering Building at Michigan State University.

sensors, one on each shoulder of the robot. The pressure is translated into the speed of the driving wheel on the corresponding side. The difference of the two readings gives the heading direction of the robot. The corridors exhibit various posters, doors (including a glass fire door), floor tiles, ceiling patterns, and "moving" floor reflections. Figs. 9 and 10 show some views at one straight section around one corner. The complex scene changes gradually when the robot is pushed along. If the human does not push the robot, it navigates on its own based on what it has learned so far. The SAIL baby robot associates what it sees with what it does in real-time. This is very challenging. First, no range-sensor, whatsoever, is used. The mapping from the intensity images to what it does is extremely complex. Second, the robot never sees exactly the same image twice. It must generalize from its experience. When the robot does well, the human teacher lets it go free more often. We found that it took about 10 trips for the robot to learn to navigate reliably around a corner, but

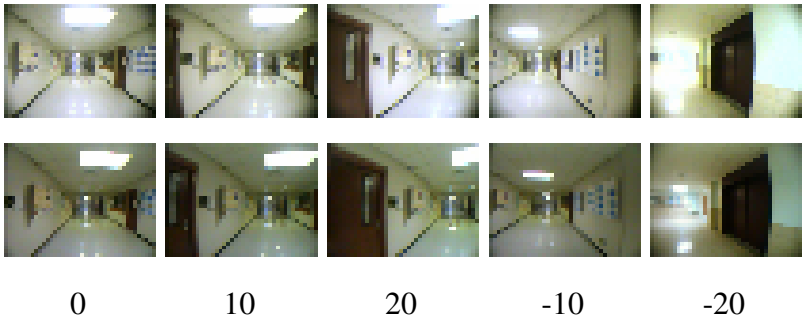

Fig. 9. The SAIL robot is at a straight section: a subset of corridor images that it sees. The number below the images shows the needed heading direction (in degrees) associated with that image. The first row shows the images from the right camera while the second row shows those from the left camera. Notice the doors and posters on the sides and "moving" reflections on the floor.

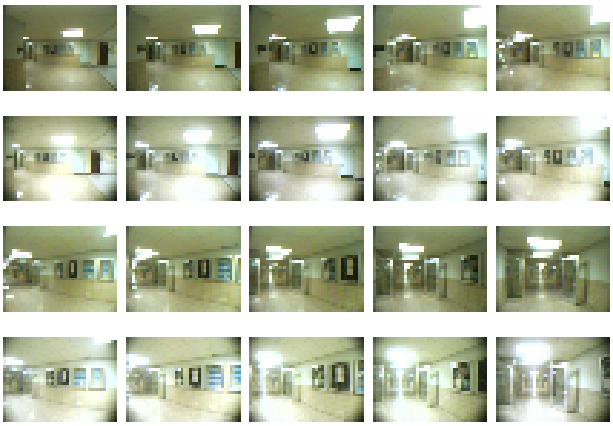

Fig. 10. The SAIL robot is at a corner: A subset of images it sees. Rows one and three show the images from the left camera. Rows two and four show the images taken from the right camera. Notice the posters on the wall and the glass fire door.

one trip is mostly sufficient to learn along a straight section. When the robot does better, the trainer lets it go free completely. Fig. 11 shows the SAIL robot autonomously navigating on its own around a corner, before going through a glass fire door.

What happens inside the SAIL-2 program is as follows. At each time instance, the SAIL-2 program accepts a pair of stereo images, updates its states, which contains past sensory inputs and actions, and then outputs the control signal C to update the heading direction and speed. The resolution of each image is 30×40. The input dimensionality of the IHDR algorithm is $30 \times 40 \times 3 \times 2 = 7200$, where 3 is the length of history in state (context). This is a very challenging approximation problem since the function to be approximated is for a very high dimensional input space and the real application requires the navigator to perform in real-time.

The SAIL robot was trained through 10 laps on the second floor of the Engineering Building. The test showed that the SAIL robot can successfully navigate after the interactive training, except for occasional

Fig. 11. The SAIL robot navigating around the corner of the Engineering Building at MSU, on its way to circle the entire floor.

mistakes at five locations. At these locations, there are extensive wall-height windows which caused strong ambient outdoor lighting. The fixed exposure cameras became over-saturated and the navigation at these points were sometimes unreliable. Due to the unavailability, at the time of the robot's construction, of micro-cameras with auto exposure, the microcameras on the SAIL robot can only use a fixed exposure. We plan to acquire auto-iris cameras for the SAIL robot. The SHOSLIF robot, the predecessor of SAIL, equipped with an automatic exposure camera, has also used the appearance-based approach. However it did not make mistakes at these locations [46] [8]. As demonstrated in face recognition under different lighting conditions [19], HDR can deal with lighting variations if they are well included in the training samples. However, the SAIL robot demonstrated flawless navigation along the very long navigation path, as long as the cameras were not over-saturated, although many of the sections of the corridors take outdoor light in through classroom windows.

It is impossible to measure error in the heading direction from a real-time system, since the exact desired heading direction is not available. For performance evaluation, we recorded the training experience and run in simulation. Some of the recorded sample input images are shown in Fig 10. A total of 3533 color stereo images with the corresponding heading directions were used for simulated training. A set of remaining 3386 stereo images, labeled with desired heading directions were used to measure the error in heading. Fig 12 shows the error rate versus the number of training epochs, where each epoch corresponds to the feeding of the entire training sensory sequence once (one lap around the building). As shown, even after the first epoch, the performance of the IHDR tree is already reasonably good. With the increase in the number of epochs, we observed the improvements of the error rate. Fig. 13 shows a more detailed error histogram after 16 epochs.

A part of the automatically generated IHDR tree is plotted in Fig 14. During the training, the IHDR algorithm receives both the color stereo images as input and the heading direction as output. It rejects samples

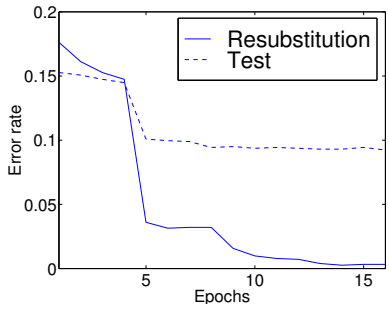

Fig. 12. Heading error rates vs. epochs in vision-guided navigation.

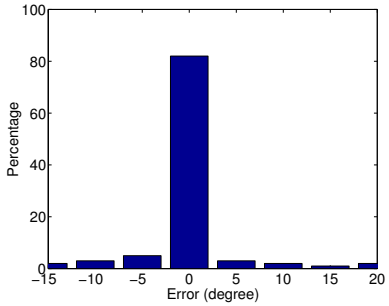

Fig. 13. The error histogram of the testing set after 16 epochs.

(not used for learning) if the input images are similar to samples already learned. The visualization of the IHDR tree provides us an example about the concepts of Information Levels discussed in Section II:

1. The coarse architecture of the SAIL's CNS is designed, as shown in Fig. 6, but the fine architecture, corresponding to the IHDR mapping (a network), is developed automatically from the experience, guided by the SAIL-2 program, including IHDR.

2. The representation level corresponds to the clusters in the nodes of the IHDR tree. The IHDR representation depends on the designed and developed architecture, but also depends very much on the experience. Different experience results in a different tree. Hand-designed features such as Gabor wavelets are not task-adaptive, but the discriminant feature vectors shown in Fig. 14 are architecture-adaptive (growing according to the current fine architecture) and task-adaptive. That is why the former is applicable to only early pre-processing, but the latter is applicable to everywhere from early to later processing. Further, the former does not have a discrimination optimality, but the latter does.

3. Different branches of the tree may be considered, in some sense, as function-specific regions (modules) of the "brain."

4. The representation is very much distributed all over the tree. For example, a physical corner may not necessarily be represented by a single monolithic form. Instead, it may be represented by different nodes at different locations with different degrees of specificity.

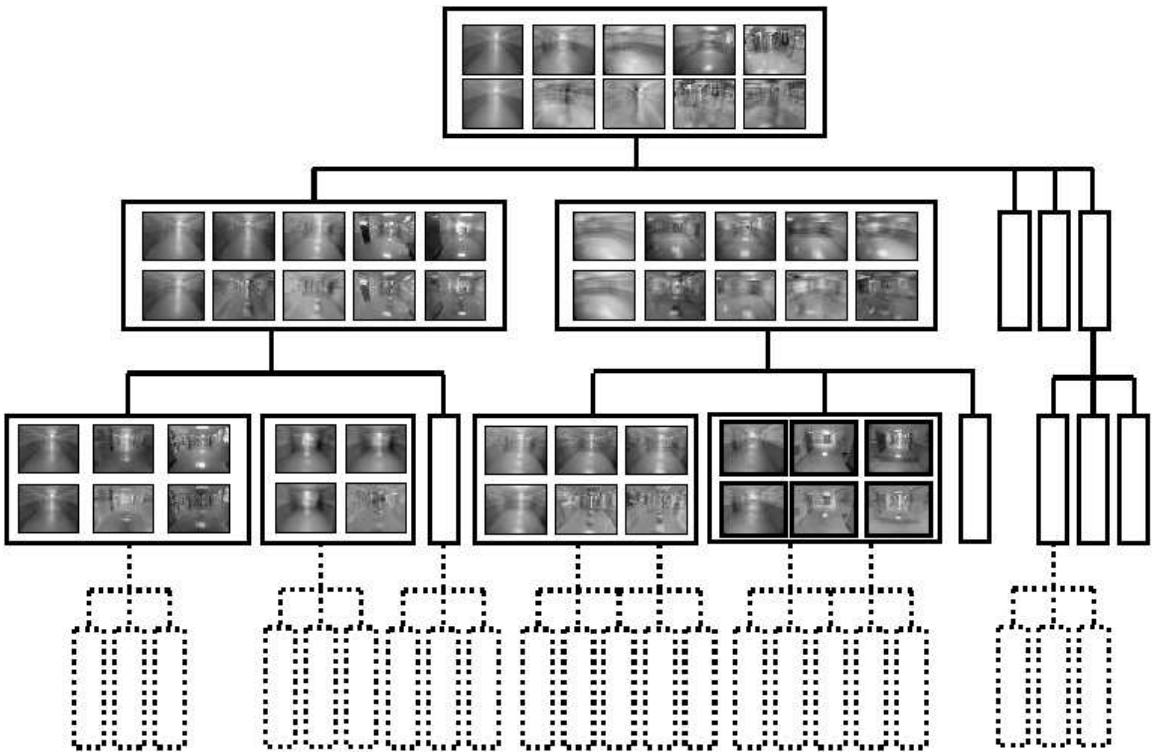

Fig. 14. An illustration of the IHDR tree automatically developed from real-time experience, guided by the SAIL-2 developmental program. For conciseness, only training images from the left camera are shown. Each block indicates a tree node. The first row of each node shows the x-cluster centers presented as images. The first image of the second row is the grand mean of all the x-clusters. The remaining images of the second row are the discriminating features represented as images. See text for discussion.

5. The actual representation used to generate behaviors changes very fast through time. Within 100 milliseconds, the retrieval search path changes from one to another through the tree, each time using a very different sequence of nodes together with their specialized subspaces. There does not seem to exist a practical way to hand-design such a complex representation, even if the scene is known beforehand.

6. "Scaffolding" occurs automatically: the current network enables the following action and newer cognitive and behavioral capabilities. For example, the capability of making a turn around the first corner provides a "scaffold" for turning the second time around the same corner and turning around the second corner. In fact, at the high dimensional space, every view of a corner is different. Generalization is observed at the response level: the observed behaviors. Very complex changes: posters, doors, floors are automatically disregarded. The invariance is achieved through a mapping from many different corresponding leaf nodes to the desired behavior output (e.g., heading direction). Invariance is learned through experience instead of relying on human selected intermediate features, which is hopeless for complex tasks like the one above. E.g., floor edges are often not clear (see Fig. 10) and they are mixed with the edges of the wall tiles and floor tiles. Thus,

navigation relying on floor edges is doomed to fail.

7. At the timing level, internal nodes are frozen so that the samples that are channeled into deeper nodes come from a steady partition (or distribution). Further, the IHDR program monitors the percentage of the physical memory used and it adjusts the speed of the node-deleting (forgetting) process to guarantee that the robot performs well without memory overflow. Therefore, the timing level is still experience-dependent.

In order to teach the robot for more experiences, we allowed it to try on its own. We moved the robot to the third floor of the Engineering Building and trained it further using a combination of supervised and reinforcement teaching techniques. If the robot's actions show a danger of making an error, a negative reward is given by pressing its "bad" button and a push corrects its heading, if needed. If the robot does well, a "good" button is pressed, or it is simply allowed to go ahead without reward. The reward pattern from the trainer discourages changes toward the wrong direction and encourages those toward a better one. While the robot's performance is being improved, the trainer gave fewer and fewer rewards. The robot experiment showed that the reinforcement learning was effective. The developed "brain" size is under 300 MB.

To accurately evaluate the effect of reinforcement learning in the SAIL-2 program, we have developed a simulation environment whose graphical user interface is shown in Fig. 15. If the robot navigates well within the navigation path, human intervention is not necessary. If the robot is heading off the road, the human teacher will press the "bad" button. If the robot is correcting its heading direction toward a desired direction, a "good" button is pressed. If the robot does well, no human intervention is given. Fig. 16 shows the improvement of navigation through the laps around the navigation loop, using exclusively reinforcement learning.

C. Tasks 2 through 4: Object finding, eye-pan, and pre-reaching

The teacher now has three more tasks in mind: Task (2) object-finding: find object in the field of view. Task (3) vision-guided attention: eyes pan together with the arm to make the eyes aim toward a familiar object. Task (4) attention-based pre-reaching: After the above is done, the arm is aiming at the object. The arm reaches forward to the object. The later tasks cannot be learned effectively without the help of skills in the earlier ones. Thus, the earlier skills provide scaffolds for the later ones. These three tasks were learned in a laboratory environment, which is very different from the navigation environment outside the laboratory for Task 1.

Existing studies on visual attention selection are typically based on low-level saliency measures, such as edges and texture [4]. In Birnbaum's work [5], the visual attention is based on the need to explore geometrical structure in the scene. In our case, the visual attention selection is a result of past learning experience. Thus, we do not need to define any task-specific saliency features. As we know, it is the SAIL robot that

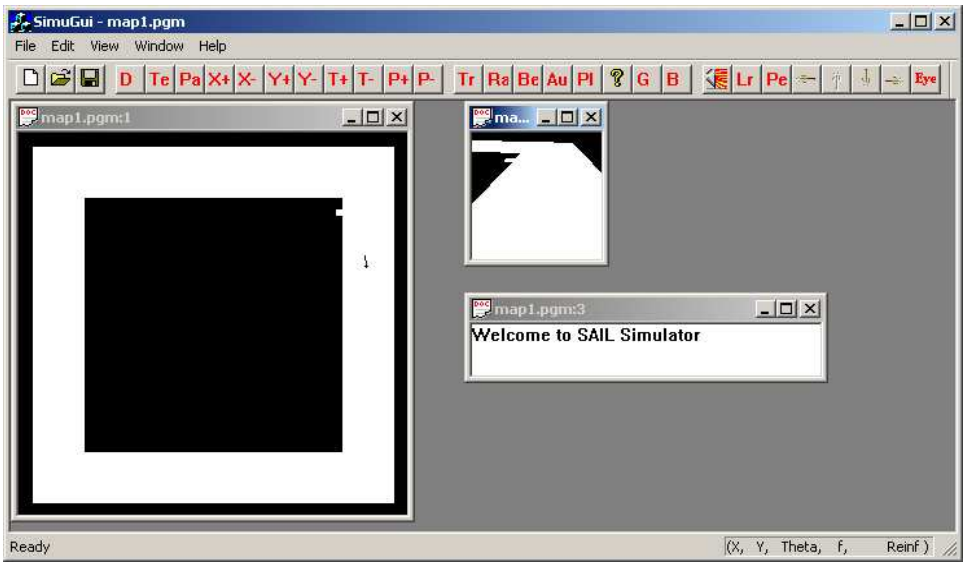

Fig. 15. The SAIL simulator for autonomous navigation, allowing supervised and reinforcement learning. The large window shows the top view of the navigation environment, where a needle marks the position and heading direction of the robot, and the small window shows what the robot sees.

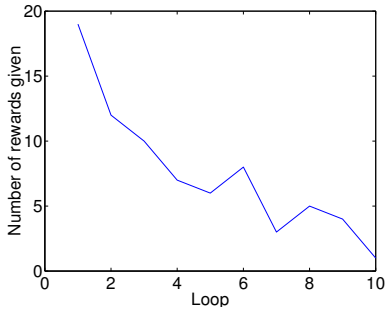

Fig. 16. The simulation results with reinforcement learning. The x-axis shows the number of laps the robot went through along the navigation path. The y-axis indicates the number of rewards the robot received due to the need of human intervention, which decreases while the robot's performance improves.

automatically derives the most discriminating features for the tasks being learned. At the time of learning, the object was presented in the Region Of Interest (ROI) inside the stereo images (the concept of ROI is only in the mind of the trainer but is not used for programming the SAIL-2 program). The human trainer interactively "pulls" the robot's eyes toward the object (through the touch sensors for the pan-tilt heads) so that the object is located on the center of the region of ROI — fovea (fixating the eyes on the object)[1]. The inputs to the algorithm are the continuous sequence of stereo images and the sequence of the pan-tilt head control signal. The arm pans with the head so that it aims at the object when the eyes stop. Three actions are defined for the pan-tilt head in pan direction: 0 (stop), 1 (move to the left), or -1 (move to the right). The size

[1]This is not done with human infants, since we cannot pull an infant's eye. However, this makes robot learning faster than that of a human baby. This is in fact an advantage of robots over humans in that the robot can be built to fascinate training.

of ROI is defined as 120×320. When the object moves again within the field of view, the learned robot will pan the eyes and arm again to bring it into the fovea. When running continuously in time, the robot appears to be interested in the object, since it keeps tracking it. Fig. 17 shows some example images seen by the robot during the training session for Tasks 2 through 4.

Fig. 17. A subset of image stream observed during training for Tasks 2 through 4. The number below the image shows the PTU position associated with that image.

The training session was run in the following way:

1. When the robot sees a scene that does not contain an object of interest, it does not do anything (default, innate action).
2. An object of interest is presented to the robot as any position within its ROI.
3. The human teacher imposes an appropriate pan action (constant absolute speed) to the eyes and the arm, by pressing a pressure sensor. The two images from the two eyes change due to the eye pan.
4. When the eye pan has reached an appropriate position so that the object of interest is at the center of the fovea, the imposed action disappears so that the pan action stops (innate action takes place) for both the eyes and the arm. Therefore, the arm is taught to synchronize with the eyes' orientation.
5. As soon as the pan action stops, the human teacher presses another pressure sensor to bring the arm forward to the object (which is just in front of it after a proper pan).
6. A press on a "home" sensor triggers an "innate" programmed action sequence for all of the effectors involved to move to their default home location. Another practice session can begin if the teacher desires.

Similarly, the testing session was run as follows:

1. The human tester presents an object that the robot has seen so that it falls into the ROI.
2. Since no action is imposed from the environment, the robot derives the behavior for this context, which is the pan for the eyes and the arm in the learned direction.
3. The images change during the pan, but the learned experience enables the robot to continue to pan, as long

as the object is not at the fovea.

4. When the object is at the fovea, the derived action is to stop (zero pan for the eyes and the arm).
5. The current context brings up the following action: the arm reaches toward the object.
6. A press on the "home" sensor triggers the innate homing action.

What the robot learns is the mapping between context $(x(t), w(t))$ and the action $a(t+1)$ using the updated long term memory $l(t)$. The context state $w(t)$ in the working memory provides crucial information about the recent history. Otherwise, the mapping to be learned is often ambiguous: The same input $x(t)$ requires different action outputs. In the above example, the context state $w(t)$ keeps information about the previous pan action (as a part of last context). If $w(t)$ is not used, only $x(t)$ is the input to the IHDR mapping. The images and the pan position will be very similar near the point where the action should stop. Thus, very similar inputs $x(t)$ require very different action outputs: continue to pan to finish if the object is judged by humans as having not been centered yet, or stop if otherwise. This will make the action output unpredictable near the position for the pan action to stop. The context $w(t)$ tells the direction from which the arm is from (left, right or static), and, thus, resolves such an ambiguity. This example explains why it is difficult to deal with raw signal inputs, much harder than symbolic inputs.

The online training and testing were performed successfully and the robot can perform Tasks 2 through 4 effectively, after interactive training, although the SAIL-2 algorithm was not written for these tasks in particular. Multiple objects have been trained for these three tasks. Note that even if the same object is used, the changes in the position and apparent size of the object are significant. The SAIL-2 program was able to deal with such a large number of changes. The current study aims at feasibility for an early-age robot. A larger number of objects and the associated high performance requires the robot to "live" longer and to mature further.

To quantitatively evaluate the performance of online learning and performance, we recorded the sensory data and studied the performance off-line. Since the online learning algorithm runs indefinitely, does its memory grow without bound? Fig. 18(a) shows the memory usage by the "brain." In the first stage, the tree grows, since the samples are accumulated in the shallow nodes. When the performance of the updated tree is consistent to the desired action, the tree does not grow and, thus, the memory curve becomes flat. The tree will grow only when the imposed action is significantly different from what the tree comes up with. Otherwise, the new inputs only participate in the average of the corresponding cluster, simulating sensorimotor refinement of repeated practice, but there is no need for additional memory. This is a kind of forgetting — without remembering every detail of repeated practice.

How fast does the online program learn? Fig. 18(b) shows the accuracy of the PTU action in terms of the

percentage of field of view. After the third epoch (repeated training), the robot can reliably move the eyes so that the object is at the fovea. Does the algorithm slow down when it has learned more? Fig. 18(c) gives the profile of the average CPU time for each sensory-action update, through a large number of training sequences. The average CPU time for update is within 100 millisecond, meaning that the system runs at about 10 Hz, 10 refresh of sensory input and 10 updated actions per second. Since the IHDR tree is dynamically updated, all of the updating and forgetting are completed within each cycle. This relatively stable time profile is due to the use of the tree structure by IHDR, which results in a logarithmic time complexity. In other words, the depth of the tree is stable.

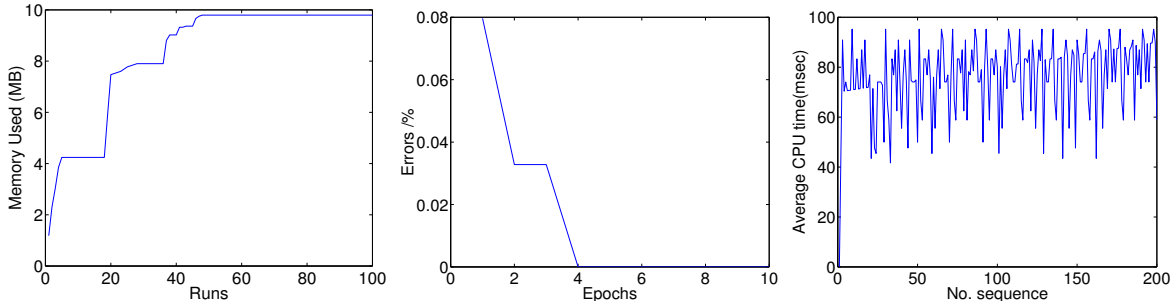

Fig. 18. (a) The memory usage for the off-line simulation of Tasks 2 through 4. (b) The accuracy of object-finding versus the number of training cases. (c) The CPU time for each update.

VI. Scale up: Space and Time Complexities

Development is meant for scaling up. Although we have addressed the time and speed issues of the SAIL-2 program along with actual experimental data, we would like to further comment on the computational scale-up issue for more general settings.

First, an important issue is the demand on storage space by such online learning. The program presented here does not use human designed task-specific tricks but instead autonomously derives task-specific representation using statistical properties of the data. It is possible that such an autonomously generated representation takes larger space than a traditional, human designed representation for a particular constrained, known environment. The SAIL-2 "brain" size for the four tasks is well below 400 MB running in RAM. A RAM of 400MB costs about US$100 at today's street price. Compared with a system that cannot work in an unknown complex human environment, the space storage required by such online learning is a cost that is worth paying for. It is estimated that the storage size of a human adult brain is approximately 10^{14} bytes. This size can be reached by an internal hard disk system at the street price of about US$75,000 by 2004, down 13 times from 6 years before. If the disk price continues to drop at this rate, the hard disk of a human brain size could be

purchased with $6,000 by the year 2008. In other words, many desk-top computers could afford a human brain size storage at that time.

The speed of the computer is another important issue. The SAIL robot's computer engine is a Micron Powerdigm PC with dual 333MHz Pentium II, a fast machine in March 1998 but now considered slow. The logarithmic time complexity of the IHDR mapping engine implies that when the processor speed increases, the number of cases that the IHDR can process in the same time period exponentially explodes. For this matter, we perform a simple exercise which gives a rough estimate but is not very realistic. Our experimental result showed that the SAIL robot ran at a refreshing rate of 10 Hz for a memory of 10 MB = 10^7 bytes, as shown in Fig. 18. Recall that the time requirement for one retrieval is $O(\log_a(m)) \approx O(\log_a(n))$, where m is the number of leaf nodes and n is the total size of the IHDR tree. Suppose that the IHDR tree has grown from 10M bytes to our human brain size of $n = 10^{14}$ but it still wants to keep a refreshing rate of 10 Hz. The required computer speed-up is approximately:

$$\text{speed-up} \approx \frac{O(\log_a(10^{14}))}{O(\log_a(10^7))} \approx \frac{14}{7} = 2, \tag{6}$$

which is a moderate number. This shows the power of logarithmic time complexity of a coarse-to-fine memory structure, such as that of the IHDR. Of course, when more modalities are added, the base cost will also increase, contributing to an increase in the constant factor of the time complexity. It is expected that the speed up of computers can be well used by an increased number of sensing modalities and their integration.

VII. Conclusions and Discussions

The four tasks that the SAIL-2 robot has demonstrated do not seem very difficult judged by a human layman, but they mark a significant methodological and technical advance. First, the environment and the tasks are all muddy. Second, the same online learning program can be continuously used to train other tasks without any re-programming. Third, each individual task has not been achieved by a traditional approach in such a muddy environment. Real-time vision-guided navigation in such an extensive complex indoor environment has not been achieved by other traditional methods so far (see a survey in [46] [8]). Detecting an arbitrary object from an arbitrary background is one of the most challenging tasks for a robot. Note the perception part of these tasks. No traditional robot that we know of can do these tasks without predesigning task-specific representation, let alone real time incremental learning. The main reason for these successes is that the SAIL-2 online learning algorithm does not need a human to design a representation, a holy grail in AI.

Of course, the autonomously developed representation in the SAIL's CNS is not one of the traditional representations which Brooks is against [7]. For example, there is no monolithic "models" in the autonomously

developed representation (see, e.g., [7] [40] for reasons.) The new kind of distributed representation is able to use context intimately. Every action is tightly dependent on the rich information available in the sensory context. The complexity of the rules of such context dependence is also beyond human direct hand programming. A human-designed representation is not able to keep track of such very rich information.

The mental capability of a developmental robot (as well as a human being) in its muddy living environment depends on five factors: (1) its sensors, (2) its effectors, (3) its computational resources, (4) its developmental program, and (5) how the robot is taught. A discussion of the online development paradigm for robots as well as a comparison with traditional approaches is available in [39] [50] [40]. The arrival of early online developmental robots and current knowledge about their associated properties seem to encourage various related research communities to pursue further research in this direction. As recently demonstrated in [52], the architecture with internal actions is able to perform "logical reasoning" without using a logic internal representation.

CALL OUT: *The mental capability of a developmental robot (as well as a human being) in its muddy living environment depends on five factors: (1) its sensors, (2) its effectors, (3) its computational resources, (4) its developmental program, and (5) how the robot is taught.*

ACKNOWLEDGEMENTS

The authors would like to thank Changjiang Yang for writing a preprocessing program for the touch sensors of the SAIL robot, and Rebecca Smith and Matthew Ebrom for their assistance in conducting experiments. The work reported here was supported in part by The National Science Foundation under grant No. IIS 9815191, DARPA ETO under contract No. DAAN02-98-C-4025, DARPA ITO under grant No. DABT63-99-1-0014 and research gifts from Microsoft Research, Siemens Corporate Research and Zyvex.

REFERENCES

[1] N. Almassy, G. M. Edelman, and O. Sporns. Behavioral constraints in the development of neural properties: A cortical model embedded in a real-world device. *Cerebral Cortex*, 8(4):346–361, 1998.

[2] J. R. Anderson. *Rules of the Mind*. Lawrence Erlbaum, Mahwah, New Jersey, 1993.

[3] R. C. Arkin. *Behavior-Based Robotics*. MIT Press, Cambridge, Massachusetts, 1998.

[4] M. Bichsel. *Strategies of Robust Object Recognition for the Automatic Identification of Human Faces*. Swiss Federal Institute of Technology, Zurich, Switzerland, 1991.

[5] L. Birnbaum, M. Brand, and P. Cooper. Looking for trouble: using causal semantics to direct focus of attention. In *Proc. 4th Int'l Conf. on Computer Vision*, pages 49–56, 1993.

[6] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, March 1986.

[7] R. A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47:139–160, 1991.

[8] S. Chen and J. Weng. State-based SHOSLIF for indoor visual navigation. *IEEE Trans. Neural Networks*, 11(6):1300–1314, Nov. 2000.

[9] M. Cole and S. R. Cole. *The Development of Children*. Freeman, New York, 3rd edition, 1996.

[10] D. R. Cox. Statistical analysis of time series: Some recent developments. *Scand. J. Statist.*, 8(2):93–115, 1981.

[11] G. L. Drescher. *Made-Up Minds*. MIT Press, Cambridge Massachusetts, 1991.

[12] S. E. Fahlman and C. Lebiere. The cascade-correlation learning architecture. Technical Report CMU-CS-90-100, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, Feb. 1990.

[13] J. H. Flavell, P. H. Miller, and S. A. Miller. *Cognitive Development*. Prentice Hall, New Jersey, 3rd edition, 1993.

[14] H. Gardner. *Multiple intelligences: The theory in practice*. Basic Books, New York, 1993.

[15] S. Grossberg. Adaptive pattern classification and universal recoding: I. parallel development and coding of neural feature detectors. *Biological Cybernetics*, 23:121–131, 1976.

[16] J.D. Han, S.Q. Zeng, K.Y. Tham, M. Badgero, and J.Y. Weng. Dav: A humanoid robot platform for autonomous mental development. In *Proc. IEEE 2nd International Conference on Development and Learning (ICDL 2002)*, pages 73–81, MIT, Cambridge, Massachusetts, June 12-15 2002.

[17] X. Huang and J. Weng. Novelty and reinforcement learning in the value system of developmental robots. In *Proc. Second International Workshop on Epigenetic Robotics (EPIROB'02)*, pages 47–55, Edinburgh, Scotland, August 10 - 11 2002.

[18] X. Huang and J. Weng. Value system development for a robot. In *Proc. 2004 International Joint Conference on Neural Networks*, July 26 - 29 2004.

[19] W. S. Hwang and J. Weng. Hierarchical discriminant regression. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(11):1277–1293, 2000.

[20] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[21] M. Kirby and L. Sirovich. Application of the karhunen-loève procedure for the characterization of human faces. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 12(1):103–108, Jan. 1990.

[22] S. Koenig and R. G. Simmons. Unsupervised learning of probabilitic models for robot navigation. In *IEEE Int'l Conf. on Robotics and Automation*, pages 2301–2303, Minneapolis, Minnesota, 1996.

[23] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.

[24] J. E. Laird, A. Newell, and P. S. Rosenbloom. Soar: An architecture for general intelligence. *Artificial Intelligence*, 33:1–64, 1987.

[25] J. E. Laird, E. S. Yager, and C. M. Tuck. Robo-soar: An integration of external interaction, planning, and learning using Soar. *Robotics and Autonomous Systems*, 8:113–129, 1991.

[26] T. Martinetz and K. Schulten. A 'neural-gas' network learns topologies. *Artificial Neural Network*, 1:397–402, 1991.

[27] R. Miikkulainen. Script recognition with hierarchical feature maps. *Connection Science*, 2:83–101, 1990.

[28] M. L. Puterman. *Markov Decision Processes*. Wiley, New York, 1994.

[29] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of IEEE*, 77(2):257–286, 1989.

[30] R. S. Sutton and A. Barto. *Reinforcement Learning*. MIT Press, Cambridge, Massachusetts, 1998.

[31] S. Thrun. *Explanation-Based Neural Network Learning: A Lifelong Learning Approach*. Kluwer Academic, Boston, Massachusetts, 1996.

[32] D. S. Touretzky and L. M. Saksida. Operant conditioning in skinnerbots. *Adaptive Behaviors*, 5(3/4):219–247, 1997.

[33] A. M. Turing. On computable numbers with an application to the Entscheidungsproblem. *Proc. London Math. Soc., 2nd series*, 42:230–265, 1936. A correction, *ibid.*, 43, pp. 544-546.

[34] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

[35] P. F. Verschure, J. Wray, O. Sporns, G. Tononi, and G. M. Edelman. Multilevel analysis of classical conditioning in a real world artifact. *Robotics and Autonomous Systems*, 16(2-4):247–265, 1995.

[36] I. Wallance, D. Klahr, and K. Bluff. A self-modifying production system of cognitive development. In D. Klahr, P. Langley, and R. Neches, editors, *Production System Models of Learning and Development*, pages 359–435. MIT Press, Cambridge, Massachusetts, 1987.

[37] X. Wang, M. M. Merzenich, K. Sameshima, and W. M. Jenkins. Remodeling of hand representation in adult cortex determined by timing of tactile stimulation. *Nature*, 378(2):13–14, 1995.

[38] C. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.

[39] J. Weng. Learning in image analysis and beyond: Development. In C. W. Chen and Y. Q. Zhang, editors, *Visual Communication and Image*

Processing, pages 431 – 487. Marcel Dekker, New York, 1998. A revised version from "Living Machine Initiative," MSU CPS Tech. Report CPS-96-60, 1996.

[40] J. Weng. A theory for mentally developing robots. In *Proc. IEEE 2nd International Conf. on Development and Learning (ICDL 2002)*, pages 131–140, MIT, Cambridge, Massachusetts, June 12-15 2002.

[41] J. Weng. Developmental robotics: Theory and experiments. *International Journal of Humanoid Robotics*, 1(2):199–235, 2004.

[42] J. Weng. A theory of developmental architecture. In *Proc. 3rd International Conf. on Development and Learning (ICDL 2004)*, La Jolla, California, Oct. 20-22 2004.

[43] J. Weng. Muddy tasks and the necessity of autonomous mental development. In *Proc. 2005 AAAI Spring Symposium Series, Developmental Robotics Symposium*, Stanford University, March 21-23 2005.

[44] J. Weng. On developmental mental architectures. *Connection Science*, 2007. accepted and to appear.

[45] J. Weng, N. Ahuja, and T. S. Huang. Learning recognition and segmentation using the Cresceptron. *International Journal of Computer Vision*, 25(2):109–143, Nov. 1997.

[46] J. Weng and S. Chen. Vision-guided navigation using SHOSLIF. *Neural Networks*, 11:1511–1529, 1998.

[47] J. Weng and W. Hwang. Online image classification using IHDR. *International Journal on Document Analysis and Recognition*, 5(2-3):118–125, 2002.

[48] J. Weng, W. S. Hwang, Y. Zhang, and C. Evans. Developmental robots: Theory, method and experimental results. In *Proc. 2nd International Conf. on Humanoid Robots*, pages 57–64, Tokyo, Japan, Oct. 8-9 1999. IEEE Press.

[49] J. Weng, W. S. Hwang, Y. Zhang, C. Yang, and R. Smith. Developmental humanoids: Humanoids that develop skills automatically. In *Proc. First IEEE Conf. on Humanoid Robots*, MIT, Cambridge, Massachusetts, Sept. 7-8 2000. IEEE Press.

[50] J. Weng, J. McClelland, A. Pentland, O. Sporns, I. Stockman, M. Sur, and E. Thelen. Autonomous mental development by robots and animals. *Science*, 291(5504):599–600, 2001.

[51] J. Weng and I. Stockman. Autonomous mental development: Workshop on development and learning. *AI Magazine*, 23(2):95–98, 2002.

[52] Y. Zhang and J. Weng. Action chaining by a developmental robot with a value system. In *Proc. IEEE 2nd International Conf. on Development and Learning (ICDL 2002)*, pages 53–60, MIT, Cambridge, Massachusetts, June 12-15 2002.