The Developmental Approach to Artificial Intelligence: Concepts, Developmental Algorithms and Experimental Results

Juyang Weng, Colin H. Evans, Wey S. Hwang and Yong-Beom Lee

Department of Computer Science and Engineering
Michigan State University
East Lansing, MI 48824

Technical Report MSU-CPS-98-25, July, 1998
In Proc. NSF Design & Manufacturing Grantees Conference, Jan. 1999.

Keywords: AI architecture, intelligent agents, intelligent robots, human-machine interactions, multimodal integration, vision, speech, language, machine learning and cognitive development.

Abstract

This article introduces the developmental approach to artificial intelligence, which is different from other existing major approaches: knowledge-based, behavior-based, learning-based, and evolutionary approaches. The developmental approach is motivated by human cognitive development from infancy to adulthood, during which human individuals develop their intelligence through interactions with the environment. A developmental algorithm of a species, either natural or artificial, starts to run at the "birth" of the individual and it runs continuously through the entire life span. It automates the process of system development. The developmental approach does not mean just from small to big and from simple to complex. It requires the system to learn new tasks and new aspects of each complex task without a need of reprogramming. We introduce AA-learning as a basic mode for developmental learning. This paper introduces the basic concepts, the architecture, some developmental algorithms, and some experimental results from our SAIL project, which aims to built agents that are capable of performing developmental learning.

Figure 1: The SAIL robot under construction in a machine shop, awaiting its "birth." What can it learn after its birth? Can it recognize humans it has been introduced to? Can it identify gender? Can it learn to act according to what it perceives? For example, in this case, can it be taught to present the right present to guests?

1 Introduction

1.1 Integration and challenges

Existing research work on intelligent agents and intelligent robots have produced a rich collection of methods and systems. Some have demonstrated impressive capabilities. However, these studies tend to concentrate on narrowly defined scenarios that call for specialized capabilities.

A robot is an embodied agent [1]. We use the terms *agent* and *robot* interchangeably in this paper. The capability of a robot to perform complex, generalized tasks depends very much on its cognitive and behavioral capabilities. For example, if a robot has only a range sensing capability, it cannot navigate along outdoor walkways of a campus, since range sensors cannot distinguish between road and non-road surfaces such as concrete blocks, asphalt, grass, and soil. In an indoor environment, it is difficult for such a robot to identify a closed door that it must open and pass through. A robot with visual and auditory capabilities will be able to handle more general scenarios and may interact with its environment, including humans, using the visual and auditory modalities. Tasks that require manipulatory behaviors, such as pouring a cup of water into another, also require sophisticated coordination between sensors and effectors. Additionally, interaction between robots and humans is desirable and may be necessary to perform some tasks.

Currently, we are developing a robot system called SAIL at Michigan State University. As a research objective, it will learn to navigate through various settings and interact with humans using vision, speech and its arm. For example, we will teach it how to greet guests, and learn people's names and genders (see Fig. 1). Based on the current task-specific paradigm for system development, these tasks are not tractable, unless environmental conditions are well controlled and the task is drastically simplified and exactly specified.

[1] An agent is something that perceives and acts [51].

1.2 The current task-specific paradigm

The current paradigm for developing an intelligent system can be characterized by the following steps.

1. Start with a given task.
2. A human being tries his (or her) best to analyze the task.
3. The human derives a task space representation, which may depend on the tool chosen.
4. The human chooses a computational tool and maps the task space representation to the tool.
5. The parameters of the tool are determined using one or a combination of the following methods: (a) The parameters are manually specified using hand-crafted domain knowledge (e.g., the knowledge-based methods in CYC [33] [34] and WorldNet [41]). (b) They are decomposed manually into system behavior modules (e.g., behavior-based methods in the subsumption architecture [7] and active vision [1]). (c) They are estimated using a training procedure (e.g., learning-based methods in Q-learning [60], eigenfaces [59] and SHOSLIF [66]). (d) They are searched for based on a task-specific objective function (e.g., genetic or artificial life methods in Animate [67], SAGA [23], and AutonoMouse [16]).

A combination of various methods is typical in step 5 (e.g., AutonoMouse [16] used both learning and a genetic algorithm). In a typical research endeavor, steps 2 through 5 may be repeated multiple times, which might lead to a modification of the given task in step 1. We call this the task-specific paradigm since the paradigm starts with the task and all the rest of the steps depend on the task.

As an example, for an autonomous navigation task which ends up using Q-learning [60], a researcher may go through these steps: (1) Define the navigation task (e.g., in a given environment, from point A to point B, etc.). (2) Analyze the task (e.g., if collision avoidance is required, etc.). (3) Specify the classes of sensory observation, states for all locations, and legal actions at each location. (4) Adopt the Q-learning algorithm [60] and map the task-specific observations, states and legal actions to the Q-learning notation. (5) Create a simulation environment to train the system using the Q-learning algorithm with the goal to converge to an acceptable policy (such as appropriate navigation rules for each location). Since reinforcement learning using the Q-learning algorithm requires many iterations, typically a simulation environment is used to reduce the time and cost of conducting iterations by a physical robot. For an excellent survey of recent work on reinforcement learning, see a report on the NSF Workshop on Reinforcement Learning [39] and a survey article by Kaelbling, Littman & Moore [27].

This task-specific paradigm has produced impressive results for those tasks whose space is relatively small and relatively clean and exact, such as computer chess. However, it faces tremendous difficulties for tasks whose space is huge, vague, difficult to fully understand and difficult to model adequately by hand, such as vision-based recognition of general objects, vision-based autonomous navigation by a mobile robot in natural unknown indoor and outdoor environments, human-computer interaction via vision, speech and gesture, and human-computer discourse via spoken or written language.

Furthermore, the integration of multiple modalities and behaviors appears to be more difficult than the development of each individual sensing modality or behavior. Suppose that system integration is attempted through integration of several pre-developed subsystems, such as vision subsystems, speech subsystems, collision avoidance subsystems, etc. Each subsystem of a robot has been developed individually with its own environmental restrictions. The environment that satisfies all these subsystems may be very restrictive and hard to come by. Generalizations from such a restrictive environment may require a fundamental change in overall integration strategy and in the involved subsystems.

1.3 Human cognitive development

How does each human individual develop his or her capabilities? A very rich collection of studies has been conducted on human cognitive development. Jean Piaget [21] [10] [9], proposed to divide human cognitive development into four major stages, as summarized in Table 1. These stages may have a lot to do with neural

Table 1: Stages in Human Cognitive Development Proposed by Piaget

Stage	Rough ages	Characteristics
Sensorimotor	Birth to age 2	Not capable of symbolic representation
Preoperational	Age 2 to 6	Egocentric, unable to distinguish appearance from reality; incapable of certain types of logical inference
Concrete operational	Age 6 to 12	Capable of the logic of classification and linear ordering
Formal operational	Age 12 & beyond	Capable of formal, deductive, logic reasoning

development in the brain [31]. Furthermore, more recent studies have demonstrated that the progress into each stage depends very much on the learning experience of each individual and thus biological age is not an absolute measure for cognitive stages. For example, Bryant and Trabasso [8] showed that given enough drill with the premises, 3- and 4-year old children could do some tasks to construct linear orderings, a deviation from the classical Piagetian stage partition. It is known that some basic reflexes are present at birth, such as sucking and eye-blinking. It is also known that the development of cognitive capabilities in each human individual requires many years of learning during which he or she interacts with the environments and learns to perform more and more complex tasks.

1.4 Rethink the paradigm

The number of tasks that a robot can potentially perform is astronomical. Each task may require a different representation, a large amount of domain knowledge, a long list of behaviors and even more complex list of interactions among these behaviors. Thus, at the task level, we have the high complexity of a *society of mind* as described by Marvin Minsky [42]. This is the story of "what" — what in a mature brain.

However, what about the story of "how"? *How* can an infant brain, guided by a developmental program that is genetically pre-determined before birth[2] continuously develop and learn to perform an open number of tasks? It seems necessary to rethink the task-specific paradigm that we are so used to. This brings us to what is called the developmental approach.

2 The Developmental Approach

The major goal of the proposed developmental approach is to study the mechanism, predetermined at "birth," that enables an agent to develop cognitively through interactions with the environment.

There are some existing works that deal with growing a network from small to large and from simple to complex. In the works by Carpenter *et al.* 1991 [11], McKusick & Langley 1991 [40], and Wen *et al.* 1992 [61], a network is constructed automatically from a given set of feature vectors. In the work of Cresceptron by Weng *et al.* 1992 [64] [65], a network is incrementally grown whose structure is a function of automatically detected and memorized novel edge groupings and their hierarchy in incrementally fed images. Elman 1993 [17] argued about the advantages for a neural network to start small. Thrun 1995 [58] proposed lifelong learning from a feature space for classification tasks. SHOSLIF by Weng & Chen 1996 [66] incrementally grows a regression tree using principle component analysis and discriminant analysis for automatic optimal feature derivation at each node. However, the concept of the developmental approach fundamentally goes beyond the notions of small to large and from simple to complex.

Upon comparing human learning with the task-specific paradigm, we see a stark contrast: A human child is a general purpose-learner. He or she can learn new tasks in new domains without the need for a human teacher to reprogram the brain. A parent can teach a child to do new things without knowing the exact

[2]In biological organisms, interactions with the environment start from the conception.

Table 2: Comparison of Approaches

Approach	Species architecture	World knowledge	System behavior
Knowledge-based	programming	manual modeling	manual modeling
Behavior-based	programming	avoid modeling	manual modeling
Learning-based	programming	treatment varies	special-purpose learning
Evolutionary	genetic search	treatment varies	genetic search
Developmental	programming	avoid modeling	general-purpose learning

neural representation in the child's brain. In this way, the children's brain is closed to the human teacher. However, the task-specific paradigm requires a human being to go through the above steps 1 through 5 for every significantly different task.

Thus, the true developmental approach requires a drastic departure from the current task-specific paradigm. It requires the following:

1. Domain-extensibility: The machine agent must be able to learn new tasks in new domains without need for reprogramming.

2. Relieving the programmer from task analysis: It does not require the programmer to analyze every task that the agent must learn.

3. Freedom from a hand-crafted task model: It does not require the programmer to give a task space representation and to establish the mapping from the task space to the chosen computational tool.

4. A developmental mechanism: It requires a developmental learning mechanism implemented by a developmental algorithm.

5. A "living" machine for cognitive development: After its "birth", the developmental learning algorithm runs daily, enabling the machine to learn continuously through interactions with the environment using its sensors and effectors.

Table 2 outlines the major characteristics of several approaches for developing intelligent systems. From Table 2, we can see that the developmental approach stands on the middle ground between two extremes: at one extreme, the agent is totally hand-coded by human beings (the knowledge-based approach) and at the other extreme, the agent is constructed using genetic search (the evolutionary approach)[3]. The former extreme requires a large amount of human domain knowledge and thus is the most domain specific and *ad hoc* approach. The latter extreme requires the least amount of human knowledge but faces a tremendous cost in time and computation. The developmental approach relieves humans from explicit design of (a) any task-specific representation and knowledge and (b) system behavior representation, behavior modules and their interactions. However, the developmental algorithm supplied at "birth" must be designed by human beings.

[3]It is worth noting that the evolutionary approach is motivated by species evolution in nature through which the species architecture (including the learning mechanism) is the result of very long term genetic search. Thus, in principle, the evolutionary approach does not have to require a task-specific representation. However, due to the complexity considerations, the current genetic algorithms have been used with a human designed, task-specific chromosome representation and only a particular part of the evolving system is searched for genetically (e.g., Animate [67], AutonoMouse [16] and the work by Steels [53]).

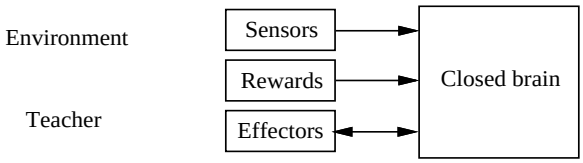


Figure 2: An AA-learning agent has three types of channels to interact with the environment, sensors, effectors and reward receivers. The reward receivers can be modeled by biased sensors. The double arrow for the effectors means that the actions imposed by the environment (e.g., human) can be sensed by the "brain."

3 AA-learning

A machine agent M may have several sensors. At the time of "birth", its sensors fall into one of the two categories, biased and unbiased[4]. If the agent has a predefined (innate) preference for the signal from a sensor, this sensor is then called biased. Otherwise, it is an unbiased sensor, although a preference can be developed by the agent later through learning. For example, a human being has an innate preference to sweet and bitter tastes from the taste sensor, but does not have a strong preference to visual images of various furniture items. By definition, the *extroceptive*, *proprioceptive* and *interoceptive* sensors are, respectively, those that sense stimuli from external environment (e.g., visual), relative position of internal control (e.g., arm position), and internal events (e.g., internal clock).

Next, we introduce a computational definition of AA-learning (named after *a*utomated, *a*nimal-like learning *without claiming to be complete*) for a machine agent.

Definition 1 *A machine agent M conducts AA-learning at discrete time instances if after its "birth" the following conditions are met for all the time instances $t = 0, 1, 2, ...$ (I) M has a number of sensors (biased or unbiased, extroceptive, proprioceptive, or interoceptive), whose signal at time t is collectively denoted by $x(t)$. (II) M has a number of effectors, whose control signal at time t is collectively denoted by $a(t)$. The effectors include extro-effectors (those acting on the external world) and intero-effectors (those acting on internal mechanism, e.g., attention). (III) M has a "brain" denoted by $b(t)$ at time t. (IV) At each time t, the time-varying state-update function f_t updates the "brain" based on sensory input $x(t)$ and the current "brain" $b(t)$:*

$$b(t+1) = f_t(b(t), x(t)) \tag{1}$$

and the action-generation function g_t generates the effector control signal based on the updated "brain" $b(t+1)$:

$$a(t+1) = g_t(b(t+1)) \tag{2}$$

where $a(t+1)$ can be a part of the next sensory input $x(t+1)$. (V) The "brain" of M is closed in that after the birth (the first operation), $b(t)$ cannot be altered directly by human teachers for teaching purposes. It can only be updated according to Eq. (1).

Fig. 2 illustrates the relationship between an AA-learning agent and the world. The design for a "brain" representation $b(t)$, the time-varying state-update function f_t, and the action-generation function g_t determines the AA-learning mechanism as well as the maturation schedule. It is worth noting that t can be dropped from f_t and g_t in the definition since $b(t)$ is not restricted in the definition. For example, any time varying function $f_t(b(t), x(t))$ can be represented by a time invariant function $f(x(t), b(t), t)$ and $(b(t), t)$ can be defined as the "brain." The definition for continuous time is analogous.

From the definition we can see that AA-learning does not require two separate learning and performance phases. The machine agent learns while performing. This is important for continuous, open-ended cognitive development. AA-learning does not require humans to provide edited and segmented sensory input (i.e., no need to spoon-feed data). The system accepts a continuous, unsegmented sensory input stream on-line.

[4]This is an engineering definition. For a biological organism, it is hard to say that any of its sensors is absolutely unbiased.

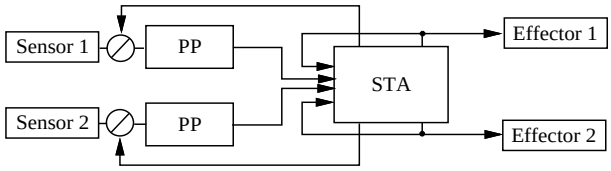


Figure 3: A schematic illustration of the coarse architecture of the proposed learning mechanism. A circle represents an attention selector. It is also an effector. PP: preprocessor. STA: spatiotemporal associator.

The design of the AA-learning mechanism can take into account various phenomena known about animal learning and human learning. Therefore, this seems to be new ground where artificial intelligence and biological intelligence can converge.

4 Concepts

From the above definition, we can expect that the algorithm design of AA-learning is very challenging. As the first step toward AA-learning, we must introduce simplifications in order to make the stagewise research scope manageable, with measurable advances. These simplifications will inevitably affect the capabilities of the proposed system in various ways. A simplified model may serve as the first step towards future AA-learning systems that are more sophisticated, richer, and thus more powerful. On the other hand, it is naive to expect that an engineering system can or should mirror a biological system faithfully.

4.1 Innateness

In principle, our definition of AA-learning allows a rich array of characteristics that are present in biological organisms, including innate tendency for certain behavior and emotions. However, what does it mean to be innate?

Based on recent results in psychology and neurosciences, more and more researchers realize that we need to rethink innateness. The resent book "Rethinking Innateness" [18] (written by J. L. Elman, E. A. bates, M. H. Johnson, A. Karmiloff-Smith, D. Parisi and K. Plunkett) is a good example of the new thinking. The authors propose that innateness corresponds to constraints that operate at three possible levels: *representation, architectures, and timing (of development)*. *Representational* constraints have the most specific and direct relationship to knowledge. *Architectural* constraints operate at a more general level with less directly obvious relationship to resulting knowledge. *Timing* constraints are typically the most opaque (in humans) with regard to outcome. However the authors explain that representational innateness (so defined) is relatively rare in higher organisms. Mouth sucking tendency of a human new born is an example. In biological organisms, such an innate tendency (realized by e.g., starting weights in synapses) facilitates acquisition of certain basic behaviors that are essential for survival right after the birth, before any learning can take place. However, the innateness in humans that accounts for human intelligence operates mainly at the levels of architectures and timing (of development).

The developmental algorithm we introduce here concentrates on the architectures and timing levels, which is a lot more manageable than the representation level. Basic behaviors, including the small portion that is facilitated by innate tendency in higher organisms, can be learned by artificial systems. This design decision is expected to simplify the developmental algorithm to be designed. To be highly focused, we will not address the issue of emotion in this paper.

4.2 Architecture outline of the proposed AA-learning mechanism

Fig. 3 gives a schematic illustration of the coarse architecture of the proposed AA-learning mechanism SAIL (named after Self-organizing Autonomous Incremental Learner). The preprocessor in Fig. 3 performs some

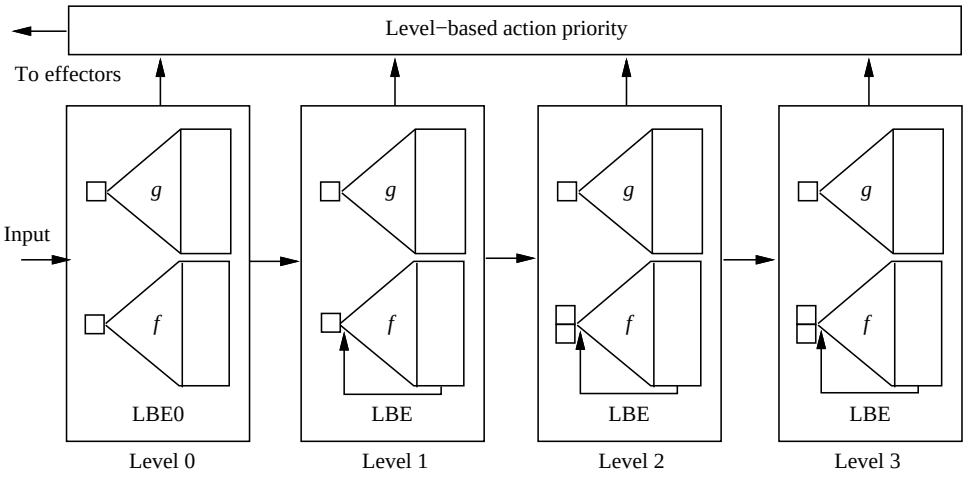


Figure 4: Levels that are built automatically in the STA. Each level corresponds to a level-building element (LBE).

sensor-specific transformation, such as intensity normalization, automatic gain (contrast) control, filtering, etc. The filtering may also include motion detectors. The attention selector in Fig. 3 is an intero-effector, which selects, e.g., a subpart of the image frame for later processing. The spatial temporal associator (STA) is the "brain" of the system. As shown in the figure, the input to the STA includes not just information from the sensors, but also the current control signal of the effectors.

4.3 Levels

As a basic point of developmental approach summarized in Table 2, we do not define architecture levels in terms of either domain knowledge hierarchy or system behavior hierarchy, as is common in existing works (e.g., see a recent survey article about machine learning by Langley [32]). Instead, our proposed level-based architecture corresponds to temporal context.

In the AA-learning mechanism, the global state s of the "brain" $b(t)$ at any time t is represented distributedly by states at different levels: $s = (s_0, s_1, s_2, ..., s_L)$, where s_i, $i = 0, 1, 2, ..., L$, represents the state at level i. The current number of levels is determined automatically based on the maturation schedule of the system which depends on the experience as well as the virtual age [5] of the system. Level 0 is context free, to model S-R (stimulus-response) reflex. Starting from level 1, temporal context is incorporated. The higher the level i, the more temporal context each state at level i represents. Fig. 4 illustrates the level building in STA. The basic mechanism of the level-building elements for each level is basically the same. The differences between levels will be explained later. From Fig. 4, one may immediately see the similarity between this scheme and the level arrangement with Rodney Brooks' well-known subsumption architecture [7, 6]. The major differences are: (1) Levels in SAIL are not defined in the sense of behavior as in the subsumption architecture, but rather in the extent of temporal context that is recorded. Each level in SAIL can incorporate many behaviors as long as each behavior has a similar amount of temporal context. (2) Mediation among many behaviors both within each level and among different levels are automatically learned in SAIL, instead of being programmed in. Such a mediation is extremely difficult to hand craft and program when the number of behaviors is large. In the following discussion, we will concentrate on a single level, and later we will discuss the issues of integrating different levels.

[5] The virtual age is the time of operation since the birth of the system.

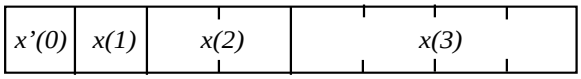


Figure 5: The state representation at $t = 3$ for 1-D signal with 2:1 uniform resampling in space, from sensory inputs $x(0)$, $x(1)$, $x(2)$, ... For this illustration, $\mathcal{S}$ is an 8-dimensional space. $x'(0)$ is the average of $x(0)$ and the initial zeros in the state vector.

4.4 States

In behavior-based learning approaches, the states of an agent are manually bound to a set of predefined task concepts before training (e.g., see an excellent survey article [27] on state-based reinforcement learning methods). An AA-learning algorithm must automatically generate states without being given any task.

Let us first consider level 1 in Fig. 4. The part of the "brain" state at this level is denoted by a state vector $s(t)$. If $s(t)$ is considered a random process, Eqs. (1) and (2) are closely related to the formulations for Markov decision processes (MDP) [46], or HMMs (hidden Markov models) if the action part is omitted [48] [25]. Indeed, the state transition function f and the decision function g can be based on probability distributions shown below to take into account the uncertainty in states, observations and actions:

$$\mathrm{P}(s(t+1) = s' \mid x(t), s(t) = s)$$

and

$$\mathrm{P}(a(t+1) = a \mid s(t+1) = s')$$

where $\mathrm{P}(\cdot)$ denotes the probability. However, the states in MDPs have been typically defined as a set of symbols and there is no distance metric defined to measure the similarity between any two symbols (see, however, various MDP generalization techniques surveyed by Kaelbling, Littman & Moore [27]).

We define a state s to be a vector in a high dimensional space $\mathcal{S}$. Thus, our state has an explicit representation. $\mathcal{S}$ must contain all the possible sensory input $x \in \mathcal{X}$. In contrast to existing MDP methods, we require that the state records temporal context. Thus, we define the state space at level 1 to be $\mathcal{S} = \mathcal{X} \times \mathrm{R}(\mathcal{S})$, where $\times$ denotes Cartesian product and $\mathrm{R}(\cdot)$ denotes a re-sampling operator.

The design of the re-sampling operator need to take into account (a) the nature of the signal and (b) the desired temporal span in the state vector. The re-sampling can be performed in space S, in time, or in space-time. We discuss space-resampling in the following. Consider a case of a one-dimensional signal with a 2:1 decimation resampling. The Cepstrum vector of speech input at each time frame is a good example of a 1-D signal. For $s \in \mathcal{S}$, $\mathrm{R}(s)$ reduces the resolution of s by merging immediate neighboring components in s. If $s = (s_1, s_2, ..., s_{2m})$ represents the subsampled digital representation of a 1-D (univariate) function $f(x)$ on the real x-axis, $\mathrm{R}(s) = (s'_1, s'_2, ..., s'_m)$ can be computed by merging s_{2i-1} with s_{2i}: $s'_i = (s_{2i-1} + s_{2i})/2$. This corresponds to reduce the subsampling rate of the state space by a factor of two. If the input is an image, the resampling should take into account the 2-D nature of the input. A non-integer reduction ratio may also be necessary. The resampling operator does not have to be uniform for every component in s either. In summary, the resampling operator should not change the topology of the input data in $\mathcal{S}$ unless the topology cannot be represented by the resulting resolution. As can be seen, the faster the resolution of state s is reduced by the resampling operator $\mathrm{R}(s)$, the faster the history is sunk out of the state s.

Thus, the state transition function in Eq. (1) represents a simplified mapping $f : \mathcal{X} \times \mathrm{R}(\mathcal{S}) \mapsto \mathcal{S}$ at level 1. First, since the state space cannot be manually designed, we let f map $(x(t), \mathrm{R}(s(t)))$ directly to itself:

$$s(t+1) = (\mathrm{R}(s(t), x(t)). \tag{3}$$

In other words, the next state $s(t+1)$ keeps all the information of sensory input $x(t)$ and the re-sampled version of the current state $s(t)$. Given sensory inputs $x(0), x(1), ...$, this simplified f defines a trajectory of states $s(1) = (0, x(0))$, $s(2) = (\mathrm{R}(s(1), x(1))$ and so on. Fig. 5 gives an illustration of a state at time $t = 3$, which uses 1-D, 2:1 uniform resampling. The 2:1 resampling rate reduces the resolution by a factor

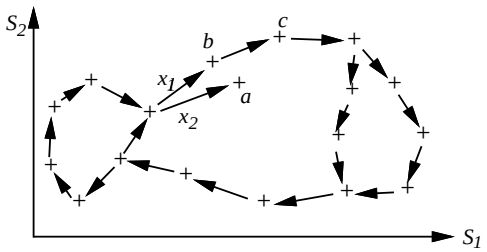


Figure 6: This representation of states enables generalization across states. Here, the high dimensional state space $\mathcal{S}$ is illustrated by a 2-D Euclidean space only. a is a newly generated state, which does not have any outgoing transition experience. The existing state b is the nearest neighbor of a in the state space $\mathcal{S}$. The transition path, from b to c, that is learned by b can be used for predicting the next state and the action choice for b, $g(b)$, can be used as the next action for a.

of 2 through time. Thus, if the dimensionality of x is d, roughly $\log_2(d)$ frames are kept in the state s. The earlier that a frame $x(t)$ is in a state s, the lower its resolution. If $x(t)$ of a particular sensing modality has low dimensionality but a longer history is necessary in the state representation, a slower resolution reduction rate is necessary. For example, 3:2 or 5:3 ratios can be used.

Since the input $x(t)$ is what is selected by attention action, such a state representation enables the system to take into account a time-discounted context. A more sophisticated transition function f will be discussed in the later sections. In either case, after the power of a machine agent M is turned on for a day, the states $s(0)$, $s(1)$, $s(2)$, ... result in an ever extending trajectory in the space $\mathcal{S}$ as shown in Fig. 6, until the power is turned off. In principle, the "brain" can memorize all the states $s(t)$ that have occurred and f records all of the state transition. In the later sections, we will discuss state clustering to control the number of states in memory.

A vector space with a defined norm (distance metric) is called a normed space [37] [50]. With a normed $\mathcal{S}$, the system can generalize across the states, as illustrated in Fig. 6. The predicted states and the action from a newly generated state a can be determined from those of the nearest neighbor state b in $\mathcal{S}$. This state representation also facilitates the following important functionalities: (1) States can be generated online as they are being recorded. (2) The distance metric in $\mathcal{S}$ makes it possible to access a huge number of states using a tree-based function approximator for real-time operation. (3) State clustering and forgetting can be naturally applied.

4.5 Learning types

Eqs. (1) and (2) identify four components of the AA-learning agent for each time instance t:

$$(a(t+1), s(t+1), s(t), x(t)). \tag{4}$$

They involve three entities: action, state, and sensor.

Depending on whether the action is imposed or not, the learning can be classified into *action-imposed* learning and *action-autonomous* learning. *Action-imposed* learning is such that the extro-effector part of $a(t+1)$ is supplied by the trainer. For example, hand-in-hand learning can be used by human adult to teach a child how to use a pen. Otherwise, the learning is *action-autonomous* learning.

Depending on whether the state $s(t)$ is imposed or not, learning can be classified into *state-imposed* and *state-autonomous*. The state-imposed learning is such that $s(t)$ and $s(t+1)$ are set by the human trainer during the learning. If a learning method requires a task-specific representation, the representation typically determines the meaning of states and thus the learning must use state-imposed learning. AA-learning is state-autonomous learning. As explained earlier, with AA-learning, the state of the system is determined by the developmental algorithm autonomously. Another concept is the state-readability. If the state of the system is not directly readable to the teacher, the learning is state-readable. Otherwise, it is state-unreadable.

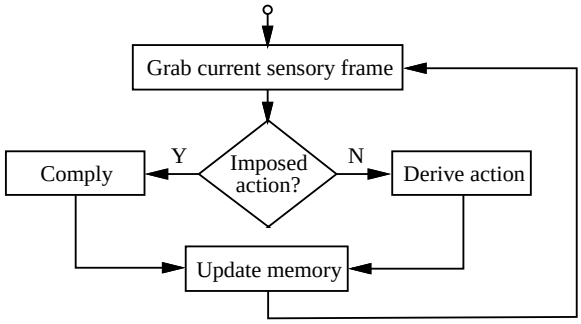


Figure 7: The flowchart for AA-learning. The system learns while performing.

Depending on whether the unbiased sensor is used or not, the learning can be classified into reinforcement learning and communicative learning. *Reinforcement* learning is such that a *biased* sensor is used to reinforce or punish certain response from the machine agent. *Communicative* learning is such that only *unbiased* sensors are used in learning. This requires that the agent to correctly interpret the signal from unbiased sensors, either it is an instruction for action, an encouragement, an explanation, etc. Learning by a human adult is mostly conducted in the communicative learning mode.

The learning type can be represented by a 3-tuple (A, S, X) where $A \in \{i, a\}$ denotes if action is imposed or autonomous, $S \in \{i, a\}$ denotes the state is imposed or autonomous, and $X \in \{r, c\}$ denotes the biased sensor is used or not. There are 8 different 3-tuples, representing a total of 8 different learning types. AA-learning is state-autonomous learning. Thus, there are 4 types of AA-learning: Type (1) action-imposed and reinforcement, Type (2) action-imposed and communicative, Type (3) action-autonomous and reinforcement, and Type (4) action-autonomous and communicative. It is worth noting that these four types are typically interleaved in a natural learning environment for animals and humans. These definitions are required beyond the coarse classic definition of supervised and unsupervised learning[6]. Fig. 7 illustrates a flow chart of AA-learning. If the trainer imposes an action on an effector at any time through, e.g., through a joystick, the system performs action-imposed learning for that effector. Otherwise, the system performs action-autonomous learning, during which reinforcement learning or communicative learning are used.

4.6 Simple action-imposed learning

To start training using our proposed AA-learning mechanism, most of the learning activities will be action-imposed with simultaneous positive reinforcement signals applied to the biased sensor [7] to allow the agent to learn some basic behaviors that are probably innate in biological organisms.

To aid in understanding, we describe an oversimplified and thus very inefficient and weak version of action-imposed learning. Suppose that the machine agent M has recorded in memory $B = \{(x(i), s(i), a(i)) \mid i = 0, 1, ..., t-1\} \cup \{s(t), a(t)\}$. Note that $s(t), a(t)$ are the result from sensory input $x(t-1)$. According to the flow diagram in Fig. 7, M grabs the current sensory frame $x(t)$. Then, M computes the next state $s(t+1)$, e.g., using Eq. (3). If an action is imposed, $a(t+1)$ is supplied by a human being (or environment) and thus M complies by sending $a(t+1)$ to the effector and then updates its memory by replacing B by $B \cup \{x(t), s(t+1), a(t+1)\}$. If an action is not imposed, M derives action $a(t+1)$ based on the past experience using a simplified g in Eq. (2) as follows. First, M finds the best matched state:

$$j = \text{argmin}_{0 \le i \le t} \|s(t+1) - s(i)\|. \tag{5}$$

[6]From the most strict definition of unsupervised learning, all the above learning modes are supervised by human to some degree. It is difficult to identify any type of learning that is completely unsupervised.

[7]Assume here that the reinforced action is desirable. Otherwise, a negative reinforcement (punishment) should be applied to the biased sensor, meaning "do not do this next time".

Then, the output action is determined as the action associated with the best matched $s(j)$: $a(t+1) = a(j)$. The memory update is done as before. After $x(t+1)$ is grabbed in the next machine cycle, which may include the resulting reward sensed by the biased sensor, the system memory becomes $B = \{(x(i), s(i), a(i)) \mid i = 0, 1, ..., t+1\}$.

As can be seen, this oversimplified version of AA-learning can do only a little generalization by extending the action learned by the nearest neighbor $s(j)$ (or multiple neighbors with action interpolation) to the current new state $s(t+1)$, whenever no action is imposed by the human.

4.7 Simple reinforcement learning

When no action is imposed, the learning is action-autonomous. The system generalizes using the nearest-neighbor rule. Such a generalization may or may not be good. Thus, a feedback signal in the range $[-1, 1]$ can be used a reward (positive or negative). When the agent has learned more and more basic behaviors through action-imposed learning, it can perform more and more action-autonomous learning.

During the action-autonomous learning, the reward should be delivered as needed and as frequent as possible to facilitate what is known as *behavior shaping* in animal learning [55] [15][8]. Lin 1992 and Dorigo & Colombetti 1994 have used the concept of shaping for teaching robots using reinforcement learning.

An oversimplified reinforcement learning method incorporated into the above action-imposed learning algorithm is as follows: Modify the step of finding the nearest neighbor in Eq. (5) so that only the states whose corresponding action has received non-negative rewards are searched for. The other parts are the same.

As we know, reward in reinforcement learning could be delayed. Thus, we face a well-known credit assignment problem: to what event is the reward or punishment received due ? Existing reinforcement learning methods have used a single time-predicted reward value for each state-action pair (e.g., expected time-discounted reward value $Q(s, a)$ in Q-learning [60] [27] or expected time-average reward value in R-learning [52] [38]). These methods have been extensively studied for problems that start with a predefined task [27]. However, such time-based optimality criteria suffers from fundamental limitations. First, the actions adopted can differ significantly according to different time models for rewards (e.g., finite horizon, infinite horizon, and average reward models [27]). Second, the model is task specific (e.g., $Q(s, a)$ has to be computed for each task for all the states s and possible actions a). Thus, the agent is not able to accept a new goal (e.g., told through an auditory command) and plan on its own according to its experience. Third, it may not correctly evaluate many conflicting goals. For example, daily short-term pleasure goals are often in conflict with a long term goal (i.e., painstaking study everyday for an advanced degree in the future). Fourth, credit assignment should be a learned behavior that changes according to the goal and situation, instead of a static hand-crafted rule.

As pointed out by many researchers using reinforcement learning, agents should be given reinforcement signals that are local in time whenever possible [27]. A correct context at the time of reward is very important for association of the effect of reward with the intended action. Even if a reinforcer is delayed, stimulus should be used to bring the agent into the correct context while the reinforcer is delivered. Consider the following example: Suppose that a task takes 5 steps. The agent made a mistake only at the 3rd step and the overall result is a failure. Instead of letting the agent try all the possible action combinations over the 5-step task, the human trainer should reinforce the actions associated with the correct actions in steps 1, 2, 4 and 5 by bringing the agent into the right context and delivering the reward (i.e., saying "For step 1, you did well"). Using our simplified version of the reinforcement learning algorithm, this will require the teacher to take the agent to the correct context $(s(t+1), a(t+1))$ and then give the reward which is used by the algorithm to replace the record associated with $(s(t+1), a(t+1))$.

[8] For example, suppose you want to teach a rat to stand up after hearing your instruction (e.g., an audio tone). You modify your reinforcement schedule according to how much the rat has learned. At first, food is given if the rat gets up on its hind legs any time, without tone. During this period, the audio tone can be introduced to teach the rat to respond after hearing the tone by rewarding food for only response after the tone. Later, food is given only if rat raises to a new height after the tone. Finally, you give it food only when the rate stands up on its rear legs immediately after the audio tone.

Table 3: A temporal sequence of communicative learning

Period		T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}
$x(t)$	See $v(t)$	B	P_2'	P_2	P_2	P_2	P_2	P_2	P_2	P_2	P_2'	B
	Hear $h(t)$	0	0	0	W	0	0	G	0	0	0	0
	Imposed action $a(t)$	0	0	0	0	P_2	0	0	F	0	0	0
	Reward $r(t)$	0	0	0	0	0	0	0	0	0	0	0

4.8 Simple communicative learning

In order to gradually develop the agent's capability of communicative learning, communicative instructions should be conveyed during each instance of either action-imposed learning or reinforcement learning. According to the mechanism of classical conditioning and higher order conditioning in animal learning, a communicative instruction (e.g., an audio tone) will become a conditional stimulus which elicits the conditional response (desired action).

Let's consider a simple example to see how this can be done using our oversimplified AA-learning algorithm explained in Sections 4.6 and 4.7. Suppose that we want to teach the machine agent M to do two tasks: Task 1: telling the name of a human individual from his or her face images, and Task 2: telling the gender from the individual from his or her face images.

Suppose that M has two unbiased sensors as shown in Fig. 3, one is a visual sensor (video camera) and the other is an auditory sensor (microphone). It also has a simulated biased sensor with sensory range $[-1, 1]$ from which we can deliver reward if we like. It has an effector (speaker). For simplicity, we can model each sensor as a frame grabber which gives a vector from the current input frame. At each time instant t the video camera coupled with an frame digitizer and a preprocessor gives a brightness-and-contrast-normalized video frame $v(t)$ which is a vector of pixels. The microphone with sound digitizer and preprocessor gives a Mel-Cepstrum vector $h(t)$ which characterizes the shape of the vocal tract at time t (A good tutorial about Mel-Cepstrum can be found in [47] [25]). Suppose that a different vector $o(t)$ sent to the speaker will give a different phone. The reward vector at time t is sensed as $r(t)$. Thus, for our simple system, the sensory input at time t is $x(t) = (v(t), h(t), o(t), r(t))$.

If M was born not long ago, it does not have a language. Thus, we cannot teach it to do the tasks without teaching it a language. The trainer thus has a simple language in mind for our simple system. He uses steadily voiced phones[9]. For simplicity, assume that each phone is represented by a single constant vector h_i although in reality the $h(t)$ vector is not perfectly constant over t during the utterance. Suppose that the trainer's family has 6 persons and he has designed a very simple language that has 10 phones: $\{h_i \mid i = 1, 2...., 10\}$. In his mind, the first 6 phones correspond to the name of the 6 persons and are denoted by $P_1, P_2, P_3, ..., P_6$, respectively. The remaining 4 phones denote "who?", "gender?", "male" and "female", respectively. After the power of the machine agent is turned on, the trainer lets each person enter the view of the camera so that the face fills the entire camera frame well, stay for a while, and then leave. During the presentation of each person, he asks about the name of the person by voicing the phone representing "who?" and then he immediately imposes an action on the machine agent by imposing the corresponding vector m_i, $i = 1, 2, ..., 6$, to the speaker effector. If he wants the agent to learn the gender of the person, he voices the phone representing "gender?" and then immediately imposes m_9 or m_{10} making M respond with the phone representing the correct gender. Therefore, this process uses action-imposed and communicative learning.

Table 3 shows the temporal transition of a session of such a learning mode, where B denotes background, P_i person i, P_i' partial view of person i when entering or exiting the view, W "who?", G "gender?", M "male" and F "female". Each period T_i denotes a period of about a few dozens of machine cycles. The teaching section is designed with the following in mind. The end of each period T_i is indicated by the drop

[9]Non-steady phones such as consonants and English words require multiple levels which are to be discussed later.

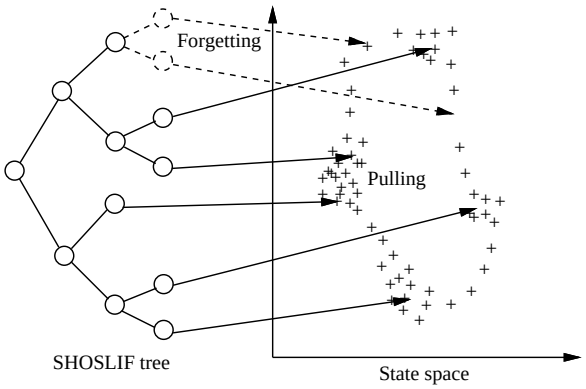


Figure 8: Vector quantization through pulling and merging in state space. Forgetting is used to control the size of the SHOSLIF-tree.

of the corresponding sensory input to zero (i.e., the offset of the sensory input). For example, the offset means the end of a question. Note that the state vector $s(t)$ records the past context of $x(t)$ to a certain extent up to the current frame $x(t)$. The action-imposed teaching sequence is such that an action is imposed after the offset of the corresponding sensory input that is designed to trigger the desired action. As soon as the offset of the corresponding sensory input appears in the state $s(t+1)$, the action vector $a(t+1)$ is produced. For example, consider period T_4. The last state at the period T_4 records the end of question "who" at the presence of person P_2 and this state $s(j)$ is associated with the action "replying P_2" as taught by the imposed action. Suppose that in a future testing session, P_2 enters again. With his face filling the camera view, the trainer asks "who?". At the end of question, the state vector $s(t)$ is used to find the best matched state $s(j)$ as described in Eq. (5). The associated action "replying P_2" is sent to the effector. A similar analysis is applicable to every state, including the reply "female" at the presence of P_2 at the offset of the question "gender?". Of course, with this simple single-level algorithm, the system is not able to smartly generalize the concept of gender beyond the basis of visual similarity.

5 Algorithm Outline

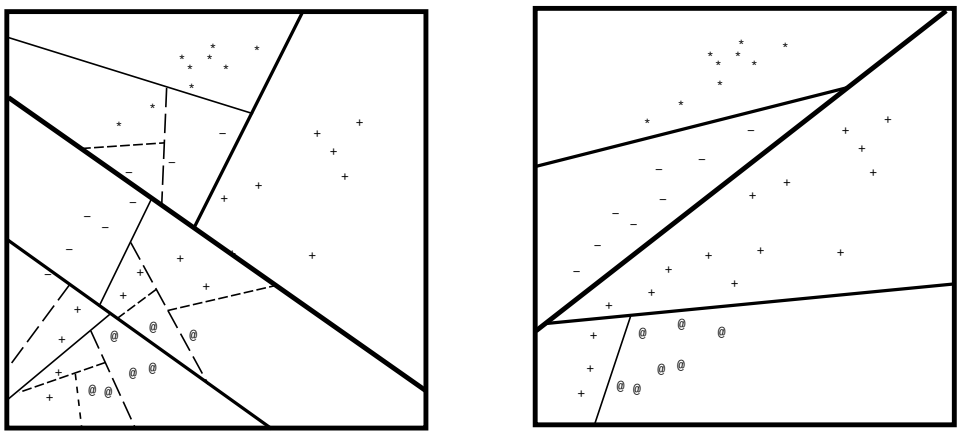
As discussed in Section 4.4, the state trajectory is directly observed from sensory input and the state definition in Eq. (3). However, there are two major problems with this simple-minded scheme. First, because memory is limited, it is not possible to memorize all of the states as defined in Eq. (3) for every time instance t. Second, recording all of the detailed states will make generalization (e.g., finding the best matched case) more difficult and slow.

5.1 State indexing using trees

Clearly, many state vectors that occur through time are very similar. We intend to store only state centers, each representing the centroid of a cluster of nearby state vectors. This is similar to the ideas of vector quantization (VQ) [29] [30] and clustering [26]. Fig. 8 explains the scheme. Since the number of centers can be very large, we use a regression tree to quickly find the top $k > 1$ nearest-matched centers, required in Eq. (5).

The SHOSLIF tree [62] [56] is basically a classification and regression tree (CART)[10] well studied in statistics [5] [22] and pattern recognition [12] [20]. It approximates a function f by mapping a high-dimensional input x to the corresponding output vector $y = f(x)$.

[10] A classification tree outputs class labels while a regression tree gives vector-valued outputs.



(a) (b)

Figure 9: (a) Hierarchical partition of the PCA binary tree. (b) Hierarchical partition of the PCA+LDA binary tree, which corresponds to a smaller tree. The symbols of the same type indicate the samples of the same class. The root of the tree represents the entire space. The thickest line partitions the space into two, represented by two child nodes of the root. Each node is partitioned again, and so on.

The major differences between the SHOSLIF tree and conventional regression trees are: (1) the SHOSLIF tree works for very high dimensional sensory input space directly. For example, for an image of 64×64 pixels, a $64 \times 64 = 4096$ dimensional input space in $x(t)$ is required, each dimension corresponding to a pixel[11]. (2) Unlike a conventional regression tree, which takes a number of features as input, the SHOSLIF tree must derive features on its own. It uses principle component analysis (PCA) to find the subspace in which the data vectors lie. Then, it uses Fisher's multi-class, multidimensional linear discriminant analysis LDA inside the PCA subspace to find the most discriminant subspace. Such a PCA+LDA analysis is performed in every internal node of the SHOSLIF tree. Thus, the tree cuts the class boundaries much better than a PCA only tree and results in a much smaller tree, as illustrated in Fig. 9. (3) We have developed an incremental way of constructing a SHOSLIF tree, which allows the tree to be built and trimmed incrementally. (4) The time to update the SHOSLIF tree with n leaf nodes (the number of centers) is $O(\log(n))$. This logarithmic time complexity is very important to AA-learning since it is meant to be real time.

More sophisticated dimension reduction techniques than a plain PCA can be used before the LDA. For example, the sliced inverse regression (SIR) and the principle Hessian directions (pHD) of Li (1991, 1992) [35, 36] can be adopted. Since the action vector uses continuous values in each component, in LDA the within class variance and the between class variance should be replaced by the expectation of the conditional variance and the variance of the conditional expectations, respectively.

Given a sequence of index vectors of the form $X = (\mathrm{R}(s(t)), x(t))$ (containing the sensory input and the context), a SHOSLIF tree is constructed incrementally through t. Each node has a center in X space representing the center of all the leaf nodes rooted from the node. At each time t, X is used to search for the top $k > 1$ nearest centers (represented by leaf nodes). The search starts from the root. During the search, at each level k nodes are further explored if there are that many. The children of these nodes are examined. Among them, the top k children nearest X are marked as active and are recursively explored further. This precess is carried on until all the nodes to be explored are leaf nodes. Each leaf node stores the centroid of the state clusters it represents. Thus, given X the top k matched states are found, instead of just the top one in the simplified version in Eq. (5).

[11] This way of direct treatment of an image as a high dimensional vector is now well known as the appearance-based approach in the computer vision community. Statistical methods have been used on this high dimensional space for a wide range of practical computer vision problems. This approach has achieved performances that are significantly better than feature-based methods (e.g., edge-based methods) for many difficult vision problems [28] [59] [43] [19] [57] [3].

An outline of the tree-traversal algorithm is given below. Its goal is to find the k leave nodes that are near X.

```
traverse-tree(k_nodes, X) {
    initialize candidates to empty;
    If (all nodes in k_nodes are leaves)
        process_k_leaves(k_nodes, X);
    else for (each node in k_nodes) do {
      if (node is a leaf)
         add node to candidates;
      for (each child of node)
         add child into candidates;
    }
    k_nodes = nearest_k_nodes(X, candidates);
    traverse-tree(k_nodes, X);
}
```

The tree is a binary tree for efficiency. The procedure `process_k_leaves` finds the nearest neighbor of X. If their distance is within the minimum distance required by a specified resolution, no new node is added by splitting the current leaf node. Otherwise, a new node is added that has X as the center. The remaining processing in procedure `process_k_leaves` is the state-pulling and merging discussed in the next section.

The distance metric used in selecting the top nearest k nodes is important. In SHOSLIF, we used the projections onto the discriminant vector as the distance measure [56]. When class-information is not available, Euclidean distance to the center of each node can be used. Each internal node in the tree maintains the statistics (e.g., mean) of all the nodes going into each of its left and right children, so that the node (e.g. center) can be updated. However, updating the center of an internal node requires a redistribution of all its children (unless we allow that some nodes are not retrievable) which is expensive. This node updating and children redistribution can be performed during regular robot sleeping periods.

5.2 State pulling and merging

For efficiency, we do not want to store all of the states that the robot has experienced. Instead, we want form state prototypes, each representing a larger number of similar states. Thus, we need to merge states that are nearby. However, we do not have access to all of the states at the same time, since the number of states is very large. The SHOSLIF-tree gives the top k nearest states. We use simulated gravity pulling on these k states. First, we compute the centroid of the k states. Each state is pulled toward the centroid, if it is within a specified *gravity limit*, according to the mass of the centroid and the mass of the state (whose mass is the number of states it has merged). All the states that are within the *merge distance* from the centroid after the pulling are merged as a one state (prototype).

5.3 Forgetting

Pulling and merging are mainly for state vectors that occur very often. The forgetting process takes care of state vectors that do not occur very often. Each node of the SHOSLIF tree has a memory residual register whose updating curve is shown in Fig. 10 which may resemble what we know about human memory characteristics [2] [24].

Whenever a tree node is visited, its trace is reset to 1 and then the trace curve declines using a slower speed. We define a series of memory *fade factors* $\alpha_1 < \alpha_2 < ... < \alpha_m \approx 1$. α_i is used for a node that has been visited i times, The memory trace r can be updated by $r \leftarrow r\alpha_i^t$ where t is the number of system cycles (refreshes) elapsed since the last visit to the node. When a node is visited, its memory trace is updated first from what remains from the last visit. If the memory trace falls below the designated threshold, the node should be deleted and so it is marked as to-be-deleted. If what is deleted is more than a single element (i.e.,

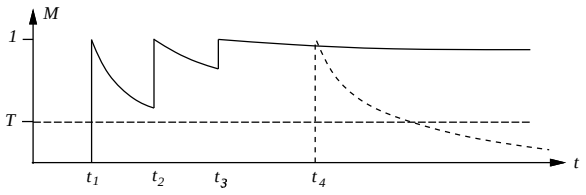


Figure 10: Update of memory trace M through time t. The solid curve represents a node which is visited often enough to be kept. The dashed curve indicates an element that is not visited often enough and thus, it falls below the threshold T before being visited again.

a subtree), the deleting process will not delete it right away to avoid consuming too much CPU time in a real time process. Instead, it puts the subtree in a garbage buffer which is to be cleaned when the learner is sleeping.

5.4 Three major mappings

At each level, the sensory space is denoted by $\mathcal{X}$, the state space by $\mathcal{S}$, the effector action space by $\mathcal{A}$, and the expected reward space by $\mathcal{R}$. The state transition function defines the mapping $f : \mathcal{X} \times \mathcal{S} \mapsto \mathcal{S}$. The action generation function defines the mapping $g : \mathcal{S} \mapsto \mathcal{A}$. We also define an expected reward function $r : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{R}$. The construction and approximation for g and r mappings are analogous to that of f. In our experiment, all the reinforcement is local with one of the three values (1, 0, and -1) and it is applied at the right time. In this sense, the reward function is given. The single-level algorithm is listed below.

At each level, the derive-action part is listed below:

1. Grab $x(t)$, update f from $(s(t), x(t))$ to give $s(t+1)$ (top k states).
2. Update g from $s(t+1)$ to give $a(t+1)$ (top m actions).
3. Update r from $(s(t+1), a(t+1))$ to give predicted reward $r(t+1)$. Select the non-negative actions as candidate actions from this level. Go to step 1.

5.5 The multi-level algorithm

As shown in Fig. 4, each level has a level building element. The input $x(t)$ to level i, $i > 1$ is the consecutive states from level $i-1$. temporal context. To avoid increasing the dimensionality, the resolution of input is reduced. Thus, a higher level state represents a state with longer temporal context. The maturation schedule is such that when the number of nodes at the highest level is large enough, a new level starts to be constructed automatically.

The level-based action priority in Fig. 4 raises a very interesting design issue. At each level, each generated candidate action $a(t+1)$ has an uncertainty. This uncertainty is represented by a Gaussian distribution in the action space $\mathcal{A}$ centered at the action $a(t+1)$. The variance of the Gaussian is numerically estimated by the derivative of action function g.

To determine the actual action to apply, candidate actions from different levels vote for a consensus. Graphically, the voting process is equivalent to adding all the Gaussian together in $\mathcal{A}$ space as votes. An uncertain action will vote for neighboring actions in a large area but a certain action votes only for actions in a small nearby area. The action that has accumulated the most votes is the action sent to the effector. This informal description can be formulated as conditional probability.

5.6 Information fusion and attention selection

An agent may have a number of sensors, e.g., visual sensor and auditory sensors. Not all of the sensory components in $x(t)$ are related to the goal of the agent. Therefore, information fusion requires an attention

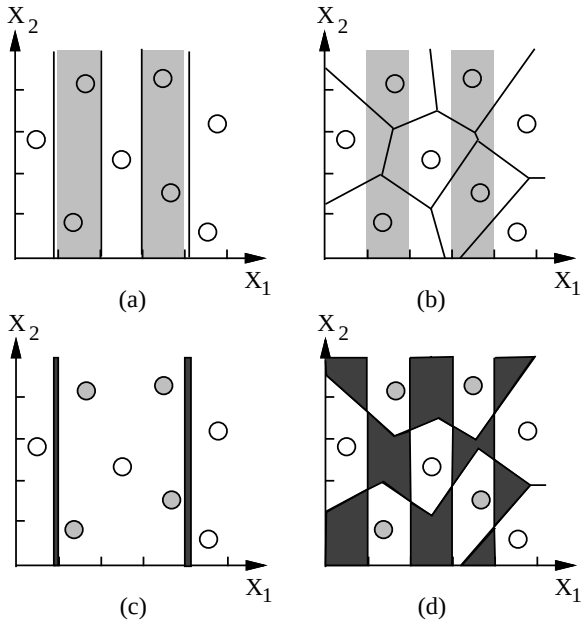


Figure 11: Importance of attention selection. (a) The decision boundary when attention mechanism selects X_1 only. The regions of two classes are marked by two types of intensities: white and gray. The circles are training samples whose intensity corresponds to the class. The decision boundaries, marked by straight line segments, are determined by the nearest neighbor rule in the space of X_1. (b) The decision boundary of the nearest neighbor rule when attention mechanism selects both X_1 and X_2. (c) Misclassified areas in case (a) are marked by dark shade. (d) Misclassified areas in case (b) are marked by dark shade.

selection capability. Thus, attention selection [14] [4] [44] [49] is a crucial mechanism for the agent to learn efficiently.

Attention selection can be divided into two types, intermodal and intramodal. Intermodal attention selection allows the agent to attend only to sensors that are related to the current task (e.g., one pays little or no attention to auditory information when one reads a book). Intramodal selection allows the agent to select a part of input from a single modality (e.g., pay attention to the face area in face recognition).

Fig. 11 explains why attention selection is critical for information fusion from multiple high-dimensional sources. In the figure, a classification problem using the nearest-neighbor rule is used for explanation. The underlying class boundaries shown in Fig. 11(a) indicate that they are independent of X_2. The random training samples are relatively sparse, which is the case in high-dimensional inputs. If the attention selection mechanism selects the right input components (X_1 in the Fig. 11(a)), the misclassification rate is small, as shown in Fig. 11(c). If the both sensory inputs X_1 and X_2 are used (Fig. 11(b)), the misclassification rate is large, as indicated in Fig. 11(d).

Cover and Hart [13] proved that if the number of independently drawn samples approaches infinity, the average classification error rate of the nearest neighbor rule is not larger than a double of the Bayes error rate. However, in reality, the number of samples is finite. Thus, as illustrated in Fig. 11, a method that does not discard unrelated components may result in a much larger error rate than the ones that do (e.g., Bayes classifier).

We model attention selection by two modules: the learned module and the programmed (innate) module. For example, when we read a book, we will not pay much attention to hearing, because the context is "reading." This is the learned module. We will switch our attention away from reading if a loud thunder is sensed by our ears. This is the programmed module. How do they interact? Consider the following phenomena: It is difficult not to notice a loud thunder (the programmed module can take over). We can

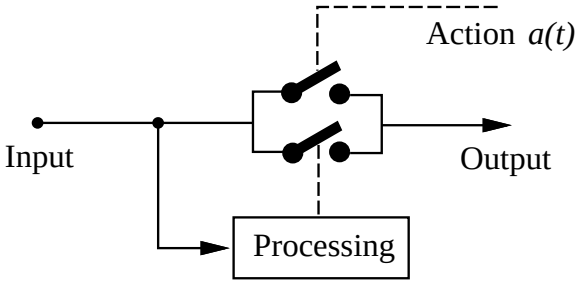


Figure 12: The attention selection mechanism has two modules, the learned module which controls the upper switch and the programmed module which controls the lower switch.

pay attention to soft clock clicking if we want to (the learned module can take over). Therefore, in our implemented attention model, the learned module and programmed module for each sensing channel have a logical OR relationship: the channel is on either the learned module or programmed module is on, as shown in Fig. 12. More sophisticated models will be incorporated in future studies.

The programmed module of attention selection depends on two factors, the intramodel novelty and intermodal capacity limit. Attention is given to channels that have a large relative novelty, and attention is shut off on channels that have a low relative novelty. To avoid constant switching of a channel on-and-off when the novelty is at a border level, the attention selection uses hysteresis: Two thresholds are used, $T_l < T_h$. If the attention is currently off, it is turned on only when the relative novelty is higher than T_h. If attention is currently on, it is turned off only when the relative novelty is lower than T_l. The novelty is currently measured as Euclidean distance between consecutive time steps of input from a sensor. The intermodal capacity limit is useful to avoid information overload. The number of channels that are turned on by the programmed module of attention selection cannot be larger than the capability limit.

The learned module of attention selection is a part of the action $a(t)$ of the system. Unlike other actions, attention selection actions typically cannot be imposed by the teacher. However, the teacher may present sensory input as needed to attract attention toward certain sensors or certain parts of a sensor using properties of the programmed module of attention selection. Attention patterns that occur either due to the programmed module or the learned module are learned and remembered by the system as inter-effector actions. Therefore, the human teacher can present sensory stimuli during various training sessions to establish the desired attention selection behaviors in the right context.

For complex sensors, such as visual sensors, there is also an intra-modal attention selector. An intra-modal attention selector for vision selects a subregion of the image for processing. For example, the region of attention in the experiments given below is a circular shape parameterized by the position and radius of a circle. The image part that is covered by this circle is normalized to a standard size before being fed to the STA. This intramodal attention mechanism will allow the agent to automatically associate partial sensory input of an object to the the identity of the object or the desired action. This will allow the system to recognize an occluded object from a partial view. For example, if the agent attends to only an eye region of a human face when it learns to recognize human faces, it will be able to recognize the person from the eye region only if the eye region is unique among the faces that the system has learned.

6 Experiments

6.1 Experimental setups

A real robot called SAIL is currently being assembled at MSU, as shown in Fig. 1. It has a computer-controlled mobile robot base adapted from a wheelchair by KISS Institute for Practical Robots. Such a base is suited for both indoor and outdoor operations. SAIL has a pan-head which allows neck-pan actions.

Two micro-cameras are mounted on two micro-pan-tilt units for independent pan and tilt by each camera. Mounted at the front of the robot is a light-weight robot arm with five degrees of freedom and a gripper. A dual-processor 330MHz Pentium II PC is its main computation engine. Microphones and speakers are its speech sensors and effectors. This platform will be used to test the architecture and the developmental algorithm outlined here.

Before real-system online tests, it is necessary to carefully study the performance characteristics of the proposed architecture and the algorithm. Here we present some studies for this need. In order to fully control the timing of sensory input presentation, action guidance and performance measurement from effectors, we used virtual time simulations while real images and speech signals were used as input. After the assembly work of our SAIL robot has been finished, we will be able to conduct real-time tests.

We have tested two simulated architecture configurations. To facilitate identity memorization, we call them "robot receptionist" (RR) and "robot horse" (RH), respectively, according to the tasks they are trained and tested upon in the experiments presented here, although neither of them has a physical body yet and neither is limited to learning these tasks. As explained earlier, the goal of the proposed architecture is to learn new tasks without a need for reprogramming. The major emphasis for RR is to study its capability to learn directly from video images in conjunction with other sensors and effectors, while that for RH is to study its capability to learn directly from sound waves using a microphone.

It is convenient to introduce simulated sensors and effectors. We call them numerical sensors and numeric effectors, respectively. A numerical sensor gives a vector of a predefined dimensionality at each time t. A numerical effector is similar to the numerical sensor, except that it is for output. A numerical sensor can be realized by a graphic user interface, a digital dial, a joystick, a key board, a file, or anything that can convert a physical measurement into one of a set of predefined codes. A numerical effector can be realized by a graphic user interface, a printer, a light, a file, or anything that can convert a set of predefined output codes into what a human being can sense directly.

6.2 Robot receptionist

We equipped the robot receptionist RR with two unbiased sensors — a video camera as the visual sensor and a numerical sensor as the simulated auditory sensor. It has also a biased sensor for applying reinforcement. Each vector at time t from the auditory sensor is considered the feature vector of a sound segment from a microphone, although such feature vectors are manually defined for simplicity. We also equip RR with two numerical effectors, one as its speaker effector and the other as its hand effector. We have implemented a two-level version of the developmental algorithm for RR, including tree indexing, pulling of states, merging of states, forgetting, and novelty-based attention selection for every sensor. In this version, we used a simple level-based priority: level 1 has a higher priority than level 0. Only when level 1 does not provide an action can the action at level 0 be used. The system diagram for RR is shown in Fig. 13.

The virtual time sessions are designed with the following scenario in mind. After the robot's "birth," we will introduce it to humans so that it can recognize people, determine their genders and act according to what its perceives and what it is taught. To proceed, training and testing sessions are interleaved appropriately through the time, according to what the teacher wants. The training events are similar to what is explained in Section 4.8. Each person presented himself or herself to the robot. During the presentation, we told the robot the person's name, the gender, and possibly the desired action. Questions about name ("who?") and gender ("gender?") were asked and the correct responses were imposed to the corresponding effector in the context. A reinforcement signal was fed into the system at appropriate times to facilitate learning.

6.2.1 Experiment using synthetic data

First, we report the result of our tests using generated synthetic data. The purpose of this study was to step through virtual time to see how the system responds exactly at each time instance. The teaching is in the action-imposed mode and no reinforcement signal is used. All of the sensory input is synthesized in the following way: only two "people" are introduced, and each is represented by a synthetic image (we do not

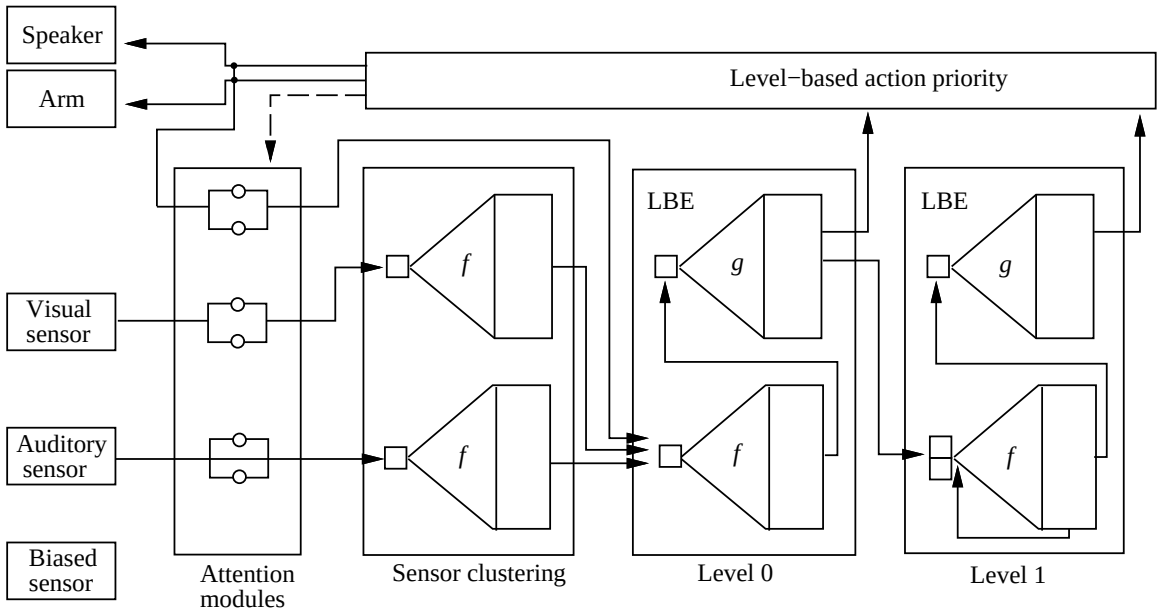


Figure 13: The configuration of RR. Learned and novelty-based attention patterns are regulated in the attention modules. Visual and auditory signals are initially clustered and then integrated with two levels of temporal context. The path from the biased sensor for reinforcement is not shown, which leads directly to action record as reinforcement in each LBE.

need it to look like a real face for this synthetic study) which is repeated enough times to form a sequence representing a static "face." A question heard through the the auditory sensor is represented by a constant vector simulating a constant tone. An action from the robot is represented by a sequence of vectors of a certain duration output from the robot's numerical effector. In the mind of the trainer, it is the value of the vectors that tells the meaning, not the time duration.

The "visual" and "auditory" sensors are made noisy by adding random noise to each signal throughout the test period, which includes interleaved training and testing sessions. The machine is trained using action-guided training to respond to a "who?" question with the action code representing the name of the person in the view. The correct recognition rate was 100(no noise) and 14db (a typical video source has a SNR better than 30db). The first recognition error occurred when the SNR gets down to 2 db. Of course, this does not mean that the system can handle SNR as low as 2db for other tasks, since the number of faces is only two here.

6.2.2 Experiment using real visual data

In this experiment, the robot went through two consecutive training sessions, one for introducing names and the other for genders. In order to see if RR can act according to the nearest case, we also taught the robot to perform follow-up actions after recognizing gender. The trainer has a simple language in mind. Each question and answer is represented by a pre-defined code in the numerical sensor and effector, respectively.

In the experiment, RR is introduced to 12 people one after another. During the teaching sessions and testing sessions, each person enters the scene, stays for a short time, and then exits the scene. The camera is fixated closely on the faces of each person so that face recognition is based on individual variations between faces and not on clothing, background, or position and occlusion of the face image. Fig.15 illustrates a segment of the video stream used for training. To teach RR how to act after a question is asked, the appropriate action is imposed to the effector at the right context.

Sequences of 100 images were used for each person for each session, with an image resolution of 50 by 50

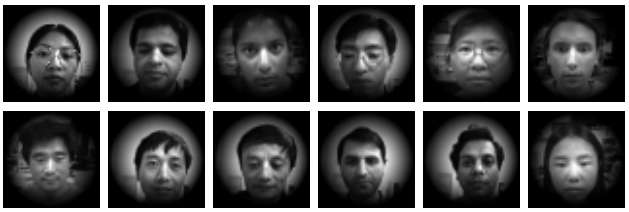


Figure 14: Individual frames of each person that RR was trained to identify. These images are taken from the video streams used for training.

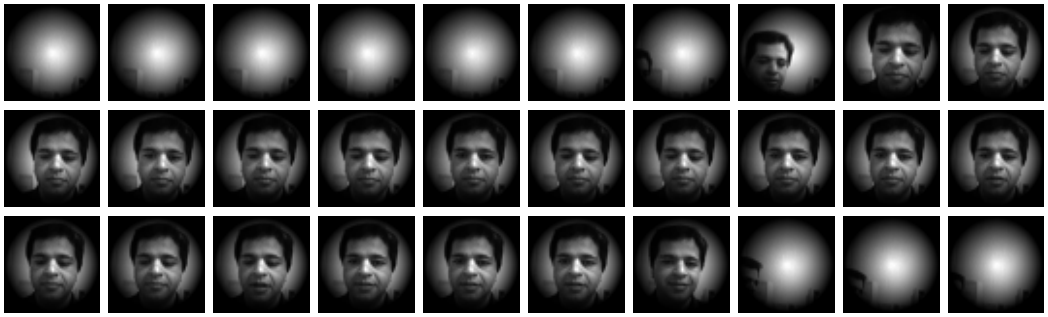


Figure 15: A temporally subsampled segment of the real-time video stream during which twelve persons were presented to RR.

pixels. The preprocessor of the visual camera includes normalization of each image frame so that the mean over all the pixels in each frame is zero and the variance of the image pixel intensities is 1. Each pixel in each image frame is weighted before being fed through as a part of the sensory input $x(t)$. The center pixel has more weight than those in the periphery to take into account the fact that human fovea has a higher visual acuity than the periphery. Zhang and his colleagues 1993 [68] explicitly investigated experimentally how the degree of "softness" of an aperture affects the segregation of figure and ground in human subjects.

In the following, we explain how each session was conducted. The first session is to introduce names to RR. When a person is presented, RR is given a coded numerical signal that corresponds to "Tell me the name of the face," and the correct coded numerical response is imposed, followed briefly by reinforcement (value 1 into the biased sensor). Therefore, in this session, we used action-imposed and reinforcement learning.

The goal of the second session is to teach RR about the gender and some behavior related to the gender. In this session, RR is given a coded numerical signal that corresponds to "Tell me the gender," and the correct coded numerical response is imposed. We like to teach RR some follow-up actions after the gender response. In this experiment, the follow-up actions we have in mind depend on the gender of the person recognized. They are "presenting a gift of a flower for a lady" and "presenting a gift of a hat for a man," respectively. To test the generalization power of RR, the follow-up action is trained only for the last male and female faces presented. Our intent in training is that this gender specific action only has to be taught once, and then RR will chain together the action of identifying gender and the presentation of an appropriate gift in every subsequent case. To accomplish this, reinforcement is applied in order to facilitate the chaining of the two actions, as we will explain later.

The training schedule of both teaching sessions is outlined in Table 4. An outline of the signals used in training at each time step is given in Tables 5 and 6.

In the following two sessions, we test RR using 12 disjoint series of images for the same 12 people it was introduced in the first two sessions. In session 3, we ask about the name while in session 4, we ask about the gender. The images presented to RR are not exactly the same as those in the first two sessions, because they are taken at different times. The size, position and expression of the faces of the human subjects may change slightly. RR is simply given the coded commands "Tell me the name of the face" and "Tell me the

Person:	First Session: "Name?"	Second Session: "Gender?"	Additional action
1	1	"Female"	None
2	2	"Male"	None
3	3	"Female"	None
4	4	"Male"	None
5	5	"Female"	None
6	6	"Male"	None
7	7	"Male"	None
8	8	"Male"	None
9	9	"Male"	None
10	10	"Male"	None
11	11	"Male"	"Present Hat"
12	12	"Female"	"Present Flower"

Table 4: A schedule of how RR was trained in the two consecutive sessions. In the first session, RR was trained to identify faces. In the second session, RR was trained to recognize gender, and to perform a gender-specific action that is chained to the action of recognizing gender.

Time Step	1200-1239	1240	1241	1242-1299
Vision	Person 1	Person 1	Person 1	Person 1
Numerical Sensor	0	"Gender?"	0	0
Imposed Action	0	"Female"	0	0
Reinforcement	0	0	1	0

Table 5: An example of the training schedule at each step. The "Gender?" signal is given at time step 40, and the response of "Female" is immediately imposed and reinforced.

gender" when a face is presented and the responses from RR are recorded. Since the AA-learning algorithm does not have different modes for training and testing, the concept of testing in this session is only in the mind of the trainer. RR still learns in this session while performing. This is the action-autonomous learning.

RR used two levels of temporal context, as shown in Fig. 13. The lower level records only the sensory signals of the current time step as a state. The higher level combines its previous state and the current and previous states of the lower level to construct its current state. This means that the lower level has virtually no context memory when making predictions, and the higher level has a memory of only a few time steps. As shown in Fig. 13, the programmed inter-modal attention selection mechanism is placed before the lowest level. Relative sensory novelty is measured on each sensory channel, and RR automatically shuts off channels with low relative novelty and focuses on channels that have a high relative novelty. The internal state of RR at the lowest level is given in Tables 7 and 8 — as can be seen, the mechanisms of involuntary novelty based attention and proprioceptive feedback induce states that are more complex than the external stimuli outlined in Tables 5 and 6. The input image, the signal from the numerical sensor, and the proprioceptive feedback are combined into a single vector $x(t)$ after the automatic attention process has modified each signal. The weights of each channel as it is represented in $x(t)$ were tuned by hand so that each channel would have a similar amount of signal variation in $x(t)$. In the future research, those weights will be adjusted automatically based on statistical variance measurement for each channel.

The test results are summarized in Table 9. As can be seen, person 1 is misidentified as person 6. This is mainly because of a significant face position difference between the training and testing sessions. This problem can be addressed once intra-modal visual attention selection is incorporated so that RR can actively

Time Step	2200-2239	2240	2241	2242	2243	2244	2245-2299
Vision	Person 11	Person 11	Person 11	Person 11	Person 11	Person 11	Person 11
Numerical Sensor	0	"Gender?"	0	0	0	0	0
Imposed Action	0	"Male"	"Present Hat"	0	0	0	0
Reinforcement	0	0	1	1	1	1	0

Table 6: Another example of the training schedule at each step. An additional action "Present Hat" is imposed after the response signal "Male" is imposed.

Time Step	1200-1239	1240	1241	1242	1243-1299
Vision	Person 1 Att on	Person 1 Att off	Person 1 Att off	Person 1 Att off	Person 1 Att on
Numerical Sensor	0 Att off	Gender? Att on	0 Att on	0 Att off	0 Attention off
Proprioceptive Feedback	0 Att off	0 Att off	"Female" Att on	0 Att on	0 Att off
Imposed Action	0	"Female"	0	0	0
Reinforcement	0	0	1	0	0

Table 7: The novelty-based attention selection mechanism and the proprioceptive feedback automatically induce states different from the raw sensory input described in Table 5

move its attention to the center of face. Since RR can learn from mistakes, we could have imposed the correct answer to extinct the incorrect answer (but we did not). The gender specific follow-up actions are chained successfully to the actions of "Male" and "Female." Table 8 illustrates why this works: a combination of proprioceptive feedback and novelty-based attention creates a state that has a response independent of the visual channel. The variation and novelty in the visual channel is relatively small compared to the novelty in the numerical sensor and proprioceptive channels, and this causes the attention system to shift away from the visual channel, thus creating a state that is characterized solely by the proprioceptive feedback of the action signal "Male." This proprioceptive feedback operates as a discriminant stimulus for the next imposed action "Present Hat" and reinforcement after the action insures that it will be repeated in the future.

Time Step	2200-2239	2240	2241	2242	2243	2244	2245-2299
Vision	Person 11 Att on	Person 11 Att off	Person 11 Att off	Person 11 Att off	Person 11 Att off	Person 11 Att on	Person 11 Att on
Numerical Sensor	0 Att off	Gender? Att on	0 Att on	0 Att off	0 Att off	0 Att off	0 Att off
Proprioceptive Feedback	0 Att off	0 Att off	"Male" Att on	"Present Hat" Att on	0 Att on	0 Att off	0 Att off
Imposed Action	0	"Male"	"Present Hat"	0	0	0	0
Reinforcement	0	0	1	1	1	1	0

Table 8: The novelty-based attention selection mechanism works to set up the "Present Hat" action as a specific response to the action "Male." These states that are automatically generated are different than the raw sensory input described in Table 6

Person:	First Session: "Name?"	Second Session: "Gender?"	Additional action
1	6	"Male"	"Present Hat"
2	2	"Male"	"Present Hat"
3	3	"Female"	"Present Flower"
4	4	"Male"	"Present Hat"
5	5	"Female"	"Present Flower"
6	6	"Male"	"Present Hat"
7	7	"Male"	"Present Hat"
8	8	"Male"	"Present Hat"
9	9	"Male"	"Present Hat"
10	10	"Male"	"Present Hat"
11	11	"Male"	"Present Hat"
12	12	"Female"	"Present Flower"

Table 9: The result of two test sessions immediately following the two training sessions described in Fig. 4. The test results indicate that person 1 is misidentified as person 6, and thus person 1 is identified with person 6's gender. The gender specific actions of presenting either a hat or a flower are successfully chained to the appropriate gender responses in all cases.

6.2.3 Discussion

The reinforcement signal is used in the training sessions — the actions of "doing nothing" (an action signal of 0) are reinforced at the conclusion of the gender-specific actions for persons 11 and 12, but "doing nothing" is never reinforced for the other gender responses for persons 1 through 10. This teaching strategy is to encourage the gender-specific follow-up action when RR sees the other people again that it knows. In animal and human learning, such a generalization is not guaranteed and it depends on the environmental and individual conditions [15]. A process called behavior shaping should be executed by the teacher to encourage every slight behavior change that is in the direction of the desired actions. In the RR case, it turns out the above reinforcement schedule is enough for the goal we have in mind.

What will happen if our reinforcement schedule was not so designed? Suppose that "doing nothing" was originally reinforced for persons 1 through 10 and then performing the gender-specific follow-up action was still required. This would represent a changing reward function — specifically, not rewarding "doing nothing" after answering gender questions for persons 1 through 10. In animal learning, this process is called extinction (extincting the behavior previously reinforced). This can be done by updating the reinforcement record with the default reinforcement value 0, meaning "doing nothing is not encouraged now." After the reinforcement for "doing nothing" was extincted, RR would select the next closest state representing a reinforced state-action pair, and would adopt the gender-specific follow-up action.

This brings up the important issue of individualized education well-known to human education. If robots were "raised" in very different environments, different teaching schedules may be required to teach them the same behavior.

The system of inter-modal novelty-based attention used here is loosely based on human and animal models of involuntary attention and attention capacity. It is simplistic, but it illustrates that attention selection can work to facilitate discrimination between stimuli. More complex attention behavior would be characterized by learned attention patterns, and those would work to deal with more complex situations.

Figure 16: The mobile robot Rome running SHOSLIF navigates autonomously at a walking speed, along hallways, turning at corners and passing through a hallway door. The real-time, on-line, incremental learning and the real-time performance is accomplished by an on-board Sun SPARC-1 workstation and a SunVideo image digitizer, without any other special-purpose image processing hardware. No active sensors, such as sonars or infrared sensors were used.

6.3 Robot "horse"

6.3.1 Motivation of experiment

Since our goal with "robot horse" (RH) is to test the capability of learning directly from a sound signal, we give RH an auditory sensor. We give it also a numerical sensor, simulating its touch sense for rein when it is being pulled. In our prior work for autonomous navigation [66], we have used SHOSLIF to implement a single f function shown as sensor clustering in Fig. 13 to map visual input directly to a navigation signal (heading, direction, and speed). It has been demonstrated that this tree-based mapping enables our Rome robot to navigate in real-time in our Engineering Building, using only a single sensor — a video camera, as shown in Fig. 16. However, Rome does not yet have a way to act interactively according to human verbal commands, like what a horse can do. Here, we are interested in the possibility of adding an interactive mode for a "robot horse."

The rein sensor is used to teach the horse to listen and act only when the rein is pulled. When the rein is not pulled (simulated by 0 of the numerical sensor), it will roam along using the low-level reflex actions that have been demonstrated with the Rome robot. When a verbal command is needed, e.g., to make a turn at an intersection, the human teacher will pronounce a vowel sound and at the same time pull the rein (simulated by the numerical sensor). The verbal commands tell RH to turn left, turn right, speed up, slow down, etc. With multisensor fusion, in this case, the coupling of verbal signal with the rein signal, RH will less likely be confused by various background sounds in the environment.

The architecture configuration of RH is similar to that of RR shown in Fig. 13, except that it has no attention selection for simplicity.

We used four vowels: "a" (hot) , "e" (bet), "i" (tree) and "u" (boot), representing "left," "right," "faster" and "slower," respectively.

6.3.2 Sound

According to acoustic theory, there are three separate areas for modeling in a human speech production [25], [45], [47]. These include the source excitation, vocal tract modeling, and the effect of speech radiation. The first two components are the major two parts which decide speech characteristics. Speech is considered a signal which is composed of an excitation sequence *convolved* with the impulse response of the vocal system model:

$$s(n) = e(n) \star \theta(n)$$

where θ is a vocal tract component and $e(n)$ is an excitation sequence. The Cepstrum of $s(n)$ is defined as

$$c_s(n) = F^{-1}\{\log|F\{s(n)\}|\}$$

where F denotes the discrete Fourier transform. In digital version, $s(n)$ is divided into equal-length slightly overlapped segments. The length of each segment is about 20 ms in time. Each segment corresponds to a discrete time t in our AA-learning algorithm and it is used to compute a Cepstrum vector $c_s(t)$. It has been shown that first several components in the vector $c_s(t)$ result in a good feature vector that characterizes the

vocal tract components for voiced speech $\theta(n)$ during this short time period. We used "Mel-based Cepstrum" in which DFT magnitude spectrum is modified to better approximate the human auditory characteristics [45] [54].

In our experiment, the preprocessor of the auditory sensor does the following. The sound is digitized at 10 kHz. Every segment of 256 points in time is used to compute a 16-dimensional Mel-Cepstrum feature vector $c_s(k)$. Thus, after the preprocessing, the sound signal can be considered a series of 16-dimensional vectors.

The human subjects are composed of three adult males, one adult woman and one young girl. For generating training and testing data, each person uttered five times for each vowel. All the data was recorded in a noisy air-conditioned lab for a more realistic environment.

It is known that for men, the possible pitch range is usually found somewhere between the two bounds of 50-250 Hz, while for women the range usually somewhere in the interval 120-500Hz. Thus, in general, recognizing women's speech may be more difficult than men's.

6.3.3 State

Using the 16-dimensional Cepstrum vector, a 2:1 dimension reduction in state definition shown in Fig. 5 provides too little temporal context. To provide more context for the state vector, we define the state vector $s(t)$ to have five segments. They correspond to $x(t-4)$, $x(t-3)$, $x(t-2)$, $x(t-1)$ and $x(t)$, respectively. The first two segments $x(t-4)$ and $x(t-3)$ are the 2:1 resampled version. The others are the full resolution versions. Each segment $x(t-i)$ is computed from $x(t-i+1)$, $i = 2, 3, 4, 5$. Due to the state pulling and merging, the state vector $s(t)$ is a recursively wrapped version of input sequence $x(t-4)$, $x(t-3)$, $x(t-2)$, $x(t-1)$, $x(t)$, instead of a direct shifted version. This is an important property for dealing with the time-warping phenomena of speech.

6.3.4 Training and test sessions

We conducted speaker independent tests. In other words, the persons who were tested were not among the persons whose sounds have been used in the training sessions. To get an average performance for RH, we conducted leave-one-out tests. In other words, five experiments were conducted. For each experiment, a different person's utterances were used in the test session. The other four persons' utterances were used for the training session. The performance recorded is the average correct action rate over the five experiments.

Each training session is conducted in the following way. All of the five utterances of each person are played one at a time with a sufficient interval. During each sound, the rein sensor is turned on. At the off-set of the rein sensor signal, the desired action is imposed. There are four actions corresponding with four vowels. For each action, a total of 20 training sounds (4 persons, 5 utterances from each person) are heard by RH and the corresponding action is imposed. With four actions, RH has acted 80 times in the training session. This is action-imposed learning.

During the test session, all the five utterances of the test person whose sounds were not used in the training session, are played one after another. In the middle of the sound, the rein sensor is set on. The response action of RH is recorded. The test result is summarized in Table 10. Persons 1, 2, and 3 represent male adults, 4 the female adult and 5 the young girl. The recognition rates are all 100%. Since this is a speaker independent test, the test person is different from the training persons in gender and age. For sample, RR has recognized all the little girls verbal commands correctly after it has been trained with the voices of 3 adult men and one adult lady. We can expect that the small vocabulary is a major factor for the perfect performance.

To push the system to the limit, we added another vowel "o" to the vocabulary. It may represent a new action. It is known that the waveform and Cepstrum coefficient vectors of "o" and "u" are close. The similar leave-one-out experiments are performed. This time, we conduct speaker dependent test. For every person, four of the utterances are used for training and the remaining one for testing. The average rate of correct actions were recorded during the test sessions. The results are summarized in Fig. 11. Since the recognition rate is not perfect, we retain the information about the vowels in the table.

Vowel	Meaning	Person 1	Person 2	Person 3	Person 4	Person 5
"a"	left	100 %	100 %	100 %	100 %	100 %
"e"	right	100 %	100 %	100 %	100 %	100 %
"i"	faster	100 %	100 %	100 %	100 %	100 %
"u"	slower	100 %	100 %	100 %	100 %	100 %

Table 10: The average recognition rates for speaker independent 4-vowel tests.

Vowel	Person 1	Person 2	Person 3	Person 4	Person 5
"a"	100 %	100 %	100 %	100 %	100 %
"e"	100 %	100 %	100 %	100 %	100 %
"i"	100 %	100 %	100 %	100 %	100 %
"o"	100 %	100 %	100 %	100 %	96 %
"u"	100 %	100 %	100 %	100 %	100 %

Table 11: The average recognition rates for speaker dependent 5-vowel tests.

As one might expect, misclassification only happens for the vowel "o". Actually, about 4% of the "o" vowel sounds are recognized as "u" and the rest of cases are correct. On the other hand, it is not always trivial for a human to perfectly distinguish individually pronounced "o" and "u." Complete words and sentence context can help greatly. Future studies along this line will test words and sentences using more levels of the proposed architecture.

7 Conclusions

The developmental approach is introduced here along with an architecture for development and a description of our developmental algorithms. It is known that subsystems of biological organisms are developed and integrated through their developmental process, during which nature (what is innate) and nurture (experience) interact. This suggests that system integration for an intelligent machine can be accomplished in the process of development. With the proposed developmental approach, the integration of subsystems is a natural part of developmental learning, during which, the multi-modality context can be naturally taken into account.

The developmental approach requires a new way of learning for machines, one termed AA-learning in this paper. The required algorithm is called the developmental algorithm. A fundamental implication of this new way of learning is a general purpose learner — one that is capable of learning new tasks and new behaviors without a need for reprogramming. The capability of such a machine will depend on its sensors, its effectors, its computational resources, its developmental algorithm, and how it is taught. The kinds of tasks that such a machine can accomplish do not depend on whether humans have ever thought about them, let alone whether humans have ever found a good representation to program about them. This opens the possibility for machines to learn tasks that are too complex for humans to provide an adequate task-specific representation, such as many tasks that require vision, speech, language acquisition, thinking, and innovation.

However, the developmental approach is just at its fledging stage. Some important subjects that this article does not have space to cover can be found in [63]. The result of the work presented here is very encouraging but still preliminary. A rich array of interesting research topics need to be studied and a large amount of challenging research work remain to be done.

Acknowledgements

This work was supported in part by NSF grant No. IIS 9815191 and DARPA contract No. DAAN02-98-C-4025.

References

[1] J. Aloimonos. Purposive and qualitative active vision. In *Proc. 10th Int'l Conf. Pattern Recognition*, pages 346–360, Atlantic City, NJ, June 1990.

[2] M. H. Ashcraft. *Human Memory and Cognition.* Harper Collins College Publishers, New Royk, NY, 1994.

[3] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman. Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection,. In *Proc. European Conf. on Computer Vision*, April 1996.

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

[5] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees.* Chapman & Hall, New York, 1993.

[6] R. Brooks. Intelligence without reason. In *Proc. Int'l Joint Conf. on Artificial Intelligence*, pages 569–595, Sydney, Australia, August 1991.

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

[8] P. E. Bryant and T. Trabasso. Transitive inferences and memory in young children. *Nature*, 232:456–458, 1971.

[9] S. Carey. *Conceptual Change in Childhood.* The MIT Press, Chambridge, MA, 1985.

[10] S. Carey. Cognitive development. In D. N. Osherson and E. E. Smith, editors, *Thinking*, pages 147 – 172. MIT Press, Cambridge, MA, 1990.

[11] G. A. Carpenter, S. Grossberg, and J. H. Reynolds. ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural networks. *Neural Networks*, 4:565–588, 1991.

[12] P. A. Chou. Optimal partitioning for classification and regression trees. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13:340–354, 1991.

[13] T. M. Cover and P. E. Hart. Nearest neighbor pattern classification. *IEEE Trans. Information Theory*, 13:21–27, Jan. 1967.

[14] A. Van der Heijden. *Selective Attention in Vision.* Routledge, New York, NY, 1992.

[15] M. Domjan. *The Principles of Learning and Behavior.* Brooks/Cole, Belmont, CA, fourth edition, 1998.

[16] M. Dorigo and M. Colombetti. Robot shaping: Developing autonomous agents through learning. *Artificial Intelligence*, 71(2):321–370, 1994.

[17] J.L. Elman. Learning and development in neural networks: The importance of starting small. *Cognition*, 48:71–99, 1993.

[18] J.L. Elman, E. A. Bates, M. H. Johnson, A. Karmiloff-Smith, D. Parisi, and K. Plunkett. *Rethinking Innateness: A connectionist perspective on development.* MIT Press, Cambridge, MA, 1997.

[19] K. Etemad and R. Chellappa. Discriminant analysis for recognition of human face images. In *Proc. Int'l Conf. Acoust., Speech, Signal Processing*, pages 2148–2151, Atlanta, Georgia, May 1994.

[20] K. Fukunaga. *Introduction to Statistical Pattern Recognition.* Academic Press, New York, NY, second edition, 1990.

[21] H. E. Gruber and J. J. Voneche. *The essential Piaget.* Basic Books, New York, 1977.

[22] D. J. Hand. *Discrimination and Classification.* Wiley, Chichester, 1981.

[23] I. Harvey. Evolutionary robotic and SAGA: The case for hill crawling and tournament selection. Technical Report CSRP 222, University of Sussex, Brighton, U.K., 1992.

[24] Jr. J. L. Martinez and R. P. Kessner (eds.). *Learning & Memory: A Biological View.* Academic Press, San Diego, CA, 2 edition, 1991.

[25] Jr. J. R. Deller, Jone G. Proakis, and John H. L. Hansen. *Discrete-Time Processing of Speech Signals.* Macmillan, New York, NY, 1993.

[26] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data.* Prentice-Hall, New Jersey, 1988.

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

[28] 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.

[29] T. Kohonen. *Self-Organization and Associative Memory.* Springer-Verlag, Berlin, second edition, 1988.

[30] T. Kohonen. *Self-Organizing Maps.* Springer-Verlag, Berlin, second edition, 1997.

[31] B. Kolb and I. Q. Whishaw. *Fundamentals of Human Neuropsychology.* Freeman, New York, third edition, 1990.

[32] P. Langley. Machine learning for intelligent systems. In *Proc. 14th National Conf. on Artificial Intelligence*, pages 763–769, Providence, RI, July 1997.

[33] D. B. Lenat. CYC: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38, 1995.

[34] D. B. Lenat, G. Miller, and T. T. Yokoi. CYC, WordNet, and EDR: Critiques and responses. *Communications of the ACM*, 38(11):45–48, 1995.

[35] K. C. Li. Sliced inverse regression for dimension reduction" (with discussion). *Journal of the American Statistical Association*, 86:316–342, 1991.

[36] K. C. Li. On principle hessian directions for data visualization and dimension reduction: another application of Stein's lamma. *Journal of the American Statistical Association*, 87:1025–1039, 1992.

[37] D. G. Luenberger. *Optimization by Vector Space Methods.* Wiley, New York, 1969.

[38] S. Mahadevan. Average reward reinforcement learning: Foundation, algorithms, and empirical results. *Machine Learning*, 22:159–196, 1996.

[39] S. Mahadevan and Leslie P. Kaelbling. The national science foundation workshop on reinforcement learning. *AI Magazine*, 17(4):89–93, 1996.

[40] K. McKusick and P. Langley. Constrains on tree structure in concept formation. In *Proc. Int'l Joint Conf. Art. Intell.*, volume 2, pages 810–816, Sydney, Australia, Aug. 1991.

[41] G. A. Miller. Worknet: A lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995.

[42] M. Minsky. *The Society of Mind.* Simon and Schuster, New York, NY, 1986.

[43] H. Murase and S. K. Nayar. Visual learning and recognition of 3-D objects from appearance. *Int'l Journal of Computer Vision*, 14(1):5–24, January 1995.

[44] B. A. Olshansen, C. H. Anderson, and D. C. Van Essen. A neurobiological model of visual attention and invariant pattern recognition based on dynamic routing of information. *Journal of Neuroscience*, 13(11):4700–4719, 1993.

[45] Douglas O'Shaughanessy. *Speech Communication Human and Machine.* Addison-Wesley, Reaing, MA, 1987.

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

[47] L. Rabiner and B. Juang. *Fundamentals of Speech Recognition.* Prentice Hall, Englewood Cliffs, NJ, 1993.

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

[49] D. Reisfeld, H. Woldson, and Y. Yeshurun. Context-free attentional operators: the generalized symmetry transform. *Int'l Journal of Computer Vision*, 14:119–130, 1995.

[50] W. Rudin. *Real and Complex Analysis.* McGraw-Hill, New York, 3 edition, 1987.

[51] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach.* Prentice-Hall, Upper Saddle River, New Jersey, 1995.

[52] A. Schwartz. A reinforcement learning method for maximizing undiscounted rewards. In *Proc. Int'l Joint Conf. on Artificial Intelligence*, pages 289–305, Chambery, France, 1993.

[53] Luc Steels. Emergent functionality in robotic agents through on-line evolution. In R. A. Brooks and P. Maes, editors, *Articificial Life IV*, pages 8–14. MIT Press, Cambridge, Massochusetts, 1994.

[54] S. S. Stevens and J. Volkman. The relation of pitch to frequency. *American Journal of Psychology*, 53:329, 1940.

[55] P. Stokes and P. D. Balsam. Effects of reinforcing preselected approximations on the topography of the rat's bar press. *Journal of the Experimental Analysis of Behavior*, 55:213–231, 1991.

[56] D. Swets and J. Weng. Discriminant analysis and eigenspace partition tree for face and object recognition from views. In *Proc. Int'l Conference on Automatic Face- and Gesture-Recognition*, pages 192–197, Killington, Vermont, Oct. 14-16 1996.

[57] D. L. Swets and J. Weng. Using discriminant eigenfeatures for image retrieval. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 18(8):831–836, 1996.

[58] S. Thrun. Lifelong learning: A case stuty. Technical Report CMU-CS-95-208, Carnegie Mellon University, Pittsburgh, PA, Nov. 1995.

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

[60] C. Watkins. Learning from delayed rewards. Technical report, PhD thesis, King's College, Cambridge, England, 1989.

[61] W. X. Wen, H. Liu, and A. Jennings. Self-generating neural networks. In *Proc. Int'l Joint Conf. Neural Networks*, volume 4, pages 850–855, Baltimore, Maryland, June 1992.

[62] J. Weng. Cresceptron and SHOSLIF: Toward comprehensive visual learning. In S. K. Nayar and T. Poggio, editors, *Early Visual Learning*. Oxford University Press, New York, 1996.

[63] J. Weng. The living machine initiative. Technical Report CPS 96-60, Department of Computer Science, Michigan State University, East Lansing, MI, Dec. 1996.

[64] J. Weng, N. Ahuja, and T. S. Huang. Cresceptron: a self-organizing neural network which grows adaptively. In *Proc. Int'l Joint Conference on Neural Networks*, volume 1, pages 576–581, Baltimore, Maryland, 1992.

[65] J. Weng, N. Ahuja, and T. S. Huang. Learning recognition using the Cresceptron. *Int'l Journal of Computer Vision*, 25(2):109–143, Nov. 1997.

[66] J. Weng and S. Chen. Incremental learning for vision-based navigation. In *Proc. Int'l Conf. Pattern Recognition*, volume IV, pages 45–49, Vienna, Austria, Aug. 25-30 1996.

[67] S. Wilson. Classifier systems and the Animat problem. *Machine Learning*, 2(3):199–228, 1987.

[68] J. Zhang, S. L. Yeh, and K. K. De Valois. Motion contrast and motion integration. *Vision Research*, 33:2721–2732, 1993.