Learning in Computer Vision and Beyond: Development

John J. Weng

Department of Computer Science

Michigan State University

East Lansing, MI 48824

weng@cps.msu.edu

Abstract

This chapter introduces what is called the *developmental* approach to computer vision in particular and artificial intelligence in general. It discusses the current basic paradigm for developing a system and its fundamental limitations. The developmental approach is motivated by human cognitive development from infancy to adulthood. A developmental learning algorithm is determined before the "birth" of the system. After the "birth", it enables the system to learn new tasks without a need for reprogramming. The major goal of the developmental approach is to realize automation of general-purpose learning that enables machines to perform developmental learning over a long period. Such learning is conducted in a mode similar to the way animals (and humans) learn. The machines must learn directly from continuous sensory input streams while interacting with the environment including human teachers. In this learning mode, developing intelligent programs for various tasks is realized through real-time interac-

tions with the machines, including demonstration, communication, action imposition, granting rewards and executing punishments. This requires a fundamentally new way of addressing the learning problem, one that unifies learning and performance phases and requires a systematic self-organization capability. This chapter discusses these fundamental issues and explains our related work in this direction.

1 Motivations

Despite the power of modern computers, whose principle was first introduced in 1936 by Alan Turing in his now celebrated paper [103], we have seen a paradoxical picture of artificial intelligence: Computers have done very well in some areas that are typically considered very difficult (by humans), such as playing chess games; but they have done poorly in other areas that are commonly considered easy (by humans), such as vision [77].

1.1 Challenges

It seems a relatively simpler task to write a program to solve a symbolic problem where the problem is well-defined and the input is human-preprocessed symbolic data. However, it is very difficult for a computer to solve a problem that is not definable mathematically using the raw sensory data in their original form.

For example, although there have been some limited applications in controlled settings [34], image understanding is extremely difficult, especially for tasks such as recognizing objects in a general setting. The recognition must cope with many variation factors, such as lighting, viewing angle, viewing distance and object changes (e.g., facial expressions). The difficulties encountered in computer vision might not be surprising if one realizes that it is the most difficult sensing modality in human. In fact, a half of the cerebral cortex in the human brain is given over to visual information processing. As is well known, a large portion of human knowledge is acquired through vision.

The speech recognition field has some limited applications for recognizing words and short sentences in a controlled setting [1] [106]. However, further advances for general settings requires understanding the situation, context, speaker's intention, speaker characteristics, language and

the meaning of what is said [45] [123]. In the natural language understanding field, we have seen various low-level applications in language processing from text inputs, such as spell checkers, grammar checkers and keyword-based search [18], but these low-level applications do not require true understanding of text. It has been known that language understanding requires not only syntax but also semantics and "common sense" knowledge. Several grandiose language knowledge-bases are being developed, such as the common-sense knowledge-base, CYC [57] [58] and lexical database, WordNet [67]. However, it is an open question whether a machine can really understand anything in a pure text form without its own experience. Is linking from one string of letters to another true "understanding"? Can the system use text properly in, e.g., language translation? Language discourse and language translation are good tests for language understanding. However, tremendous difficulties persist in these areas. Like speech recognition, hand-written character recognition without understanding the meaning of the context cannot go very far.

Recently, several alternative methods have been actively studied, including various artificial neural networks and genetic algorithms. Many encouraging studies have demonstrated that these alternative methods can achieve impressive results that are difficult to achieve using traditional text-based or knowledge-based methods. More recently behavior-based robotics research puts emphasis on embodiment, situatedness and behavior generation through interactive learning [11], challenging the traditional disembodied methods. A more flexible learning mode, reinforcement learning, has also been actively studied for various problems [47]. However, it is not clear how these alternative methods can be scaled up for dealing with the challenging tasks discussed above.

1.2 The task-specific paradigm

The current major paradigm for developing a system, either intelligent or not, 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) They are manually specified using hand-crafted domain knowledge (e.g., the knowledge-based methods in CYC [57] [58] and WorldNet [67]). (b) They are decomposed manually into system behavior modules (e.g., behavior-based methods in the subsumption architecture [13] and active vision [2]). (c) They are estimated using a training procedure (e.g., learning-based methods in Q-learning [107], eigenfaces [104] and SHOSLIF [116]). (d) They are searched for based on a task-specific objective function (e.g., genetic or artificial life methods in Animate [120], SAGA [36], and AutonoMouse [27]). It is typical to use a combination of the various methods in step (5) (e.g., AutonoMouse [27] 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 it task-specific paradigm since it starts with a task and all the rest steps depend on the task.

This task-specific paradigm has produced impressive results for those tasks whose space is relatively small, relatively clean (or exact) and relatively easy to model, such as chess or printed character recognition. However, it faces tremendous difficulties for tasks whose space is huge, vague, difficult to fully understand and difficult to model adequately by hand. Two major restrictions are direct consequences from such a task-specific paradigm:

1. The task-space and representation defined manually by humans cannot deal with the full complexity of the real world.

2. Low quantity and low quality of information fed into the system for training.

Due to the limitation of human in modeling tasks, especially in uncontrolled real-world environments, the task space defined by the humans are of a limited scope. Such a limited scope is the major reason for the high brittleness of the existing systems. They cannot extend their capabilities to deal with more complex cases without humans to remodel the task and its representation again.

The brittleness of the system is also attributed to the quantity and quality of the training data. In terms of quantity, the computers are allowed to observe far less environmental variation, context variation, and content domain variation than they really need for the tasks. It is difficult to conduct extensive system training due to the large amount of manual labor that is required in preparing the training data. In terms of quality, these compiled training data are very much *disconnected* from the environment from which the data arise. The rich meaning of the live sensory experience is degenerated into isolated segments, each being tied to a class label which is meaningless to the system. The lack of environmental context in manually fed training data makes it impossible for machines to learn beyond what is possible from the "spoon-fed" data.

Some other systems do not use learning. Instead, the required knowledge is directly programmed into the system. Unfortunately, what is modeled by such knowledge-based programming is typically insufficient for the challenging tasks at hand, due to human limitation in understanding and modeling the complex mechanisms of the required cognitive process. For example, recognition of human faces must cope with a wide variety of variation factors, such as lighting, viewing angle, viewing distance, and facial changes (e.g., expressions, hair styles, eye wear, etc). A similar situation is

true in speech recognition (e.g., variation in time warping, coarticulation, intonation, age, gender, etc) and language understanding (ambiguity without context, ambiguity without understanding, cultural differences, language differences).

In summary, it is extremely difficult, if not impossible, for human designers to adequately represent and model many factors in a challenging task, to design effective knowledge-level rules for them, to collect sufficient training data to cover the variations, and to keep the system up to date. Certain degree of automation is very desirable for these challenging problems.

Therefore, we need to automate not just only the learning phase, but also to automate the design of task-space (or problem space).

2 The Developmental Approach

How does a human being establish his or her cognitive capability? Studies in developmental psychology may shed light on this important question.

2.1 Human cognitive development

Jean Piaget [35] [17] [16], a well known developmental psychologist, proposed to divide human cognitive development roughly into four major stages, as summarized in Table 1. There is no doubt that these four stages have a lot to do with neural development in the brain. 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 [15] 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 stage partition proposed by Piaget.

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

However, it is known that the development of cognitive capability of each human individual requires many years of learning during which he or she interacts with the environments and learns, while the brain develops to store, accumulate, enrich, generalize and verify the knowledge learned. The biological learning algorithm that is determined at birth time of a human being enables the human individual to learn more and more tasks through his life time without a need of reprogramming.

2.2 AA-learning

In order to bring out the stark contrast between the developmental learning by an animal and the current task-specific paradigm, we first introduce some basic concepts.

We introduce the concept of AA-learning (named after *a*utomated, *a*nimal-like learning *without claiming to be complete*) for a machine agent[1]. AA-learning has the following characteristics. (1) Domain extensibility. (2) Closedness of the brain. (3) Simultaneity of learning and performing. (4) Integration of different learning types. (5) Automatic level building. (6) Real-time while scaling up.

[1] An agent is something that perceives and acts [89].

Domain extensibility Domain-extensible means that the system is applicable to an open number of task domains and can learn new task domains without a need for re-programming. A domain-expandable system like a human can continuously switching among deferent domains. For example, while reading a magazine, human can switch among recognizing human faces, recognizing human genders, recognizing written characters and recognizing other objects. For a domain-extensible system, the system designer cannot use a task-specific objective function (e.g., in training a robot leg hopper, the objective is fixed — keeping balance). He cannot define a specific task space for programming either. Existing learning systems that use a general-purpose learning method typically require the system designer to map a task into the program internal representation (e.g., to map all the possible situations of a task into a set of manually defined system states, such as in Robo-Soar [55]). It is worth noting that domain-extensibility is not inconsistent with brain modularity. Each domain task may be realized by a particular set of modules in the brain[2].

Closedness of the brain The internal states of the "brain"[3] cannot be directly set by external probes for teaching purpose, although they are accessible for research purpose. This is an important characteristic of automated animal learning. Otherwise, a mother must understand the internal brain representation of her daughter before she can teach her. The environment, including human teachers, can only affect the learner's brain through his or her sensors (showing examples, or giving appetitive or aversive stimulus) and effectors (forced actions). Many existing learning methods require humans to directly link internal representation to actual meaning of the content to be

[2]For example, a tree can be used to partition the input space into regions, each may be considered as a module. For example, human faces may be assigned to a set of such regions represented by a set of notes in a tree.

[3]For simplicity, we will drop the double quotes for the term "brain" with an understanding that an artificial brain is very different from a real biological brain.

learned. The closed brain requirement relieved human from manually performing this extremely difficult task.

Simultaneity of learning and performing The learning phase is also the performance phase. Humans are not in the loop of collecting training data. Humans are a part of the environment that the system *continuously* interacts with. The system will learn from the real-world environment directly and continuously using its sensors and effectors, without requiring humans to serve as a feature detector or sensory-input-to-symbol converter. This is in contrast with symbolic methods (which works in a symbolic domain) and softbots (which work only in a simulation world). The real-word excludes simulated computer world, such as that used by Shen in his study of autonomous learning from simulated symbolic world [92].

Integration of different learning types Reinforcement learning (using reward and punishment) and supervised learning (guided actions) are combined in the open-mode learning. No behavior is hard-wired in (or programmed in), since such hard-wired behavior cannot work well with other behaviors when the system later learns more sophisticated tasks. Some basic behaviors that are typically innate in animals, such as moving forward and simple turning, are taught and memorized through interactive effector guidance (imposition). This allows the system to start with some basic simple behaviors which will facilitate further learning.

Automatic level building Capabilities for learning stimulus-response association, reasoning and prediction are automatically built up from low level to higher levels. The system must automatically build a representation for concepts at different scales [4]. This is in contrast with existing single-level

[4]For examples, due to the need for understanding visual language (e.g., American sign language) or spoken language (e.g., English, Chinese, or French), the system must be able to automatically learn and recognize words,

reinforcement learning methods such as Q-learning [107] and R-learning [91], manually specifying and building behavior layers in the subsumption architecture [11], and human content-level design in level building for speech recognition (e.g., [83] [43]).

Real-time while scaling up The system must learn while performing in real-time, since it must observe the consequence of its actions as they happen so that it can learn from the events. During the entire developmental life of the system, this real-time requirement must be satisfied while the number of cases the system has learned increases without bound. This is a highly challenging requirement for high-dimensional sensory inputs, such as visual input, and for large-size tasks.

2.3 Living machines

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[5]. 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 the 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, an *extroceptive* sensor is one that senses external environment (e.g., visual); a *proprioceptive* sensor senses relative position of internal control (e.g., arm position) and an *interoceptive* sensor is one that senses internal events (e.g., internal clock).

phrases and sentences from an environment where one or several languages are used in teaching, without the need to be explicitly programmed for any particular language.

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

We need to accommodate a class of synthetic sensors that typically are not called sensors, such as keyboard input, numerical input from a graphical user interface, etc. For effectors, the system may also output text or numetircal numbers. Therefore, it is convenient to define *N-sensors* and *N-effectors*, where N stands for "numerical." The input from an N-sensor at time t is a vector of a dimensionality difined for the sensor. The output to N-effector is also a vector of certain dimensionality. The meaning of the input from an N-sensor and that of the output to an N-effector are pre-defined by the sensor or effector. Now, we are ready to give living machine a more precise definition.

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 state-update function f updates the "brain" based on sensory input $x(t)$ and the current "brain" $b(t)$:*

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

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

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

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 purpose. It can only be updated according to Eq. (1).

A machine agent is called a "*living machine*" if it can perform AA-learning continuously for very long time to learn to perform various tasks.

For the AA-learning, the learner has three types of channels with its environment: sensors, effectors, and reward receiver, as shown in Fig. 1 The reward receivers can be modeled by biased

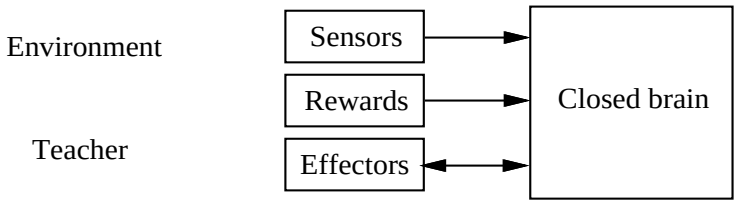


Figure 1: A living machine has three channels to interact with the environment, sensors, effectors and reward receiver. The dashed arrows mean that a human teacher or the environment have access to the corresponding component. The double arrow for the effectors means that actions imposed by humans or the environments on some effectors can be sensed by the brain (through *proprioceptive* sensors).

sensors when the agent is "young". They will be enriched by unbiased sensors after the agent has developed its preference pattern for unbiased sensors.

After its birth, a living machine learns autonomously from the environment by sensing the environment through its sensors and acting on the environment through its effectors. Among all the possible values received from the reward receiver, it has a predefined degrees of preference. It likes appetitive stimulus and dislike aversive stimulus. The teacher is a part of the environment. The environment can enforce an action of the effectors on the learner. Human teachers, as a part of the system's environment, affect how the system learns. For example, the human teachers will show different object examples, verbally state the characteristics of the object and then ask questions immediately about the characteristics of the object. He may encourage the system to act properly

using different rewards at appropriate right time. He can also directly impose the desired action to execute by impose control values to the corresponding effector. Such an imposition of action and delivery of rewards occur also in human learning. For example, manipulating a child's hand to hold a pen when teaching a child how to use a pen. Rewards to a human child can be food, a good test score etc.

We call this type of general-purpose AA-learning machines "living machines" because they "live" in the human environment and autonomously interact with the environment (including humans) on a daily basis. The emphasis of the term is not "life", but rather, the daily autonomous learning activities that are associated with a biological living thing, especially human, such as playing, communicating with humans and learning to perform tasks. Such machines are fundamentally different from a nonliving regular machine, such as an automobile or a computer, since they do not operate autonomously.

2.4 Why living machines?

Can the major problems for computer vision be solved by dealing with only the visual modality? Can the current task-specific paradigm leads us to solve the challenging problems in artificial intelligence? Why living machines? This chapter will not be able to fully discuss these issues. Here, we examine some major points.

2.4.1 Each modality must be learned

The cognitive knowledge that is required to communicate with humans in single or multiple modalities in a general setting is too vast in amount and too complicated in nature to be manually modeled adequately and manually spoon-fed sufficiently into a program. Probably few will question the fact

that language is learned. Therefore, vision, the modality that many human individuals do not feel requires much learning (i.e., learned subconsciously), is appropriate to demonstrate the importance of learning. How complete is a child's vision system when he or she is born? In fact, as early as the late 19th century, German psychiatrist Paul Emil Flechsig had shown that certain regions of the brain, among them V1, have a mature appearance at birth, whereas other cortical areas including V2, V3, V4, and V5 regions, continue to develop, as though their maturation depended on the acquisition of experience [124]. A lot of studies have been done since then. For example, a cat (kitten) that has only seen vertical lines ever since its birth cannot see horizontal lines [8]. It is known that learning plays a central role in the development of human versatile visual capabilities and it takes place over a long period (e.g., Carey [16], Hubel [39], Anderson [4], Martinez & Kessner [42]). Human vision appears to be more a process of learning and recalling than one that relies on understanding of the physical processes of image formation and object-modeling (e.g., the "Thatcher's illusion" [99] and the overhead light source assumption in shape from shading [84]). Neurologist Oliver Sacks' report [90] indicated that a biologically healthy, an adult human vision system that has not learned cannot function as we take for granted.

Furthermore, recognition by humans takes into account information sources that are not confined to vision. Sinha & Poggio's Clinton-Gore example [93] indicates that humans integrate different sensing modalities and contextual information for face recognition [6] With humans, visual learning takes place while the recognizer is *continuously* sensing the visual world around it and interacting with the environment through human actions.

A large amount of evidence seems to suggest that except for low-level processing such as edge

[6]The Clinton-Gore example shows that you recognize these two well-known individuals in a picture even when their facial areas are made to be identical to each other.

detection, many middle- and high-level sensory and motor behaviors in humans are learned on a co-occurrent basis from very early days of childhood and they continuously improve through later learning. The biological brain is so much determined by learning that the normal biological visual cortex is reassigned to tactile sensory functions in the case of the blind. Many researchers in the field have realized that the visual knowledge required by the human-level performance is certainly too vast and too complex to be adequately modeled by hand. Letting a machine learn autonomously by itself is probably the only way to meet the challenges in vision, speech and language. In other words, we should move from "manual labor" to "automation." Intuitively, manual modeling is hard and costly, while automation is more effective in productivity and less costly than human labor. Furthermore, it is extremely difficult, if not impossible, to build an adult human brain that has learned. It appears more reasonable to build a machine "infant brain" that can simulate, to some degree, brain's learning after the birth.

2.4.2 The machine must sense and act

The question is then how to automate this learning process. In the field of traditional artificial intelligence (AI), the main emphasis of the establishment has been symbolic problem solving (see Minsky's collection of annotated bibliography [68]). Later, due to the need of common sense knowledge in reasoning systems, some grandiose projects have been launched to manually feed reasoning rules with symbolic, common sense knowledge via computer keyboards (e.g., the CYC project [57]). The hope is that these rules and common sense knowledge are complete enough to derive all the needed knowledge. Learning in traditional AI is conducted at a symbolic level, even when autonomy is a goal (e.g., the autonomous learning with the LIVE system [92]). The machine does not have its own sensors and effectors. This has three fundamental problems.

1. The machine cannot deal with all the knowledge that is directly related to sensing and action, such as how to recognize a scene and move around it using vision.

2. Since a large amount of human symbolic knowledge, both low level and high level, is rooted deeply in sensing and action, a sensor-free machine can neither really understand nor properly use it if it is input manually. For example, even for seemingly symbolic problems such as language translation, no reasonable translation is possible without understanding the meaning of what is said (a lot of which is about sensing and action), except for simple cases.

3. The cost and time requirement is extremely high to require humans to figure out what knowledge is required for a challenging problem, to build models for content level knowledge, to input all the knowledge required, and to keep them up to date. It is even more uncertain how to make sure that all the scope of effort, cost and time will be sufficient to perform the tasks at the end of budgeted period.

Brooks emphasized the need of embodiment for developing an intelligent machine [11]. In his view, an intelligent machine must have a body to be situated in the world to sense and act. He advocated that intelligence emerges from robot's interaction with the world and from sometimes indirect interactions between its components [11]. A recent book by Hendricks-Jansen provides perspectives for embodiment and situatedness from psychology, ethology, philosophy and artificial intelligence [37]. Although embodiment, situatedness, sensing, and action have been common practices in robotics and vision communities for many years [105] [100], Brooks' work made more researchers in the related fields aware of the importance of embodiment and situatedness. In fact, it is increasingly clear that the emphasis on these important points is not only useful for symbolic AI, but also for other fields related to machine intelligence and natural intelligence.

2.4.3 Behaviors must also be learned

Hand-crafting knowledge-level rules is about modeling the world. If this is not desirable for challenging problems, how about modeling the system itself? Is it more tractable? Unfortunately, it does not seem so.

Aloimonos [2] and others advocated that modeling 3-D scenes is not always a necessary thing to do. Each vision problem should be investigated according to the purpose. Brooks 1991 [11] attributed the difficulties in vision and mobile robots to the so called SMPA (*sense-model-plan-act*) framework which started in the late 60's (see Nilsson's account [75] for a collection of the original reports). However, the behavior-based methods of Aloimonos [2] and Brooks [11] require the human programmer to hand-craft system behaviors. More recent work on robot shaping [27] also requires humans to manually specify behavior modules inside the system. In fact, the pattern recognition community and the machine learning community have had a long history of recognizing patterns without fully modeling it. Features have been used instead to classify patterns (e.g., [33], [28], [44] [81], [10], [65]). The basic difference between a pattern recognition problem and a typical computer vision problem is that the former has a controlled domain with a limited number of classes but the latter faces a much less controlled environment.

The fundamental problem of the existing SMPA approaches is not in reconstructing the world, but rather, it is the *practice of hand-crafting knowledge-level rules for one of both entities: the world and the system.* Avoiding modeling 3-D surface or abandoning the SMPA framework is not enough. For open-ended applications with unpredictable inputs, hand-crafted behaviors embedded into a programming representation must also be abandoned. No hand-crafted behavior seems to be generally applicable to the open world. Facing with tremendous difficulties in hand-crafting a large

number of behaviors, it is not clear how a behavior-based system can meet the challenges in vision, speech, language, etc, where the sensing dimensionality and the complexity of the understanding task are much higher than that of sonar sensors [62] [12]. As we explained in Section 2.4.1, not only hand-crafted knowledge models are questionable, the approach to hand-crafting system behaviors has a similar problem.

One may say that human brain is modularized. However, we should not confuse modules grouped according to sensors and effector anatomy with modules grouped according to behaviors. The former concerns the mechanism that guides learning but the latter concerns the content of what is learned. In an artificial system, behavior modules can be represented by branches of the state index tree, state clusters, and action clusters. The decomposition of these behaviors is too complex to be handled manually. The highly complex nature of the information stored in a mature human brain is termed as *society of mind* by Marvin Minsky [69]. This high complexity is the story of "what" — what in a mature brain. However, the developmental learning addresses the issue of "how" (the mechanism of that guides the developmental learning) which should be more systematic and more fundamental than the story of "what".

2.4.4 The machine must learn autonomously

Totally spoon-fed learning has a fundamental problem of scalability because (1) a large amount of knowledge and behavior must be learned, (2) the system must experience an astronomical number of instances, (3) the system must learn continuously while performing (no human being can practically handle this type of spoon-feeding work on a daily basis), (4) high-level decisions are based on so much contextual information that only the machine itself can handle (automatically).

During autonomous learning, a human teacher serves very much like a baby sitter (robot sitter in

this case), sending occasional feedback signals depending on how the living machine is doing. Later on, once the robot has learned basic communication skills through normal communication channels (such as speech and visual gesture), the robot sitter is replaced by school teachers. It appears that only the relatively low cost associated with autonomous learning is practical for meeting the requirement of many challenging problems.

2.4.5 The machine must perform multimodal learning in order to understand

Studies on humans who are born blind and deaf have demonstrated tremendous difficulties in learning very basic knowledge [122] [63]. Learning basic skills become virtually impossible with those few who are born blind, deaf, and without arms and legs. For example, a system that cannot see cannot really understand concepts related to vision (e.g., pictures and video, film, color, mirror, etc) and those concepts that are understood mainly from visual sensing (e.g., trees, mountains, birds, streets, signs, facial expressions, etc). A system that cannot hear is not able to really understand sentences related to speech and sound (e.g., the sound of music instruments, bird chirps, characteristics of a person's voice). The proverb, "a picture is worth a thousand words" vividly points out the deficiency of text (words) in describing information that are best conveyed visually.

Furthermore, understanding any single sensing modality, including vision, speech, language, and text, requires knowledge about other sensing modalities too. A system that does not live and interact with humans cannot really understand the concepts related to human emotions, characteristics and relationships (e.g., angry, happy, sympathy, care, cruel, friends, colleagues, enemy, spies, etc). In fact, a system that cannot see, cannot hear, cannot touch is deprived of the three most important sensing modalities through which a human acquires knowledge. Therefore, such

a system has a fundamental limit in understanding any human knowledge and in using such a knowledge even if it is manually fed in. Sensor-free systems like CYC have met tremendous difficulties toward applications that require understanding (such as language translation) and they are also very difficult to use due to the lack of any sensing modality for retrieval. Lack of multimodal sensing and action is a major reason to account for why existing knowledge-base systems do not really understand the knowledge they store.

2.5 Comparison with major approaches

Existing approaches to artificial intelligence fall into the following four categories:

The world-knowledge-based approaches typically require a predefined task space or world space. Human programmers manually-modeling-knowledge and spoon-feeding-knowledge (MMKSK). Researchers in each subfield have been manually developing knowledge-level theories and methods, and using them to write programs or build hardware. Then,they manually "spoon feed" knowledge into the systems at the programming level (e.g., CART [71], CYC [57] [58], lexical database, WordNet [67]). Such an approach may produce a system that appears to produce some intelligent results. However, the limitation of such systems have been recognized [11] [31]. Such a methodology requires a huge amount of human labor and it faces a fundamental limit of humans to fully model and specify the cognitive process required by challenging robotic tasks.

The behavior-based approaches avoid modeling world and instead model robot behavior. The subsumption architecture was proposed by Brooks to allow a more sophisticated behavior layer to be added to the existing primitive behavior layers [13]. Each layer is a finite state machine, with states defined and named by the programmer. The programmer is also responsible to program the state machine in each layer for a desired behavior. Thus, this approach can be characterized by

the terms "manually-modeling-behavior and hand-coding-behavior". Aloimonos [2] and others also advocated behavior-based approach for active vision.

The evolutionary approaches are motivated by evolution of biological species. The law of survival of the fittest is used to select advantageous genotypes which code the structure and/or behavior of simulated robots [61] [36]. So far, the selection process have been mostly simulated by computers using a simulated environment, due to the obvious difficulties in carrying on evolutionary process with a large number of robots and performing a long-time physical evolution [27]. The simulation method is attractive due to the low cost benefit. Also, the approach does not require the programmer to code knowledge or behavior rules. However, evolutionary approaches leave the hard task of intelligent system design to the process of random trials and environment selection, an extremely slow and costly process. Three major issues stand out: (1) The Chromosome representation for a sophisticated system. The more sophisticated the system is, the more sophisticated the chromosome is. (2) The extremely high cost of real-robot evolution when high-dimensional perception and sophisticated actions are required, such as vision, speech and language. Simulation is not sufficient for challenging vision, speech and language functionalities. Each system must experience the real-world. (3) The time, required to find a good chromosome, is on the order of a large number of generations. So far, genetic algorithms are typically used for simple environment (e.g., symbolic) and simple behaviors (e.g., symbolic) with carefully designed environment-specific and behavior-specific chromosome representation (see, e.g., Animate [120] and AutonoMouse [27]).

The learning-based approaches include all the existing learning methods, such as supervised learning and reinforcement learning. Learning approaches are typically more efficient than the corresponding evolutionary approaches since the learning mechanism of the system is hand-coded by the programmer with the former, but is either absent or has to emerge from the trial-and-

selection process with the latter. For perception of high-dimensional inputs, learning seems the only viable method. Various learning methods have produced impressive results for challenging cognition tasks involving complex modalities, such as visual recognition (e.g.,[104] [97] [72]), speech recognition (e.g., using HMM [83] [43]), vision-guided robot manipulation (e.g., [40]) and vision-guided navigation (e.g.,[116]). Supervised learning is typically more efficient than reinforcement learning. However, learning-based methods have not yet produced systems that truly understand anything. The major reasons include: (1) All the existing learning methods can only be used for a specific task at a time. For example, an neural network is trained for mapping from every image in a set of face images to a name label. In reinforcement learning using Q-learning algorithm, a task space must be given first. Then, the human designer must translate the task state to the internal representation (e.g., states) of the system model (e.g., Q-learning model). However, a system only for a particular task cannot truly understand anything. (2) A huge amount of manual labor required in translating a specific task to a learning tool. (3) A huge amount of manual labor required in training a system, including collecting data and testing the system. These reasons hinder further scaling up to more general, larger-size robotic tasks.

In actuality, a particular system may use a combination of several approaches. For example, Robot-Soar [55] combines a world-knowledge-based approach with a learning approach. It requires the human to feed knowledge about the environment and to define the task space. Then, the system learns to perform predefined tasks. The learning classifier system [27] uses a combination of reinforcement learning and genetic algorithm.

Table 2 summarizes the four existing approaches and the new developmental approach. Among the five approaches in the table, the developmental approach appears to require the least amount of human labor in terms of system design; but it requires a large number of population individuals

and a huge amount of genetic search time, which makes physical evolution for complex systems impractical. On the other hand, the knowledge-based, behavior-based, and conventional learning-based approaches all require extensive human labor in task-specific or behavior-specific design, which makes a general-purpose system impractical. The developmental approach seems to be in the middle in terms of human designer's effort and the system developmental cost (time and money). It requires human designers to properly design a general-purpose learning mechanism, but they do not need to explicitly program for the content of what is to be learned — neither for the world knowledge nor for the system behavior. It requires only one or a few physical systems to be built to learn. These physical systems take advantage the richness of intelligence in the human environment. For example, suppose that in reinforcement learning in a computer simulation environment, a punishment is given when a sequence of system actions eventually lead to a failure. The system does not know which action in the action sequence is wrong. In the developmental approach, however, the human teacher can analyze the observed system action sequence and can tell the system which action is wrong by (a) first bringing the system to the right context (e.g., lead it to the location of its bad action) and then (b) giving it a punishment. Such a very powerful tell-you-what-it-is-for mechanism can speed up learning tremendously. This type of information-rich learning environment is more powerful than the time-discounted average reward model in Q-learning and the time-average reward model in R-learning [80] in dealing with ubiquitous delayed-reward situations in learning.

3 Work Toward Living Machines

The task of developing living machines consists of two integral aspects:

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 [7]	treatment varies	genetic search
Developmental	programming	avoid modeling	general-purpose learning

First, develop the physical system, including theory, algorithm, hardware and software, for autonomous multimodal learning. In some sense, our goal is to build a machine counterpart of an animal new born, although an exact duplicate is neither possible nor necessary.

Second, teach the living machines to do things. In a sense, human beings try to "raise" and teach the machine "babies" properly so that every one of them will become successful in the task field assigned to each.

Although the first aspect is very fundamental, the success depends also very much on the second aspect — teaching. The cognitive development of a human individual relies not only on the learning mechanism of that individual, but also how he is taught. It is expected that the methodology for constructing a living machine individual and the methodology for teaching a living machine will both improve through close interactions between the two aspects.

3.1 Three phases

The success of this approach requires the multidisciplinary collaboration by academic researchers and the related industries. Therefore, it is beneficial to outline a rough sketch in terms of what we did and where we are heading. Our work at Michigan State University (MSU) has followed a three-phase plan. Phase 1: comprehensive learning; Phase 2: AA-learning; and Phase 3: daily

living.

3.1.1 Phase 1: Comprehensive learning

In Phase 1, the task is to develop a framework for basic brain functionalities such as memory store, automatic feature generation[8], self-organization, and fast associative recall. The major goal is *generality and scalability*. The generality means that the framework must be applicable to various domains of sensor-effector tasks. The scalability means that it must have a very low time complexity[9] to allow real-time learning and performance (i.e., scalable to number of learned cases). A wide variety of sensing and action tasks must be performed to verify the generality and scalability. However, learning at this phase is "spoon-fed", meaning that the learning process is not autonomous.

A lot of results in the fields of pattern recognition, computer vision and machine learning have contributed to the idea of comprehensive learning. At MSU, the comprehensive learning is represented by the work around what is called SHOSLIF to be discussed in Section 4.

3.1.2 Phase 2: AA-learning

This phase is to study the theory and the methodology development for AA-learning and to build one or more prototypes of living machines. Experiments using simulation may be conducted before

[8]We do not use the term feature selection here because it means to select from several pre-determined feature types, such as edge or area. The term feature extraction has been used for computation of selected feature type from a given image. Feature generation means automatic generation of the actual features (e.g., eigenfeatures) to be used based on learning samples.

[9]For the living machines, the time complexity is logarithmic in the number of cases learned. Thus, the exact term should be "logarithmic scalability" — logarithmic to the scale of the task.

using a real machine. In a virtual-time simulation experiment, the system behaviors can be fully measured in exact time steps.

In the real machine tests, the living machines are trained to perform certain tasks autonomously, such as moving around, grabbing things, saying simple words, and responding to spoken words, all in a fully autonomous mode and in unrestricted general settings. The tasks used in the training and testing are those that correspond to the *sensorimotor* stage of a human child (from birth to age 2), as described by Jean Piaget [35] [17] [16]. However, a machine development does not necessarily duplicate exactly human cognitive development stages. During the development, the theory and methodology for developing living machines will be modified depending on how well the living machine can learn.

Table 3 lists five integrated task groups which probably can be the target benchmarks for Phase 2 prototypes. These tasks to be performed by the living machines must be taught and tested in the AA-learning mode. If the benchmark test is successful, this will be a case where a machine really understands something[10]. By the time it has passed the test, the living machine actually has learned much more because the setting is unrestricted — much more has been seen, much more has been heard, much more has been tried, and much more has been learned. The section 5 discusses the first prototype of the living machine called SAIL being developed at MSU.

It is expected that the progress of this phase will be accelerated when more and more research groups from related disciplines collaborate for this direction of developmental approach. Right now, it is difficult to predict when phase 2 will be complete and when the next phase begins.

[10] A link from one text string to another, or a mapping from a camera input to a label is probably not understanding, since the machine does not understand the text or label.

Table 3: Benchmarks of Phase 2 for the Living Machines in *General Settings*

Task group	Benchmark
Visual recognition	Say hello with correct names of 5 human teachers when they enter the scene. Say the name of 5 toys when being asked.
Speech recognition	Understand sentences: Come. Call me. Hello! Goodbye! Wave your hand. Say hello to .... Say goodbye. What's it? Pick this. Put down. Pour water into this. Yes. No. Follow me. Watch this. Stop. Go home. I'm home. I got lost.
Speech synthesis	Respond using sentences: Hello! Goodbye! Yes. No. I'm home. I got lost. Call the names of 5 toys and 5 teachers.
Navigation	Autonomous indoor navigation without running into anything. Outdoor navigation following campus walkways and crossing streets. Follow a teacher to go, via an elevator, from a lab on the third-floor of a building to the parking lot outside the building. Return from the parking lot to the lab alone.
Hand action	Pick correctly one of the 5 toys. Put a toy down. Wave its hand when saying hello or goodbye. Place one toy on top of another. Pour a cup of water into another cup.

3.1.3 Phase 3: Daily living

This phase will start when AA-learning algorithms are well developed and the hardware prototypes are operational and reliable. Starting from this phase, the living machine enters what is similar to Piaget's preoperational stage (age 2 to 6). Again, exact duplication of human developmental stages are not necessary and impossible. Significant improvements of the living machine algorithm and software will continue throughout this phase, similar to the way operating systems are improved and upgraded now. Computers move to new levels of storage and speed; while their cost continues to fall. The new demands from living machines will stimulate the robotics industry to produce new generations of light weight, reliable, highly integrated hardware platforms for living machines.

A new emphasis in this phase is to investigate how to teach the living machines to learn things that are taught in human preschools. Since machine computes fast and is never tired of learning,

potentially they can learn faster than a human child, at least in some subjects. At this stage, communications between human teachers and the living machines become mostly visual or vocal, whichever is more convenient. Breakthroughs in vision, speech recognition, speech synthesis, language understanding, robotics, intelligent control and artificial intelligence are simultaneous at this stage. The benchmark to measure success is a standard entrance test for human pre-schoolers.

At this time, a new industry will appear. Living machines are manufactured and delivered to research institutions as experimental machines, to federal agencies for special tasks, to schools as educational material, to amusement parks as interactive attractions, to the media industry as machine personalities, and to homes for those who are physically challenged or just need a friend. The early popular version of all types of living machines is a low cost version, a software program that can be loaded onto a personal multimedia computer that has a video camera, a microphone and a speaker. At the end of this phase, the bright future of living machines is well known to general public.

3.2 Brain size and speed of the living machine

A question is naturally raised here: how much space does the living machine need? How fast can it recall from a large brain? These two important questions cannot be clearly discussed until the methods are outlined. See Section 7.2 for a discussion about the brain size and Section 7.3 for the speed issue. With the logarithmic time complexity of the living machine and the steady advance of computer storage technology, we predict that before very long, real-time living machines may have a storage size comparable with that of the human brain at a reasonable cost, although in many applications we probably do not need as much space as the human brain.

The future of the living machines will be discussed in Section 8. Next, we discuss our work in

phase 1 represented by SHOSLIF.

4 SHOSLIF

In phase 1, we developed the Self-organizing Hierarchical Optimal Subspace Learning and Inference Framework (SHOSLIF) [112] [111] [108] [109] [110]. SHOSLIF by itself is not domain-extensible. It is meant to be tested on challenging tasks in several domains individually. SHOSLIF is the predecessor of SAIL to be discussed in Section 5.

4.1 Motivations

First, we discuss the motivations behind the work of SHOSLIF.

4.1.1 Comprehensive learning

The major new concept introduced by the author during Phase 1 was the concept *comprehensive learning* which the author first presented in an NSF/ARPA sponsored workshop in 1994 [108] [112]. As illustrated in Fig. 2, the *comprehensive learning* concept consists of two basic ideas:

1. Learning must comprehensively cover the sensed world (visual, auditory, etc).

2. Learning must comprehensively cover the entire understanding system (visual, auditory, etc).

This concept was motivated from what is discussed earlier in Section 2.4. Let's briefly explain its meaning.

The first comprehensive coverage implies that we do not manually model the world. In computer vision, for example, this means that we do not model object shape. Unless we have a very much controlled world, no manually built knowledge-level model is general enough for AA-learning. The

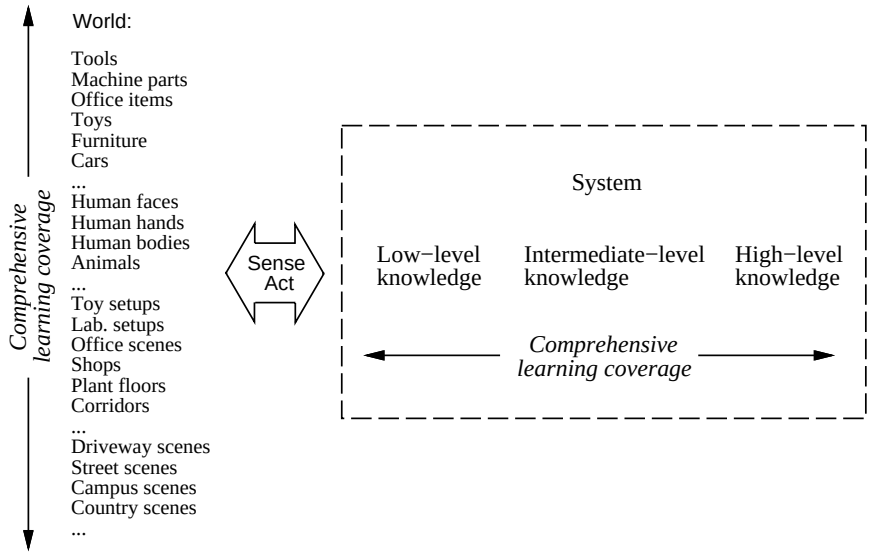


Figure 2: The concept of comprehensive learning implies that learning must comprehensively cover (1) the sensed world and (2) the understanding system (or called agent). This implies that one needs to avoid hand-crafted content-level rules for the world and the behavior rules for the system.

learning method should not assume which type of scene condition is acceptable by the system and which is not. The method must be able to learn from any scene sensed by the sensor, because there exists no automatic condition checker which can tell, given an arbitrary situation, whether the scene condition is acceptable to the method. A system is not able to operate autonomously in the real world, if it assumes conditions that it cannot verify by itself. Of course, learning does not mean perfect. The performance of a system is restricted by the sensors and actuators it is equipped with.

The second comprehensive coverage implies that we should avoid hand crafting system behaviors. In other words, hand-crafted knowledge-level rules (such as shape from shading rules, shape from contour rules, edge linking rules, collision-avoidance rules, planning rules, reasoning rules, etc) should be avoided for the programming level as much as possible. There are simply too many

behaviors to be hand-crafted effectively by humans. The astronomical number of possible combinations and coordinations of these behaviors further make hand-crafting impractical. Hand-crafted behavior rules result in brittle systems in the open-ended real world environment. Even if it is necessary that some low-level behaviors are fed into the system as "innate" behaviors, they should be well integrated into the learning mechanism of the entire system.

4.1.2 Generality and scalability

The generality (i.e., the real-world applicability) results from the comprehensive learning requirement, as stated above. The remaining issue is the efficiency. The scalability means that the method must work in real time even when the system has learned a huge number of cases. However, it is difficult to achieve both generality and scalability. A general method does not use pre-imposed special-purpose constraints and thus tends to be less efficient.

In Phase 1, we applied SHOSLIF to a variety of tasks to test its generality; and we used a large number of cases for each task to test its scalability.

4.2 Technical methods

SHOSLIF is designed for studying the conflicting goals of generality and scalability.

4.2.1 The core-shell structure

A level of SHOSLIF consists of a core and a shell, as shown in Fig. 3. The core is task independent. It has a network N as its memory, a leaner L for memory updating and retriever R for memory recall. The core serves the basic function of memory storage, recall and inference. The shell is task dependent. It is an interface between the generic core and the actual sensors and effectors. For

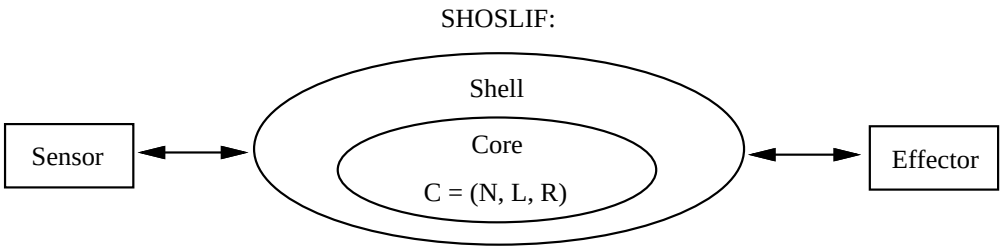


Figure 3: The SHOSLIF's core and shell, with sensors and effectors. Such a core-shell structure can be nested.

a particular task with a particular set of sensors and effectors, a shell needs to be designed which converts input data into a vector in space S, to be dealt with by the core. The output of the core is fed into the corresponding effectors by the shell. Mathematically, the SHOSLIF core approximates a high dimensional function $f : S \mapsto C$ that maps from sensor input space S to the desired output space C. This very general representation is the key to enable the core to be task independent.

4.2.2 The SHOSLIF tree

SHOSLIF was developed for the above two goals, the generality and the scalability. It must accept the high-dimensional sensory input directly and it must be fast. Specifically, it is a framework for automatically building a tree that is used to approximate the function $\mathbf{Y} = f(\mathbf{X})$ that explains the desired output $\mathbf{Y}$ given the input $\mathbf{X}$. A set of training samples $L = \{(\mathbf{X}_i, \mathbf{Y}_i) \mid i = 1, 2, ..., s\}$, where

$$\mathbf{Y}_i = f(\mathbf{X}_i).$$

The dimensionality of $\mathbf{X}$ is typically larger than the number of training samples s in the training set L. Since the high-dimensional sensory input is directly used instead of a low dimensional vector of feature measurements extracted by a feature extraction algorithm, typically the noise in sensory

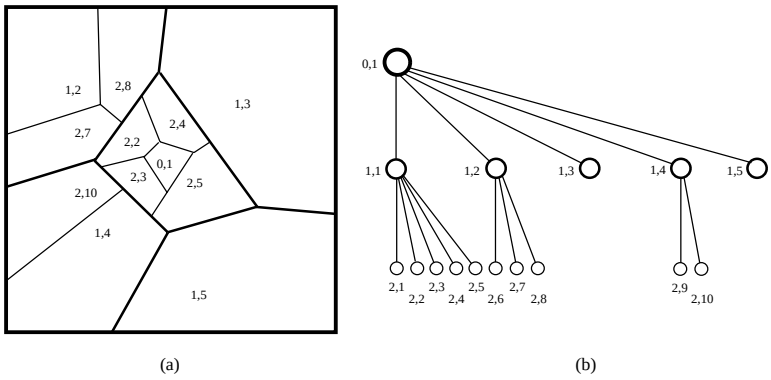


Figure 4: (a) samples in the input space marked as i, j where i is the level at which its position marks as the center of the partition cell and j is the index among the brothers in the SHOSLIF tree in (b). The leaves of the tree represent the finest partition of the space. All the samples in each leaf belong to the same class. A class is typically represented by more than one leaf. Linear boundary segments (i.e., corresponding to linear features) at the finest level are sufficient because any smooth shape can be approximated to a desired accuracy by piecewise linear boundaries.

input $\mathbf{X}$ is relatively low.

Fig. 4 shows a hierarchical space partition in the input space of $\mathbf{X}$ and its corresponding SHOSLIF tree. The SHOSLIF tree is a classification and regression tree. Therefore it shares many common characteristics with the well known tree classifiers and the regression trees in the mathematics community [10], the hierarchical clustering techniques in the pattern recognition community [28] [44] and the decision trees or induction trees in the machine learning community [81]. The major differences between the SHOSLIF tree and those traditional trees are:

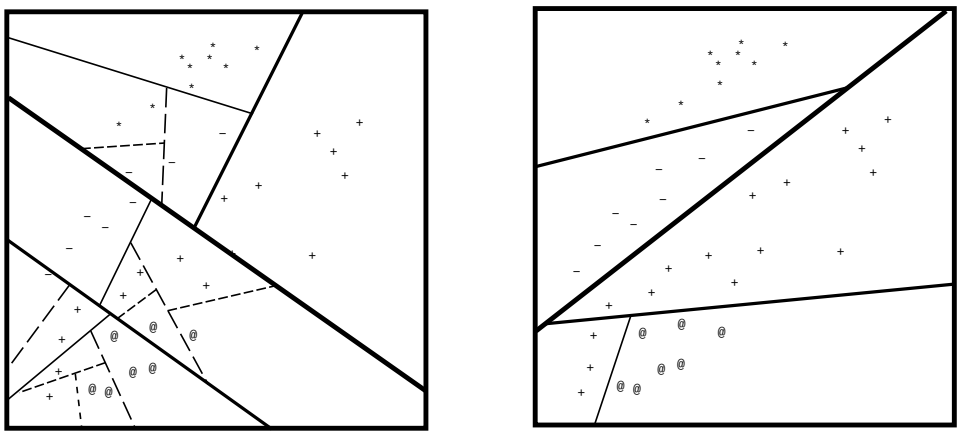
(A) The SHOSLIF automatically generates features directly from training images, while the traditional trees work on a human pre-selected set of features. This point is very crucial for the completeness of our representation.

(B) Use of PCA [49] combined with LDA [46] [33] for tree generation. The traditional trees

have been popularly univariate. That is, they use splits based on a single component of the input vector at each internal node. For example, (a) at each internal node, they search for a partition of the corresponding samples to minimize a cost function (e.g., ID3 [81] and clustering trees [44]), or (b) simply select one of the remaining unused features as the splitter (e.g., the k-d tree). Option (a) results in an exponential complexity that is way too computationally expensive for learning from high-dimensional input like images. Option (b) implies selecting each pixel as a feature, which does not work for image inputs (in the statistics literature, it generates what is called a dishonest tree [10]). There have been several studies that use multivariate splits. An early study that uses linear discriminant splitter is the work of Friedman [32], which produces a binary decision tree for every class. A recent excellent survey on these studies is written by Murthy [73]. The SHOSLIF is unique among these multivariate-split decision trees in that it must deal with such a high-dimensional input space that the number of samples is typically smaller than the dimensionality. Thus, it combines PCA with LDA at each internal node and it uses interpolation among the top-matched nodes in high dimensional space to address both classification problem (class label as output) and the regression problem (numerical vector as output).

4.2.3 Automatic generation of features

In each internal node of the SHOSLIF tree, one or several feature vectors are automatically generated to further partition the training samples. The MDF subspace is such that in that subspace, the ratio of *the between-class scatter* over *the within-class scatter* is maximized. Computationally, the MDF vectors are the eigenvector of $W^{-1}B$ associated with the largest eigenvalues, where W and B are the within- and between-class scatter matrices, respectively. In the case where class information is not available, SHOSLIF uses PCA (principal component analysis) to compute the



(a) (b)

Figure 5: (a) Hierarchical partition of the MEF binary tree. (b) Hierarchical partition of the MDF binary tree, which corresponds to a smaller tree. The symbols of the same type indicate the samples of the same class.

principal components of the sample population (which we call the most expressive features MEF in order to bring up a contrast with the MDF).

How does the system know the class label? In fact, the MDF is also suited for autonomous learning where, although the exact class is not known, the reward is given which enables a two-class discrimination at each internal node to find desired action and avoid undesired ones. At later autonomous learning, class information will be available from the living machine itself.

In the training phase, as soon as all the samples that come to a node belong to a single class, the node becomes a leaf node. Fig. 5 shows an example of SHOSLIF binary tree, for which only one feature vector is computed, resulting in a binary tree, which is very fast in retrieval since only one project needs to be computed as each internal node. As shown, the MDF gives a much smaller tree than the MEF since it can find good directions to separate classes. In reality, we explore $k > 1$ paths down the tree to get top k matches. Then the confidence is estimated from a distance-based

confidence interpolation scheme using the top k matches.

4.3 Some properties of SHOSLIF

Here we briefly describe some properties that have been established for SHOSLIF. Suppose that the learning task is to approximate a function f from its high dimensional domain to its range. We avoid a lot of mathematical equations here. In the following, the first two properties address the *generality* and the later two properties deal with the *scalability*.

Point to point correct convergence: Loosely put, as long as an image $\mathbf{X}$ has a positive probability to occur, the approximating function $\hat{f}$ represented by the SHOSLIF tree approaches the correct $f(\mathbf{X})$ as the number of learning samples increases without bound.

Functionwise correct convergence: Loosely put, if the training samples are drawn according to the real application and the function to be approximated has a bounded derivative, then the approximate function $\hat{f}$ represented by the SHOSLIF tree approaches the correct f *in the mean square sense* [11] as the number of learning samples increases without bound. The above condition does not mean that the samples must be uniformly drawn from all the possible cases, but rather, it means that one just takes the samples roughly in the way the system is used in the actual application. This theoretical result means that the system will never get stuck into a local minima and thus fails to approach the function wanted. This is a property that the artificial neural networks lack.

Rate of convergence: Let R_n be the error risk of the SHOSLIF tree and R^* be the corresponding Bayes risk (which is the smallest possible risk based on a given training set). We have proved

[11] Also in probability 1 which is stronger than the mean square convergence, as discussed in [59].

the following upper bound on R_n:

$$R_n \leq 2R^* + A(2\beta)^2 k^{2/d} (\frac{1}{n})^{2/d} \tag{3}$$

where A is the upper bound on the Jacobian of the function f to be approximated, β is the radius of the bounded domain in which f is to be approximated, k is the number of neighbors used for interpolation employed in SHOSLIF, d is the dimensionality of the feature space, and n is the number of learning samples. This result is consistent with the intuition that k-sample based interpolation is useful only for a smooth function but it slows down the approximation when the function surface is rough. When n goes to infinity, the inequality gives $\lim_{n\to\infty} R_n \leq 2R^*$, which is a well known result proved by Cover and Hart [21] for the classification task and later extended to function approximation by Cover [20]. This result gives a theoretical foundation for using the k-nearest neighbor rule since its resulting error rate is not too far from that of the best possible Bayes estimator. The Bayes estimator is impractical in our case because we do not know the actual distribution function and estimation of the distribution function in a high dimensional space is computationally very expensive, even if we impose some artificial distribution models.

Logarithmic complexity: The time complexity for retrieval from the recursive partition tree (RPT) used by SHOSLIF is $O(\log(n))$, where n is the number of samples stored as leaf nodes in the tree, which is typically smaller than the size of the training set. This result is true not only for a balanced SHOSLIF tree (guaranteed by a binary tree version of the SHOSLIF), but also true for *Bounded Unbalanced Tree* typically generated by a general version of the SHOSLIF.

Table 4: Some Tasks Tested Using SHOSLIF

SHOSLIF subproject	SHOSLIF-O (recognition)	SHOSLIF-M (spatiotemporal)	SHOSLIF-N (mobile robot)	SHOSLIF-R (robot arm)	SHOSLIF-S (speech)
Spatial recognition	X	X	X	X	X
Temporal recognition		X	X	X	X
Image segmentation		X		X	
Prediction		X	X	X	
Visual attention		X		X	
Sensorimotor			X	X	
Incremental learning			X		X
On-line learning			X		X

As we know, the above theoretical results gave only some insights into the nature of the task. Evaluating actual error rates, some of them were quoted in this proposal, is a more practical way for evaluating actual algorithms.

4.4 Functionalities tested for the SHOSLIF

The demonstration of generality and scalability is a very challenging task. It requires us to test SHOSLIF method for a wide array of tasks. We selected several domains of challenging vision tasks to test SHOSLIF theory and performance. The selection of the vision tasks was determined in such a way that they cover major functionalities that must be implemented in Phase 1. Since the living machine requires also speech recognition capability, we have also selected the speech recognition domain. Table 4 lists the representative tasks that we have selected as test domains for SHOSLIF and the related functionalities that have been tested if applicable.

In the following, a summary of each SHOSLIF subproject is presented. Due to the unified core-shell structure of SHOSLIF, the programming for each subproject was systematic. Basically an interface needs to be developed as a shell for each subproject.

4.5 SHOSLIF-O: Face and Object recognition

This project is to use the SHOSLIF method to recognize a large number of objects from their appearance. The SHOSLIF approach is different from other conventional approaches to recognition in that it deals with real-world images without imposing shape rules or shape models on the scene environment. But rather, it uses a general learning method to make the system learn how to recognize a large number of objects under complex variations. It copes with critical issues associated with such a challenging task, including automatic feature generation, automatic visual information self-organization, generalization for object shape variation (including size, position and orientation), decision optimality, representation efficiency, and efficient indexing into a large database.

In order to test the system using a database that is as large as possible, we combined several different databases for training and testing: (1) MSU face database (38 individuals); (2) FERET face database (303 individuals); (3) MIT face database (16 individuals); (4) Weizmman face database (29 individuals); (5) MSU general object database (526 classes). Fig. 6 shows some examples of face and object images used for recognition. Table 5 summarizes the result obtained. The training images were drawn at random from the pool of available images, with the remaining images serving as a disjoint set of test images.

For training efficiency, SHOSLIF is trained to recognize canonical face views only. In a canonical face view, the position, size and orientation of the face are all roughly equal to a set of predefined

Table 5: Experimental Results for Face and Object Recognition from Well-Framed Views

Type	Training set	Training classes	Test set	Top 1 correct	Top 15 correct
Face	1042 images	384 individuals	246 images	95.5%	97.6%
General	1316 images	526 classes	298 images	95.0%	99.0%

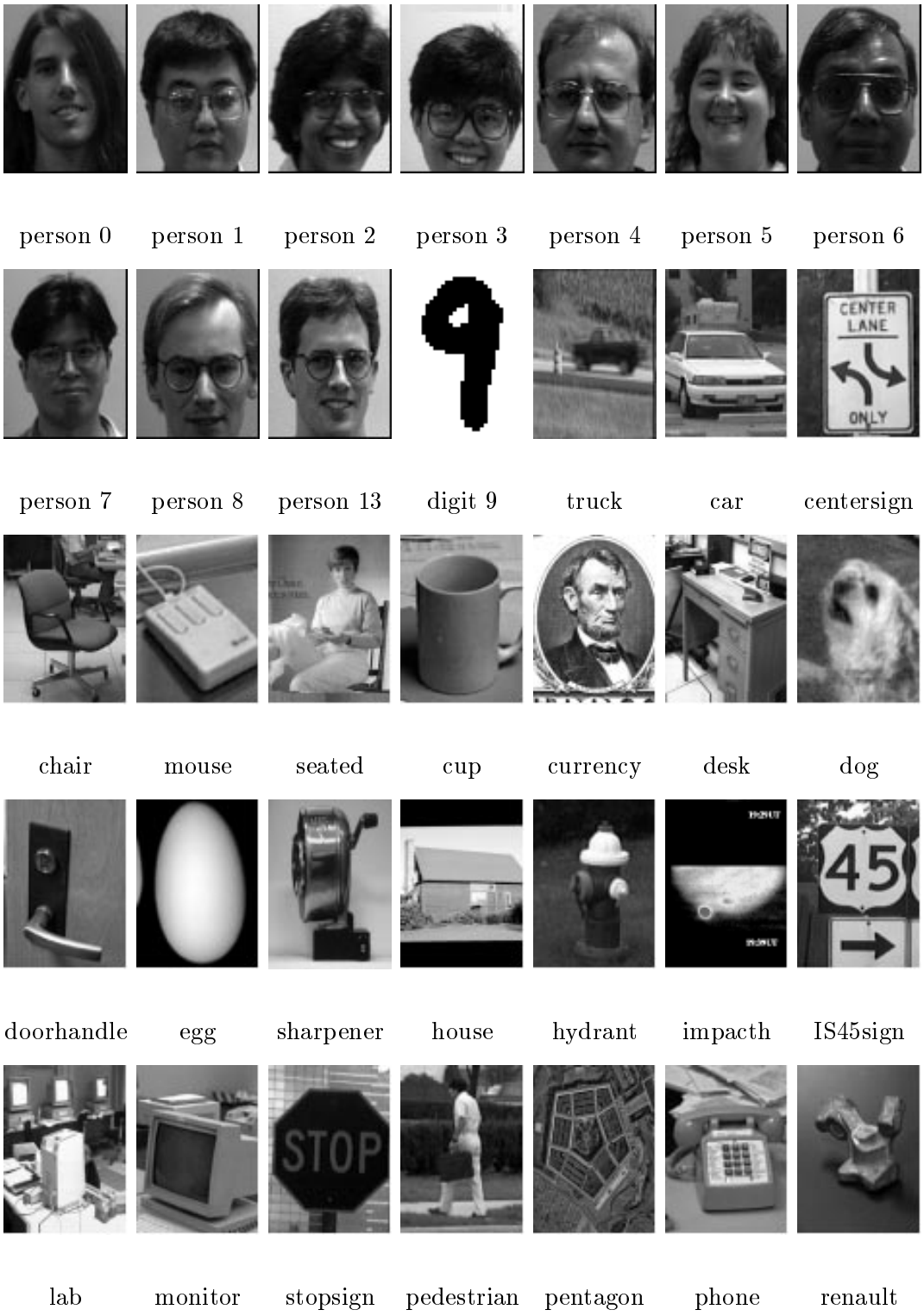


Figure 6: Some examples of face and object images used for recognition.

values. In order to deal with some variation in the canonical views, the system was also trained to handle a limited amount of variation in size, position and 3-D orientation within a certain range. We trained the system using samples generated from the original training samples to randomly vary in (a) 30% of size, (b) positional shift of 20% of size; (c) 3D face orientation by about 45 degrees and testing with 22.5 degrees. The training and test data sizes are similar to that in Table 5. The top 1 and top 10 correct recognition rates were, respectively, (a) 93.3% and 98.9%, (b) 93.1% and 96.6%, (c) 78.9% and 89.4%. The data reported here were extracted from [96]. More details are available in [97] and [98].

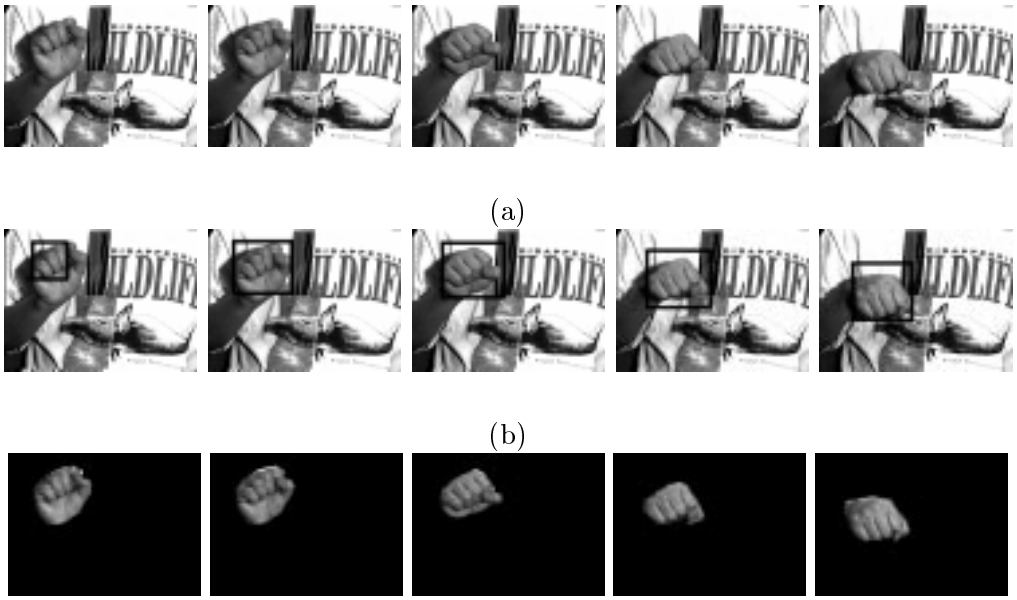
Directly treating an image as an appearance vector and recognizing the object by applying statistical methods to such a vector space is often called appearance-based approach. Research for such an approach has become very active. Examples of such appearance-based approach for face and object recognition include Turk & Pentland [104], Murase & Nayar [72], Etemad & Chellappa [30], Swets & Weng [97], and Belhumeur *et al.* [7]. The SHOSLIF-O is one of few the appearance-based face-and-object recognition algorithms that use a tree structure for a logarithmic time complexity.

It is worth noting that a successful recognition by the appearance-based approach requires that a well-framed image view is given in which the object of interest is centered reasonably well with an appropriate size, as shown in Fig. 6. It appears that general-purpose goal-directed attention selection for general scenes is a very challenging high-level task that cannot be effectively addressed without truly understanding the scenes. Currently, mechanical fixed-order image scanning has been attempted. For example, in Cresceptron [114] [112], many attention images are obtained by systematically scanning the image using different window sizes along a grid pattern of fixation points. Within the attention image in which the object is recognized, the object is segmented by Cresceptron from the background using back-projected edges that have contributed to the

recognition. Exhaustive, pixel-to-pixel scanning using an attention window has also been used, especially for the tasks of face detection. Some recent examples for face detection include Sung & Poggio [95], Colmenarez & Huang [19], Rowley, Baluja & Kanade [86] and Moghaddam & Pentland [70]. Motion information and prediction from partial-view to global-contour can also assist segmentation, as explained in the following section.

4.6 SHOSLIF-M: Motion event recognition

For spatiotemporal recognition, we selected a challenge task: recognition of hand signs from American Sign Language. Recently, there has been a significant amount of research on vision-based hand-sign recognition from images. Compared with other resent studies on hand-sign recognition from image sequences [24, 9, 94, 56, 50, 102], the SHOSLIF-M work [22] [23] has the following characteristics: (1) The capability to segment a detailed hand (a complex articulated object) from a very complex background as shown in Fig. 7. With this capability, we can significantly reduce the constraint on what kind of clothes that the signer can wear. This is accomplished through (a) using motion information to reduce the area of attention; (b) using a learning-based prediction-and-verification scheme which predicts the global contour from a local view that partially covers the object of interest. Thus the recognition result is completely independent of the background. In contrast, using a fixed-shape attention window, it is required that the background covered by a local view does not affect the local matching significantly, such as the jets scheme by Malsburg *at al.* [102]. (2) The logarithmic sign-retrieval time complexity $O(\log(n))$. (3) The system distinguishes a large number (over 140) of hand shape classes, the largest among the existing works on static hand shape recognition from images (a list can be found from Huang and Pavlovic's recent survey [38] and the workshop proceedings in which that survey was published). (4) The relatively large



(a)

(b)

(c)

Figure 7: (a) An example of many hand-sign sequences. It means "yes". (b) The results of motion-based attention mask found, shown with a bounding (dark) rectangular window. Notice that motion-based segmentation alone is not sufficient for hand sign recognition. Without a detailed shape information of the hand, reliable hand sign recognition is not possible with a large number of hand-sign classes. (c) The result of final segmentation is shown with the background automatically masked off. Such a detailed and accurate segmentation is crucial to the success of hand sign recognition with a large number of classes and vice versa.

number of hand-signs among the existing works on *handwear-free* moving hand-sign recognition (see also a recent survey article by Huang & Pavlovic [38]).

4.7 SHOSLIF-N: Autonomous navigation

Autonomous vision-guided navigation is an example for vision-guided effector control. The input to the system is the current image from the camera and the output from the system is the next control vector that specifies the heading direction and the speed. SHOSLIF-N faces a challenging

Figure 8: The mobile robot 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.

task of performing real-time indoor navigation using only a single video camera without using any other sensors. For learning, it must learn on-line, incrementally, in real-time. SHOSLIF-N uses the incremental version of the SHOSLIF core [116], which builds a SHOSLIF tree incrementally by updating the tree after receiving each training sample. It is a good example to study how real-time on-line learning might be achieved without need of special image processing hardware, thanks to the extremely low (logarithmic) time complexity of the SHOSLIF.

During the learning, using a joystick, the human teacher controls the robot on-line to navigate along the desired path by updating the control signals in terms of speed and heading direction. The system digitizes the current image frame and links it with the current control signal to form a training sample, which is used to train the SHOSLIF tree. To construct a lean, condensed tree without overlearning, the SHOSLIF tree rejects a training sample when the current tree outputs a control signal vector that is the same (according to the accuracy required) as the human's control signal vector, without learning the current training sample. When almost all the recent training samples are rejected, the system is almost fully trained and ready to perform. The on-line learning with both tree retrieval and update runs at 5 Hz. The real-time performance, with a tree retrieval speed of 7 Hz, is accomplished by an on-board Sun SPARC-1 workstation and a SunVideo image

digitizer, without any other special-purpose image processing hardware. The trained system successfully navigated along the hallways of our Engineering Building, making turns and going through hallway doors. It was not confused by passers-by in the hallway because it looks at the entire image and uses the most useful features (MEF or MDF), instead of tracking floor edges which can be easily occluded by human traffic.

A significant amount of vision-based outdoor road following has been conducted (e.g., Dickmanns' group [25], CMU Navlab [101], Martin Marida ALV [105], CMU ALVINN [78], and the work at Maryland University [85]). Outdoor navigation must deal with a large degree of light change and whether change. Indoor invigation, in contrast, must deal with the lack of large high contrast regions in typical indoor scenes (e.g., Meng & Kak [64] and Weng & Chen [116]), which pose a severe local minima problem for techniques that use iterative minimization in training.

4.7.1 The Speed-up of SHOSLIF tree

To indicate the scalability performance of the SHOSLIF and the effect of the hierarchical tree, Fig. 6 shows the computer times for SHOSLIF (tree version) and the corresponding two flat versions. A flat version uses goes through all the training samples one by one to find the nearest neighbor. The MEF flat version does such a linear search in the MEF space constructed from all the training samples. The image-space version does so in the original image Euclidean space. A total of 2850 images from various hallway sections in the Engineering Building of MSU were used for training. The data in the table show the use of MEF tree can greatly speed up the retrieval and that real-time navigation can be achieved. The computation times were recorded when the programs ran on a SUN SPARC-10 computer.

Table 6: Computer Time Difference between the Flat and the Tree Versions

Time per retrieval	MEF tree	Flat version	
		in MEF space	in image space
Time per retrieval (in seconds)	0.028	0.74	2.9
Slow down w.r.t MEF tree version	-	26.7 times	103.0 times

Table 7: MDF Results in a Smaller Tree

Tree type	MDF	MEF
Total number of nodes	69	635

4.7.2 MDF results in a smaller tree

For comparison purpose, two types of trees have been experimented with, MEF RPT and MDF RPT. The former uses MEF and the latter uses MDF in each internal node of the respective tree. Both trees used the same 318 learning images, 210 from the straight hallway and 108 from the corner. As presented in Table 7, the MDF tree has only a total of 69 nodes, with only 35 leaf nodes; while MEF tree has a total of 635 nodes, with 318 leaf nodes. Fig. 5 explained why MDF can give a smaller tree. Note that the timing data shown in Table 6 is for MEF tree, which is larger than the MDF tree. An MDF tree is typically much faster. The results quoted here are extracted from [115] [116]) where more detail is available.

4.8 SHOSLIF-R: Vision-guided robot manipulator

This is for sensorimotor coordination and task-sequence learning, SHOSLIF-O was used as the object locator and recognizer. A SHOSLIF-R network was automatically built from training temporal sensor-guided control sequences [40]. The input to the SHOSLIF-R network is the image position of the objects (from SHOSLIF-o) and the index of the task from the human teacher. The output of

the SHOSLIF-R network is the incremental values of the six joint angles of the robot manipulator.

As reported in [41], five actions were learned interactively at several places in the work space. Tests were done randomly in any place in the workspace. The success rate was 100% for all the actions, except that the liquid was spilled partially 20% of the time during the pouring action. More training can improve the pouring accuracy. This is an example of learning by doing, instead of explicit modeling. Modeling the dynamics of poured liquid is not possible because there are too many unknown and unobservable parameters in fluid dynamics. As far as we know, no other published systems exist that can perform the task of pouring water into a cup using vision. The reader is referred to [41] for more detail.

Several robotics groups (e.g., [48] and [54]) have recently published works in which a robot manipulator can repeat the action sequence from human's demonstration using a data glove. Case-specific features and decision rules are written into their programs which the algorithm will use to identify the sequence of actions (such as "when the speed of the hand is smaller than certain number, do the following ... "). SHOSLIF-R is fundamentally different from those works in that SHOSLIF-R does not contain any case-specific rules (hand-crafted knowledge-level rules). Specifically, SHOSLIF core does not contain any knowledge-level rules and the SHOSLIF-R shell contains only the arm hardware specification, i.e., the anatomy (e.g., degree of freedom of the hand) instead of knowledge-level rules. Thus, it is, in principle, applicable to any robot manipulator task. SHOSLIF-R is a robot manipulator learning system that can learn to perform tasks through interactive learning without hand-crafting any knowledge-level rules.

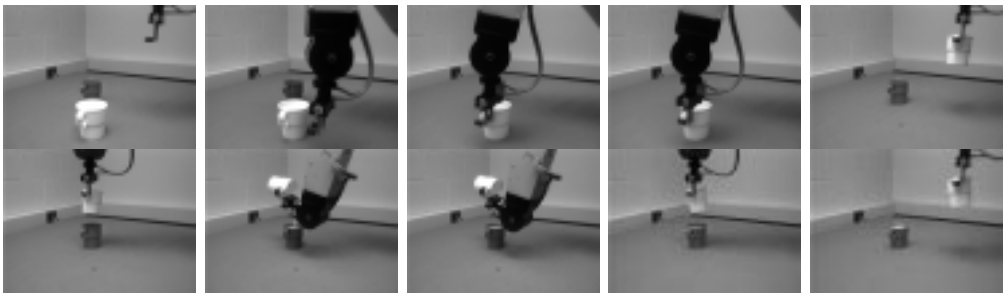


Figure 9: A demonstration of various actions learned: approaching the handle of cup A, picking the cut up, moving to top of cup B, pouring, and putting on table.

4.9 SHOSLIF-S: Speech recognition

The objective of this study [14] was to test the feasibility of using SHOSLIF for isolated spoken word recognition. It was performed as a class project in a graduate class. A spectrum feature vector of each isolated world is used as input to the SHOSLIF. The silence is used to segment each isolatedly pronounced word. The output of the SHOSLIF is the class label. An incremental learning version [116] of the SHOSLIF was used for the learning task. Due to the limited available time during the single-semester class, only a small number of training samples were collected. The preliminary experiment was performed with 10 spoken words from "zero" to "nine". The speaker-dependent testing reached 90% accuracy among 20 speakers, each word was trained with only one training sample and tested with 4 different instances.

The fully dynamic speech recognition requires the recurrent version SAIL. The recurrent version is expected to be able to learn to handle time warping, coarticulation, temporal acceleration, and pause that are very common in the real-world speech.

4.10 Cresceptron: the predecessor of the SHOSLIF

Our work along this line can be traced back to early 1990 when he conceptualized Cresceptron for general, open-ended, sensing-based learning. Cresceptron [114] [112] is the predecessor of the SHOSLIF. Cresceptron is a system that is capable of learning directly from natural images and performing the task of general recognition *and* segmentation from images of the complex real world, virtually without limiting the type of objects that the system can deal with. It has been tested for recognizing and segmenting human faces and other objects from complex backgrounds. It addressed the issue of self-organizing dynamically by growing the system on-line according to inputs. Although Cresceptron has a very high generality, it does not attempt to solve scalability. Its successor SHOSLIF solved both generality and scalability.

5 SAIL

SAIL (Self-organizing, Autonomous, Incremental Learner) is a living machine under construction at Michigan State University. Its goal is to realize AA-learning. This section discusses some basic issues of AA-learning and some design issues of SAIL. Since SAIL is an ongoing project, its design is expected to be constantly modified, refined, and improved in the future.

5.1 System overview

As discussed previously, the representation must be semantics-free. The program-level representation should not be constrained by, or embedded with, hand-crafted knowledge-level world models or system behaviors. Thus, the system design of SAIL should be conducted at signal level, instead of knowledge or content level. Fig. 10 gives a schematic illustration of the system architecture.

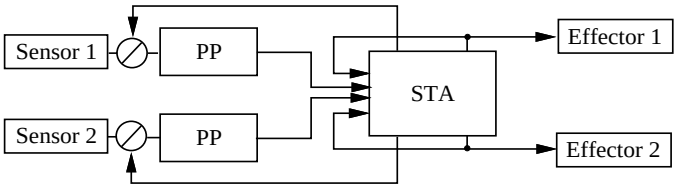


Figure 10: A schematic illustration of the coarse architecture of the presented system SAIL. A circle represents an attention selector. It is also an effector. PP: preprocessor. STA: spatiotemporal associator.

The preprocessor performs some transformations from input, such as intensity normalization, automatic gain (contrast) control, filtering, etc.

The spatial temporal associator (STA) is the "brain" of the system. It consists of automatically generated levels. The number of levels depends on the maturity of the system.

5.2 Single-level formulation

At each level, the system operates in the following way. At each time instant t, its current internal mental status is represented by its state $s(t) \in S$. It accepts sensory input $x(t) \in X$, which contains both exteroceptive sensors (e.g., visual and auditory), proprioceptive sensors (e.g., effector position) and interoceptive sensors (e.g., internal clock). As shown in Fig. 10, input $x(t)$ at the lowest level include both sensory input, effector position and possibly the internal time counts. First, assume that the system is deterministic. At each time instant t, the system accepts an input $x(t) \in X$ while it is at state $s(t) \in S$. Then, it outputs action $a(t) \in A$ and enters a new state $s(t+1) \in S$. Without loss generality, we assume that the time is discrete and hence $t+1$ is used as the next time instant. In other words, the machine refreshing cycle time is considered as a unit here. With such a deterministic system, the system state transition is represented by a time-varying function

$f_t : X \times S \mapsto S$:

$$s(t+1) = f_t(x(t), s(t)) \tag{4}$$

The state $s(t+1)$ represents what the system understands from the current input and the current mental status at this level. Then, it figures out what it should do, based on what it understands represented by $s(t+1)$. Symbolically, the action function is represented by function $h : S \mapsto A$:

$$a(t+1) = h(s(t+1)) \tag{5}$$

where A is the control signal space of all its effectors.

In order to model the uncertainty, our system is in fact nondeterministic. Thus, the system transition function is modeled by the conditional probability:

$$P(s(t+1) = s' \mid x(t) = x, s(t) = s) \tag{6}$$

where $x \in X$, $s \in S$, $s' \in S$, $a \in A$, and P denotes the probability. The action function is modeled by the conditional probability:

$$P(a(t+1) = a \mid s(t+1) = s) \tag{7}$$

The above probability depends on the experience and the reward associated with the experience.

We can see immediately from this notation that it resembles a Markov decision process [80]. However, it is more general. For this, we need a more careful examination.

First, A here is an Euclidean space, which contains infinitely many (uncountably many) elements. For a typical Markov decision processes, however, A is a finite set of legal actions. In other words, we have some distance metric defined for the elements in A. We definitely want to use this property to enable the machine to try something it has never tried for, e.g., sensorimotor refinement.

Another important issue here is the space S. It should contain certain information about the history. How should we define the state space S? In Markov decision processes, S is a predefined, finite set of elements, which accounts for a major reason why general-purpose learning fails from here. [12] The human-defined set S for states incorporates a lot of human knowledge. These symbolic states so defined are very artificial and no appropriate distance metric is defined on the finite set S. The definition of a state is not as trivial as it appears. For example, a state can be defined to record the history. Suppose that $s(1)$ and $s(2)$ are the states at time $t = 1$ and $t = 2$, respectively. One may define $s'(2) = (s(1), s(2))$ so that $s'(2)$ contains not just the current state $s(2)$, but also the history $s(1)$! Using this trick, can we use first-order Markov process notation to define Markov process of any order? Unfortunately, this is not that simple because the resulting S would become an infinite set of discrete symbols [13], which a practical Markov process does not allow. More complications arise from this too, such as the difficulties in estimating the transition probability.

In SAIL, the system states in S must be defined automatically. We define S to have the same dimensionality as $X \times \mathrm{R}(S)$, where $R(S)$ is a dimension reduction operator. For example, if each $a \in S$ is a 128×128-pixel image, the dimension reduction reduces a to a 64×64-pixel image b through neighboring pixel averaging. Given sensory input $x(t)$ when the current system is at state $s(t)$, the resulting next state, at the lowest level 0, can be simply to record the new situation: $s(t+1) = (x(t), s'(t))$, where $s'(t)$ is a dimension reduced version of $s(t)$. For example, if X has a dimensionality of 9000, we set the dimensionality of S to be 18000. Thus, the dimensionality

[12] It takes a human being to understand a particular task, using his knowledge to translate the task into the Markov decision process, including defining states, legal actions, etc. This is a machine whose "brain" is not closed. It requires humans to dictate its brain states given each particular task. This way, the machine is never able to learn on its own.

[13] This is because such compounding $s'(2) = (s(1), s(2))$ becomes endless with the progress of time. Each compounding defines a new symbol.

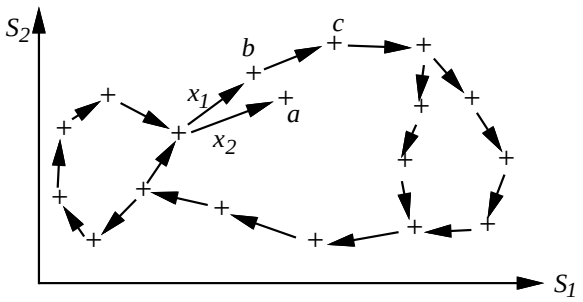


Figure 11: Representation of state enables generalization across states. Here, the high dimensional space S is illustrated by a 2-D space only. a is a newly generated state, which does not have any transition experience. The existing state b is the nearest neighbor of a in the state space S. The transition path, from b to c, that is learned by b can be used for figuring out how to act at the newly generated state a at the absence of guidance.

reduction is to reduce from 18000-dimension to 9000-dimension. With a zero-vector initial state $s'(0)$ at time $t = 0$, we can define the first state at time $t = 1$ to be $s(1) = (x(1), s'(0))$. This explains how the state can be defined automatically based the current state and current sensory input. Thus, potentially, the number of possible states is infinite. We will discuss how to automatically self-organize the states.

Since S is a high-dimensional Euclidean space, we can naturally define distance between states as their Euclidean distance. This enables our SAIL model to generalize across states which a conventional Markov model can accomplish. The idea is illustrated in Fig. 11. When a new state a is automatically generated, the current system is at this state. There is no transition probability learned for this state since it is new. With the Markov model, the system will not be able to perform from this new state a [14]. However, our distance definition (e.g., Euclidean distance) in the state space S allows our model to find the nearest matched existing state b that has a following

[14] It is worth noting that there are a lot of problems with HMM when the number of training samples is small.

state c. Thus, the predicted state from a is c as specified by $f(x, b)$ in Equation (4). Similarly, the appropriate action at this new state a can also be generated by $h(b)$ in Equation (5). In other words, generalization of transition pattern and behavior can be realized based on a distance metric in S without requiring a very large amount of data for training every possible transition among states.

Due to the dimension reduction in the representation of state $s \in S$, the new state has information about the previous state information. Thus, the single-level model in SAIL is not strictly Markov.

For generality, the system must perform for any input $x \in X$ and any possible $s \in S$. Thus, the sets of possible input x and possible state s are both infinite. However, the number of possible states at each time is finite, and it changes dynamically in the SAIL level model. When $x(t)$ changes with time, $s(t)$ changes accordingly with time, generating a trajectory in S space, as shown in Fig 11. When SAIL is nondeterministic, SAIL keeps track of $k > 1$ most probable trajectories in S.

We define a notion called *state-dictatability.*

Definition 2 *A system is state-nondictatable if the trainer of the system is not allowed to dictate the value of state in the system at any time during training.*

Since the state is very much related to the content of what the system learned, state-nondictatability means that the trainer is relieved from the intractable task of coping with content-level representation. He or she can shape the system behavior through its sensors (e.g., presenting examples) and effectors (e.g., imposing actions) but he or she cannot directly set the "brain" of system. This is a very important concept in automating learning.

A subspace $H \subset X$ is called *hardwired reward space* if the system has predefined preference for

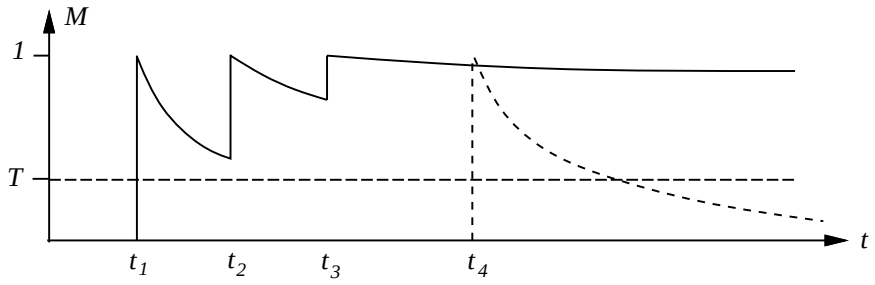


Figure 12: Update of memory trace M through time t. The solid curve represents an element which is visited often enough to be kept. The dashed curve indicates an element that is not visited often enough and thus, it falls below the threshold T before being visited again.

the elements in H. For example, we can define $X = X_1 \times H$, where $H = \{x \mid -1 \leq x \leq 1\}$. The system is such that it "likes" positive values in H (appetitive stimulus) and "hates" negative values in H (aversive stimulus). Thus, extreme pleasure, pleasure, neutral, pain, extreme pain can be represented by 1.0, 0.5, 0, -0.5 and -1.0, respectively. The *hardwired reward space* is used to enable reinforcement learning by allowing system to explore on its own while occasionally giving it some appropriate reward or punishment according to its performance.

5.3 Forgetting

Due to a finite memory space, the system cannot remember all the spatiotemporal events that it has come across. In fact, it should forget for generalization. Many associations must be forgotten. The utilization of association is indicated by *cooccurrence frequency* and *occurrence internals*.

Let us consider a memory element, a node or a link, which will be used in our system for memorizing an association it represents. Each element has a memory residual register whose updating curve is shown in Fig. 12 which resembles what we know about human memory characteristics [5] [42].

Each visit to the same element makes the trace to be reset to 1 and then the curve declines using a next slower speed. For example, we can define a series of memory *fade factors* $\alpha_1 < \alpha_2 < ... < \alpha_m \approx 1$. α_i is used for an element 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 (refresh) elapsed since the last visit to the element. Thus, we do not need to visit all the elements at every system cycle. When an element is visited, its memory trace is updated first from what remains from the last visit. If the memory trace falls below the designated threshold, it should be deleted and so it is marked as to-be-deleted. If what is deleted is more than a single element (i.e., 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."

The memory fade factor is also affected by the score level it receives. An extreme score, near 0 or 1, will result in smaller memory fade factors than a score 0.5, which means "doing OK and keep going." Thus, when the score is 0.5, the memory fade will be faster, so that many details associated with normal routine operations are forgotten quickly.

The memory fade factors control the relative speed of learning and forgetting so that they can keep in pace while growing the memory with an appropriate speed over time.

5.4 Spatiotemporal clustering

The state space S has a finite dimensionality, but the number of elements generated in S through time is unbounded. The objective of spatiotemporal clustering is to form primitive clusters (P-clusters) in S using not just spatial information, but also temporal information. A P-cluster consists of a number of prototypes each being represented by a leaf node. The P-cluster will be used as state at every level of the system.

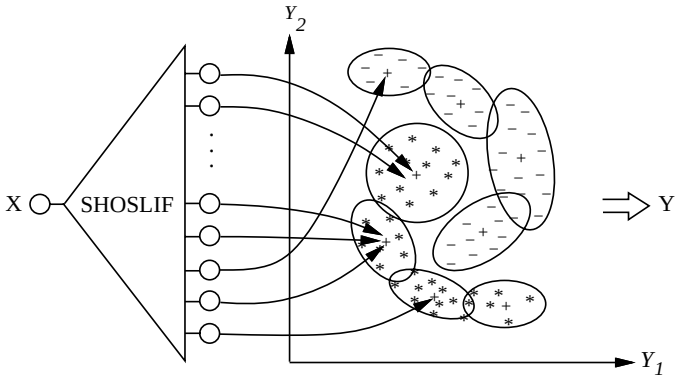


Figure 13: A schematic illustration of the temporal adjacency cluster. Each region in Y space represents a P-cluster. A dashed curve is used to roughly indicate the samples that fall into the cluster. The circles to the right of the SHOSLIF tree are the leaf nodes of the tree. Each $-$,$=$ or $*$ sign indicates the relative positions of leaf nodes (δ-prototype) in the input space to the SHOSLIF tree. Two different signs representing two semantically different input sequences. Each leaf node in the SHOSLIF tree has a pointer to the Y space, which points to the center of the corresponding P-cluster or close to it after a significant amount of gravity pulling and merging. Y is shown here as 2-D for visualization, but it is typically of the same dimensionality as X or with a reduced dimension by a factor of dimension reduction.

At level 0, our goal is to generate clusters of inputs only. Thus, the state $s(t)$ at level 0 depends only on input $x(t)$ but not $s(t)$, as a special case of (4).

Fig. 13 shows a schematic illustration of the temporal-adjacency cluster. The cluster represents a mapping $g : X \mapsto Y$ which maps from input space X to output space Y. X is the sensory input space of a particular sensor and Y is the output space containing P-clusters. Given x, $y = g(x)$ is the image of x, representing the corresponding P-cluster. In order to preserve the necessary topology in X, Y has the same dimensionality as X or has a reduced dimensionality. If X has a dimensionality $n \times n$ corresponding to an image space of $n \times n$ pixels, then $Y = R^{m \times m}$, where R is the set of all real numbers. The ratio of dimensionality difference $(n \times n)/(m \times m)$, is called the

factor of dimensionality reduction (FDR). Typically FDR=4. This reduction of dimensionality is reasonable because all we need is to have Y space to roughly keep the topology of X space. When a δ-prototype is first created, it is represented by a single training sample x_i. We let its image y be the same as x_i, $y_i = g(x_i) = x_i$ (with FDR taken into account if $FDR > 1$. (This is a short term memory effect.) Later successful matching with x_i will cause y_i to be pulled toward the Y vectors of its neighbors in X.

Consider an input stream $x = \{x(0), x(2), x(3), ..., x(t), ...\}$. where each frame $x(i)$ is in X and i is the time index. The stream x forms a trajectory in $X \times T$ space, where $T = \{0, 1, 2...t...\}$ represents the set of time indexes. We need to chunk the trajectory into segments of some appropriate length of events. Each segment corresponds to a P-cluster. Two frames $x(l)$ and $x(m)$ are *temporally adjacent* if one is followed immediately by another within a time window w, i.e., $0 < |l - m| \leq w$, where w is the adjacency window (default is 1). Given each frame $x(t)$ at time t, its top k matched prototypes are considered. The top match gives its image y. Among top k matched δ prototypes with a sufficient matching confidence, if two δ-prototypes are less than σ ($\sigma > \delta$) distance apart in X space and they are temporally-adjacent, the images of these two δ-prototypes are pulled together using a simulated force, where the number of visits of each prototypes is the "mass". Thus, among the temporally adjacent δ-prototypes with a σ radius in X, those mostly often visited prototypes tends to become the center of the cluster. When the y images of two prototypes are very close, their y images become one. Merged y vectors form the center of the P-cluster. Merged centers of y vectors have a higher mass. It can quickly pull nearby y vectors and merge them. This speeds up the clustering speed. To avoid two neighboring P-clusters to be slowly pulled together, a maturity schedule is applied so that P-clusters that are mature enough will no longer be moved — an effect of long term memory. Fig. 14 shows that the pointers emitting

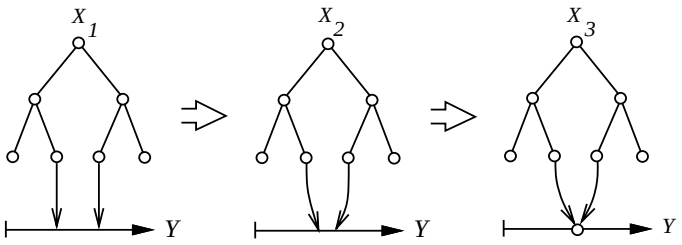


Figure 14: The effect of pulling in spatiotemporal clustering. The SHOSLIF tree is repeatedly visited by $x_1 \approx x_2 \approx x_3$ that are nearby in spatiotemporal domain. The two leaf nodes are among the top k matched leaf nodes, although they are separated early in the tree. Consequently, their Y vectors are merged when they are sufficiently close and the corresponding prototypes of the two leaf nodes belong to the same P-cluster.

from leaf nodes gradually point to the center of the P-cluster in Y space.

Note that two inputs x_i and x_j will not have the same image (state) if they are within σ distance but never occur one after another. In other words, the P-clusters are clusters of spatiotemporal clusters, but δ-prototypes in SHOSLIF leaf nodes are spatial-only prototypes and their spatial radius are typically much smaller than that of a spatiotemporal cluster.

When the spatiotemporal clustering is applied to input directly, it produces the lowest-level discrete state: level 0. The technique of spatiotemporal clustering is also used for forming states in the higher levels to be discussed in the following sections.

The techniques used here have a close relationship with learning vector quantizer (LVQ) [51] [52] and k-mean clustering [44]. The differences are (1) temporal concept; (2) unknown number of clusters; (3) the nearest neighbors must be found quickly and thus the SHOSLIF tree is used; (4) learning is incremental.

5.5 Automatic spatiotemporal level building

Each state at level i, $i = 0, 1, 2, 3...$, is a cluster at level i. Such a cluster is called level-i state, which is formed using the spatiotemporal clustering explained above.

Hidden Markov model (HMM, also called partially observable Markov model, POMM) has been successfully used in speech recognition [82] [43] and also in computer vision (e.g., [76] [94]) when what to be recognized is a temporal event. In temporal domain, the system must deal with time warping and segmentation. Typically, the HMM is used in the following way. A HMM is trained to recognize a predefined sequence class, such as a word or an action. A different sequence class uses a different HMM. Given an input sequence, multiple HMMs are being run in parallel. At each time instant, each HMM reports its accumulated probability of the current partial sequence fed into the system. A high probability from HMM indicates that a complete sequence has passed through for that class. Due to the probabilistic model of state transition, HMM can deal with time warping and segmentation effectively. Level building has been used to build short units into long units, such as linking digits to form numbers and linking words to form phrases [83] [43]. The task of level building involves a great amount of manual data preparation but uses basically the same HMM structure. Little attention has been paid to automatic level building.

Fig. 15 gives an illustration of time warping and the basic idea of level building. The probabilistic model at each level is a HMM. The model is hidden because the system is not completely sure which current state is the best state in the long run to generate actions that receive the best reward. A higher level HMM accepts inputs from the preceding lower level HMM, and takes care of a sequence that covers longer state transitions.

In manual level building in the speech community, each possible combination of concatenation

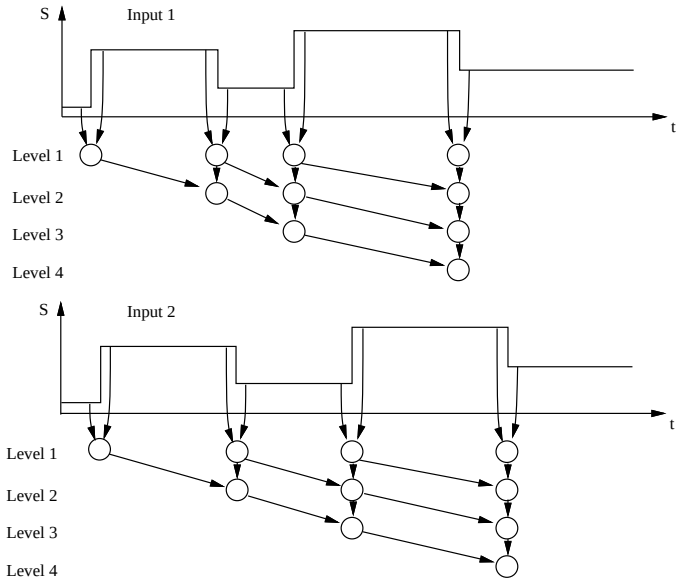


Figure 15: An illustration of time warping and level building. The horizontal axis is the time. The vertical axis represents the state value at level 0. The vertical axis indicates the Y image of P-clusters. Each circle indicates arrival at a new state, at that level, with a high probability. At each time index, a state can transit to itself, i.e., staying at the same state.

of lower-level HMMs is given a new HMM. This results in a large number typically. In the presented work, the state generation and survival all depend on our forgetting model discussed in Section 5.3. To fully utilize the timewarping capability of short state sequence with HMM, we define the length of two states (P-clusters at level 0) as a chunk represented by the state at the next higher level. However, the number of generated and survived states at the higher level depends on the actual cooccurrence and clustering at the higher level, not the number of all possible state combinations (i.e., transitions) at the lower level, which could be very large. This advantage is difficult to achieve with the manual level-building approach.

The level building element (LBE) is shown in Fig. 16. Each level has a single LBE, it dynamically generates new states and forgets states as in temporal adjacency clustering. At each system cycle,

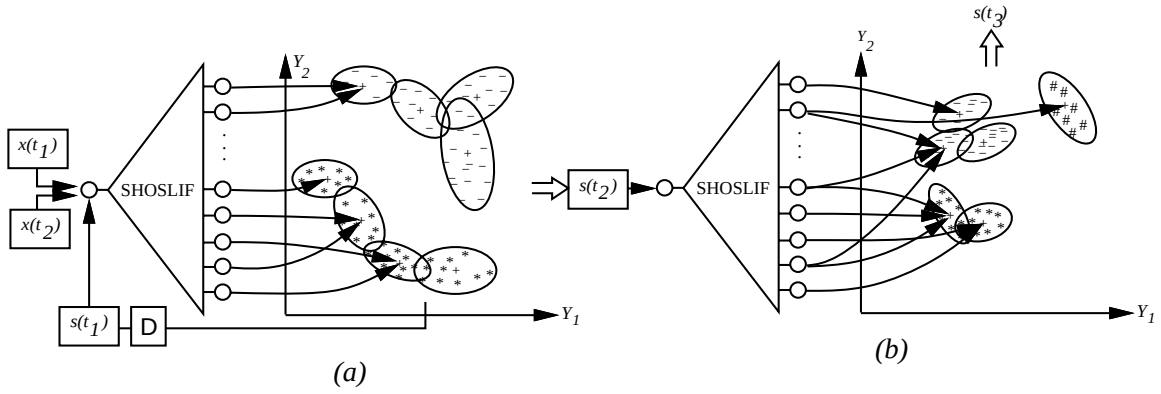


Figure 16: Level building element (LBE). At current state $s(t_1)$, it accepts input $x(t_1), x(t_2)$ from the lower level and gives the next state $s(t_2)$. $s(t_2)$ is used to predict the next state $s(t_3)$. Each LBE contains two components: (a) the transition associator for transition memory and recall, as denoted by f in Equation (4). (b) the predictor for action generation denoted by h in Equation (5) and prediction of the next state. It is to enable the system to think and plan. "D" denotes a delay register. The predictor may predict several states as the candidates for the next state. As shown in the figure, each leaf node in a predictor may have pointers to several clusters.

the LBE accepts two consecutive (different) states from the lower level. It has two components, the transition associator and the predictor.

The purpose of the transition associator in Fig. 16 is to memorize and recall the corresponding state transition. It realizes f in Equation (4). It represents a longer temporal chuck grouped from the two lower level chunks. It receives two states, the previous state and the current state, from the lower level, and its output space is the input to the next higher level. The transition associator performs spatiotemporal clustering in its Y space, as discussed in Section 5.4. This allows for better generalization. Note that the temporal window w and spatial radius σ, used in pulling, do not significantly change from one level to the next. Otherwise the system cannot have an appropriate discrimination power at high levels.

The predictor has two tasks. First, it is to generate actions as denoted by h in Equation (5). Second, it is to predict a set of most probable candidates for the next state $s(t_2)$ based on the current state $s(t_1)$. This to enable the system to predict and plan, when the sensory input is shut-off by the action "think." The predictor has only one state input. It is well known in psychology that the priming capability of the brain is very important for humans to be able to predict what is going to happen next [5]. If the system at state $s(t_1)$ predicts state $s(t_2)$, $s(t_1)$ is called the prime and $s(t_2)$ is called the target. Each visit to the transition associator with input $(s(t_1), s(t_2))$ is associated with two visits to the corresponding predictor. One visit is to use $s(t_2)$ to make predictions. The other is to use $s(t_1)$ as the input to the predictor and $s(t_2)$ as the desired output to train the predictor. In general, the predictor may give more than one predicted next states, each with a probability measure.

When the system is run for the first time, the system automatically creates a LBE. With a continuous input stream, the number of nodes of the SHOSLIF tree in the lower level LBE will increase or decrease (forgetting) dynamically. When the number of nodes in the SHOSLIF tree is sufficiently large, a new LBE is automatically created at the next higher-level, which starts to collect inputs to form clusters. The resulting architecture is shown in Fig. 17. We call it spatiotemporal associator (STA). The effect of learning mainly causes the number of nodes and clusters to increase within each LBE. At level m, each state corresponds roughly to an input sequence of length 2^m in terms of P-clusters. A sequence not of length of power-of-two is expected to appear at a lower level. Due to the time warping effect, the actual time length of the input sequence at each level can vary tremendously.

Fig. 18 is a flow diagram of the SAIL system. The system is designed to be turned on for an extended period everyday. As soon as the system is on, it updates itself according to the flow

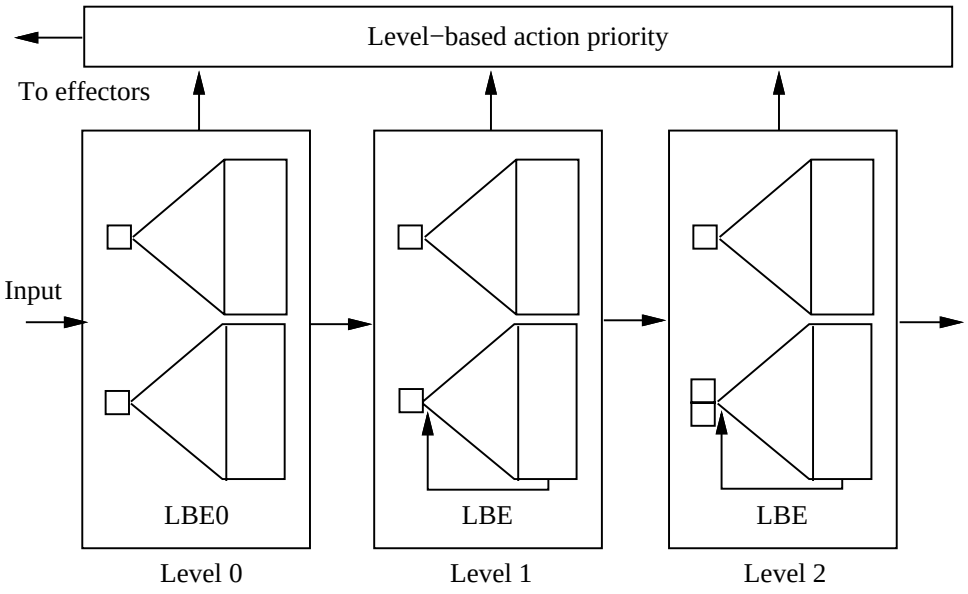


Figure 17: Automatically building levels. The number of levels is grown according to the maturity of existing levels.

chart. Each loop corresponds to an input refreshing period called machine cycle. The human teacher interactively plays with the system using two modes, supervised mode and reinforcement mode. Whenever there is an action imposed, the teaching is at supervised mode. As indicated by the flow-chart, the system always complies with the action imposed. Otherwise, the teaching is at reinforcement mode. The system is allowed to try on its own, while receiving reward, occasionally,

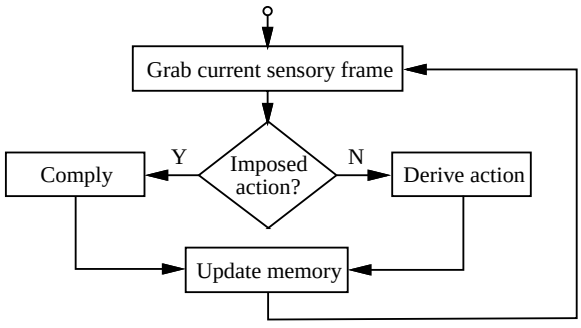


Figure 18: The flow chart of SAIL.

from the teacher.

5.6 Chunking

One very important phenomenon we will explore can be termed *selective chunking.* For example, let us consider an input sequence "`this word`" assuming that a letter corresponds to a P-cluster in the input state. All the possible sequences of length 4 will be "`this`", "`his w`", "`is w`", "`s wo`", etc. However, our automatic level building method will not generate all such states at level 2, if the string ""`this word`" is learned in a structured way. For example, a teacher may present words "`this`" and "`word`" individually to learn first. Then, when learning the phrase "`this word`", the successful response of "`this`" and "`word`" at level 2 means that the clusters for "`this`" and "`word`" have already been formed. The transition of two states is recorded at level 3 which forms a single cluster for "`this word`". Such a selective chunking will discourage other chunks to be formed at level 2 because of the effect of memory fade factor for a score near 0.5 as discussed in Section 5.3.

A well recognized property of human short term memory is called "short term memory bottleneck" [15] [66] [6]. It is easier to remember if the sequence is grouped into richer, more complex items called chunks (or units)[16] The level building scheme intends to allow the system to learn and memorize short chunks at a lower level and then remember longer sequences at a higher level from the learned chunks at the lower level.

[15] For example, if you hear a string of about ten single digits, read at a constant and fairly rapid rate, and then asked to reproduce the string, you generally cannot recall more than about seven or so of the digits.

[16] Try to remember the following 40 letters

`BYGROUPINGITEMSINTOUNITSWEREMEMBERBETTER`

No one can remember these 40 letters correctly if they are treated as 40 separate, unrelated letters in a string.

5.7 Recognition through spatial temporal association

In the above discussion, we mainly concentrate on temporal sequence. In fact, recognition of a spatial object is performed in time, since the learning process of recognition is also dynamic in the autonomous learning mode.

The recognition result is given by the system as an action through the N-effector discussed in Section 2.3. For example, we can teach the system to give a coded label which indicates the name of the object. A question can be coded as a number in the N-sensor. Thus, we need to teach the system to give a correct code whenever it recognized an object.

The presentation of an object is continuous. For example, the object is rotated continuously. The temporal sequence of the input is learned as a temporal event. We can impose a correct action (i.e., giving the correct code) as soon as a "question" is asked through the N-sensor, during the presentation of the object. Temporal frames will cause the system to cluster their trajectory into states. Since the views from different angles are presented without a particular order, a view may be followed by many other similar views in the presentation. This implies that a view can prime other similar views, represented by P-clusters. At a higher level, such a priming action covers larger P-cluster sequences. During the learning, the desired action may be associated with several views (P-clusters), at different levels. For the test, the recognizer is shown with an object to be recognized and a question is asked through the N-sensor, the STA runs to find primed P-clusters and find the best associated reward among the primed P-clusters. If the result is positive, the action is executed through the N-effector. This action is then associated with the current view, allowing it to be used for target of later priming activities. Fig. 19 graphically explains how temporal association at different levels is applied to the object, which allows the object to be recognized from different

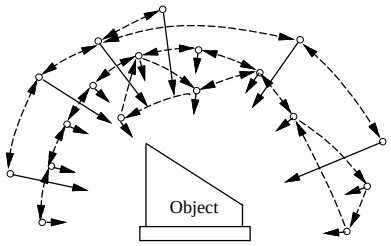


Figure 19: A graphic explanation of the multilevel priming which allows the generalization of association of different views to the same class name. A solid short arrow indicates a view represented by a P-cluster. A solid long arrow indicates a subsequence of views represented at a higher level. A dashed arrow indicates the link of priming, from the prime to the target. A double-direction link indicates that priming in both directions exist. The forgetting process will merge some of the clusters, thus resulting in some generalization and forgetting some detail.

angles. The state distribution pattern shown in Fig. 19 reflects the learning experience of the recognizer, although the behavior of recognition does not necessarily clearly show the distribution details of this pattern.

What if an object is only viewed from a limited number of viewing angles? For example, if the front view of the object is presented but not the back view. Then, the recognizer will not be able to recognize the object from the back view if the front view is very different from the back view and the SHOSLIF top-k matches will not give a matched leaf node with sufficient confidence. If the back view is presented individually, the system still cannot link the front view with the back view. Only when the front view and the back view are presented consecutively or the object is turned around slowly, can the system link the front view with the back view through the temporal association.

5.8 The effect of forgetting

As we discussed in Section 5.3, the forgetting process is performed according to the memory trace shown in Fig. 12. When a leaf node in the SHOSLIF is forgotten, the leaf node is merged into its parent and the corresponding state disappears. Later visits to the forgotten node will be directed to its nearest neighbor cluster. This means that a cluster will correspond to a larger region. Thus, the effect is that details are forgotten and generalization results. For example, in Fig. 19, the dashed links indicate the path of state transition during learning. After a certain amount of time, some of the states will be merged, making some sections along the path of learning to be forgotten and thus resulting in some generalization. This is because in an HMM with fewer states along a temporal trajectory, each state represents larger regions in the input space.

5.9 Incremental, online learning algorithm using Markov models

HMM learning algorithms in the speech community are for batch, offline, supervised learning mode. Furthermore, a separate HMM is needed for every sequence (e.g., word) and each word typically has only a small number of states (e.g., 5 to 6). In the operations research community and the AI community, the Q-learning and R-learning algorithms for reinforcement learning of POMM require a world model. There are simple algorithms for online learning, provided that a world model is available.

The presented method is intrinsically incremental and fully takes advantage of SHOSLIF incremental, online, learning-performance unification capability. The algorithm is similar to the Q-learning algorithm for POMM, but the SAIL has multiple levels. Each level works on different levels of representation. A low level represents events of short time duration while a higher level represents those of long duration. The system keeps top k most probable states (MPS) $s(t_1)$ at

each time t_1. An observation $x(t_2)$ is made from input which indicates the state transition from $x(t_1)$ to $x(t_2)$ at the lower level. It's worth noting that the input at the level 1 contains not only the sensory input, but also the previous action just executed, as shown in Fig. 10. For each of the current MPS $s(t_1)$ at each level, the vector $(s(t_1), x(t_2))$ is used by the transition associator to determine the next top k MPS $s(t_2)$. The k MPSs at time t_1, each predicting k MPSs for $s(t_2)$, results in k^2 MPSs for $s(t_2)$. The next states predicted at time t_1 for $s(t_2)$ are used to increase the confidence of those predicted. Top k of those $s(t_2)$ are kept as the MPSs for time t_2 at this level. The action that records the best reward value is executed. The predictor will predict the set of next state $s(t_3)$ and top k of them will be kept.

In summary, we use (a) internal high dimensional representation for states, which addresses the cross-state generalization drawback of Q-learning; (b) predictor for priming and warning generation, and (c) the automatic level building to address the delayed reward problem and thus avoid the drawback of the simple discounted (or average) reward model. The forgetting mechanism is central for controlling the size of the STA. The memory factors will be used to investigate their effect in the learning speed and the growing speed of the network.

5.10 Reinforcing desired behavior

The above discussion about perception is also applicable to association of sensory input to action. Once a set of sensory inputs have activated the associated action, the action should be checked for the recalled reward experience before execution.

At an early learning stage, the expected reward is recalled directly as the reward value associated with similar cases. At each state, among the learned actions associated with the current state, the one with the highest reward record is executed.

At high levels, most actions are not associated with an extreme value of the score (0 or 1). For example, many things that a human adult does everyday do not directly relate to physical pain or pleasure. The selection of alternative actions are based on the best-matched past behavior pattern that has been learned and reinforced. The physical pain or pleasure are responsible to establish those behavior patterns directly at low level but only indirectly at high levels. A hypothetical example is helpful to explain this point. Suppose the system was taught verbally by its human teacher (a) not to say a rude words to humans. It also had learned (b) that if it does not listen to the human teacher, it will feel "pain" (a 0 reward). It has also a wired-in mechanism (c) that a recall of "pain" will suppress the action. Suppose that in some scenario, it recalled some rude words. If it recalls (a) and (b), it will recall "pain" and the mechanism (c) will make it suppress using the rude words. Therefore, after basic behavior has been established through a process of using the physical reward representing "pain" and "pleasure", the system is expected to be taught by being told, without need of resorting to the physical reward very often. Of course, such a high-level behavior cannot be established without a significant period of learning. and the presented project will not reach the sophistication level indicated in this example.

6 How the Living Machines Work

This section discusses how the living machines work through interactions with its environments.

6.1 Early learning

Like the case with a human infant, robot-sitting is needed during the early stage of the cognitive development. Initially, the human teacher uses the action imposition to feed actual control signals to the system via the GUI, joystick or the robot-arm teaching pendant. This is to simulate the way

a human care-taker teaches a human baby by, e.g., holding their hands. Due to cost restrictions, SAIL does not use a compliant robot arm by which human can enforce the configuration of the arm by applying force directly on to the arm. The trainer may mostly use good actions with a good reward value. After a sufficient period of hand-in-hand training, the living machine can be set free to try by itself, first in the same environment and then gradually move to slightly different environments. During this stage, some mistakes can be made, depending on how extensive hand-in-hand training was and how different the new environment is from the training one. The human teacher enters appropriate reward value in real time, to discourage actions that may potentially lead to a failure and encourage actions that can lead to a good result. At this stage, the human teacher may want to speak and use gestures in addition to feeding reward value via a joystick so that the system can associate the visual and auditory signal from the human trainer with the corresponding reward values. Later on, spoken words and gesture will have the similar effects as the reward values.

6.2 Concept learning and behavior learning

This is probably the most interesting, most controversial, and probably most exciting part of the endeavor. Our task is to investigate how a machine can develop high-level knowledge and motives from low-level "physical feedback". The ideas presented here were inspired by studies in child developmental psychology.

SAIL associates each sensory input with the visual and auditory signals from the human to learn the concepts taught by the human. It will learn good behavior from very early stages, such as "do not run toward a wall", "do not run too fast when there are things nearby", "handle it if you see something like that", etc.

The understanding of concepts will be built up through interactions with human, such as the concepts of left, right, up, down, fast, slow, good, bad, right, wrong. Many experiences and the occasions are associated with the concept that is used. The representation of these concepts are implicit in the system. It represents a pattern of response in the network.

Later, many instances will allow the living machine to associate good reward with his actions that follow what the teacher wants. It also will gradually accumulate the information about what the human teacher likes and expects from it. It will then link the concepts of "good and bad", "right and wrong" with the many instances it experienced.

Generalization of the concepts occurs naturally. For example, the living machine first associates human teacher's words with the reward value. Some words appear together with a bad reward that they become aversive stimuli; and some other words appear with a good reward and they become become appetitive stimuli. The living machine gradually establishes a behavior habit to choose actions that will produce appetitive stimuli (to make the human teacher happy) and avoid actions that will produce aversive stimuli. This is called second-order conditioning in psychology (see, e.g., [26]). At that time, the importance of the physical reward value gradually diminish. A default physical reward value is often enough during this stage. A more mature system will receive most human feedback from normal sensory input, via speech, gestures, etc.

Further on, the system will link the sense of good and bad to other more complicated activities, such as schooling, the evaluation system in its school, and what to do in order to receive a good evaluation from its school, etc. In summary, as long as the teacher uses the reward value correctly, he or she will be able to train the living machine to do what he or she wants, from simple to complex, even during later stages when reward is used very rarely.

6.3 Cognitive maturity

In an autonomous learning process, the environment provides an endless sequence of stimuli coupled with the actions from the living machines. Certainly, the living machine should not just remember the sequence as, e.g., a single 8-hour sensing-action sequence everyday, because a lot of events do not have close relationships. The machine should only remember things that are important at its current cognitive developmental stage.

The cognitive maturity determines what a machine can learn. For example, a child in the sensorimotor stage will not be able to learn formal reasoning, even if a teacher tries that. Before a sufficient amount of low-level knowledge and skills has been learned, it is not possible to learn higher-level knowledge and skills. In the living machine, the maturity scheduling is realized automatically by the process of state self-organization, the forgetting process and the level building process.

When a set of stimuli representing a high-level concept is sensed by an immature machine, the configuration of the current SHOSLIF tree is not sufficient to learn the new concept associated with the stimuli. This is reflected by the fact that the features generated by the SHOSLIF tree constructed so far is not able to successfully recall a set of learned concepts. Thus, the corresponding stimuli looks meaningless to the living machine and are just simply temporarily memorized. Without extracting recallable features, stimuli containing the same high-level concepts in different occasions look very different. Later on, that temporary memory is quickly forgotten (deleted) by the forgetting process because there is no recall to this memorized stimuli within a certain time period.

If the living machine has learned a sufficient number of low-level concepts and skills, indicated by the corresponding nodes in the SHOSLIF tree, the same stimuli mentioned above will result in a

very significant amount of feature recall from the mature SHOSLIF tree. Thus, the corresponding stimuli are memorized, at a higher level of the STA hierarchy. Within a certain period of time, if the same concept appears in the stimuli in another occasion, the mature SHOSLIF tree can recall a sufficient number of features and retrieve the newly memorized concept. Such a successful retrieval indicates a memory reinforcement. The forgetting process will record this reinforcement at the corresponding node and apply a much slower memory decay curve to this concept. This concept is then learned by the living machine.

In this way, the living machine is able to learn autonomously, and go through various cognitive developmental stages which are probably similar to those characterized by Jean Piaget (see Table 1). During each day, it learns what it can learn and forgets what it has to forget. As is the case with humans, entering a new cognitive stage by a living machine is natural and gradual, depending on the learning experience with each machine individual. There is no need for the living machine designer to enforce each development stage into the program.

6.4 Thinking

A living machine must be able to think autonomously. How does a machine think? This is a question that has fascinated scientists and the public alike [121]. Probably not many computer researchers like to regard the computational process implemented by a computer as thinking. Otherwise, any program is doing thinking. The thinking process seems to autonomous. However, autonomy is not sufficient for qualifying thinking. Otherwise, an autonomous road-following vehicle is doing thinking.

We must not program logic rules into the living machines A fundamental characteristic of thinking is that the contents and the rules of reasoning cannot be pre-arranged — i.e., programmed in. Let's consider an example. A common sense knowledge "all human are mortal" can be represented by compound proposition (tautology) $\forall x[p(x) \rightarrow q(x)]$ where $p(x) =$ "x is a human" and $q(x) =$ "x is mortal". Then, with input $p(\text{Tom}) = \text{True}$, a program can prove $q(\text{Tom}) = \text{True}$ using logic deduction rules. When doing the above reasoning, the program is not thinking because the logic deduction rules are pre-programmed by the human. In order to make a machine think, its representation must be knowledge-free. In our example, the knowledge is formal logic. The representation for $\forall x[p(x) \rightarrow q(x)]$ is not knowledge-free.

It is worth pointing out that symbolic reasoning is not as difficult as the other parts of cognition. In the above case, e.g., it is much more difficult to figure out that Tom is a human from visual sensing than to perform the symbolic reasoning.

What is thinking? There is no widely accepted definition of thinking [74]. We probably should not attempt to define here either. The following characterization of a thinking process may be considered: (1) A thinking process is autonomous. (2) The representation of the thinking program is free of knowledge (i.e., content independent). (3) The thinking process is conducted according to knowledge that has been autonomously learned. The first two points are meant to distinguish a pre-programmed computation from an autonomous thinking process. The last point is to make sure that thinking is not a useless process or directly directed by humans.

Thinking at mental cycle level With the SAIL living machine, a thinking process might result if the attention selection for external sensors keeps off while state vectors are fed into each level as we see before. A mental cycle of the thinking process is to go through STA once for each input

frame as shown in Fig 10. A long thinking process corresponds to many consecutive mental cycles going through STA, each with a new state vector from the former mental cycle at each level. Thus, a long thinking process may allow prediction of many chained events. Naturally, all the thinking processes are originated from the stimuli in the environment. During the thinking process, the stimuli input from the sensors may be temporarily turned off to allow prediction and planning to be performed internally. When to think and how to think are determined by the system's learned behavior.

6.5 Reasoning and planning

One might think that in order to conduct reasoning and planning, there must be a controller, which controls what to think about and organizes various stages of reasoning and planning. However, no such controller can work.

Must not have a thinking controller There is a fundamental dilemma with this "controller thinking." This controller, as a supervisor, must be smarter than what it controls. In other words, it must know the global situation, understand it, and figure out what to do and how to do. Furthermore, this controller must be general-purpose because the living machine is a general purpose creature. We know that no controller can perform a reasonable control task without understanding what is going on. However, understanding is exactly our original problem. Thus, we have a chicken-and-egg problem — one cannot be solved without first having the other.

Programming behaviors into system cannot work either Another alternative is that we do not use any controller. This is the case with some behavior-based methods (e.g., see Brooks [11]). However, the behavior-based methods cannot go beyond the very limited number behaviors

that have explicitly modeled and programmed in. None of the existing behavior-based approaches really try to understand the world. Human beings, on the other hand, can gradually learn and understand more and more complex concepts in the world and their high-level behaviors are based on understanding instead of pure wired-in reflexes.

Reasoning and planning are special types of thinking process in the living machines. The thinking process described above is used for reasoning and planning as soon as the machine is thinking about the sequence of events where one event is likely to follow the next event. Humans may teach the living machine how to reason by demonstrating visual examples or by stating a story or theory after the living machine has established a certain language capability.

7 Further Thoughts

7.1 Generality

Can the system potentially do anything that a human can? This is an open question. The problem here is not just a static function approximation which can be performed by, e.g., a three-layer neural network with back-propagation. The critical issues include the dynamic generation of concepts from sensor-effector-based autonomous learning, concept generalization through experience, the capability to automatically generate feature space, and self-organizing the dynamically changing network. Autonomous learning through real-time sensing and action without hand-crafted knowledge-level rules or behaviors is a totally new subject of study. The upcoming study will answer some very important and fundamental questions.

7.2 Space complexity

The space complexity is directly related to the amount of information learned. The human brain has about 10^{11} neurons, each being connected by roughly 10^3 synapses on average [79] [53] [3]. If each synaptic link is considered a number, the human brain can store about 10^{14} numbers. This amount is now within reach by hard disks as far as the cost is concerned, thanks to the fast advance in computer storage technology. It is worth noting that the nature of the SHOSLIF tree data allows a moderate compression that is typical in video compression.

It is known that a large part of the neurons in the human brain is not activated. Furthermore, the living machine does not need its brain to control heart beating, breathing, and digestion, which are served mainly by the medulla and the pons in the human brain. It does not need much service from the somatocensory system. The taste system and the olfactory system are not needed either except for certain special applications. It probably does not need its brain to serve for sexual drive either, which is taken care of partially by the amygdala in the human brain. Further, computers have effective high-level computational mechanisms, such as the fast and effective algorithms for computing eigenvector and eigenvalues of a huge matrix, which is a major part of computations in SHOSLIF. The known methods for computing eigenvectors by artificial neural networks are slow, iterative, and not as accurate [60] [87] [88]. Thus, the living machine might be able to reach a good performance with a disk space that is significantly smaller than the absolute storage size of the human brain. Of course, this also depends on the scope of the domain to be learned and the required resolution of the sensors. Since the cost of magnetic hard disks and rewritable optical disks is going down fast and consistently, it is now possible to equip a system with a disk system of 1,000 GB (about 20% of the absolute storage size of the human brain with a moderate compression) at a

cost of about $20,000. It is not clear what kind of performance SAIL can reach with a storage size ranging from 10 GB to 1,000 GB. This is one of the major questions to be answered in our project.

If the size of the brain were not an important issue, perhaps monkey could serve as a living machine — a seemingly cheaper one. However, the size is probably just one of the problems with monkey's brain. Although a monkey can perform sensing and action tasks that no machine can do so far, the genetic coding of the monkey brain might not enable it to work up to a high-level comparable to that of humans. We cannot control the self-organization scheme of the monkey's brain, at least not now. With a living machine, we do not have such a limitation.

7.3 Time complexity

The time complexity should not be addressed in a conventional way. Here the time needed to train the system to reach a certain level of performance is on the order of months or years. It is expected that a machine can learn faster than human beings because it does not feel tired and it computes faster. The speed of learning with the living machines seems more controllable than biological systems, such as humans.

The critical type of complexity is the time complexity for each mental cycle when the size of the network becomes very large. Because of our on-line learning and the subspace method, the time complexity for each mental cycle is $O(\log(n))$, which is an inverse function of exponential explosion. In other words, when the processing speed increases, the number of cases it can handle in a fixed 100 millisecond time interval grows exponentially. To see how low the logarithmic complexity is, suppose a system whose time complexity is $O(log_b(n))$, regardless of how big the constant coefficient is in the time complexity. If this system can complete a mental cycle in 100 milliseconds with 1000 stored cases (which is about the speed SHOSLIF has reached), the same

program running on another computer whose speed is 4.7 times faster can finish a mental cycle in the same time interval for a network that has stored 10^{14} cases (the absolute size of the human brain)! This displays a very high potential for the logarithmic time complexity.

7.4 Knowledge-base

The subject of knowledge representation and knowledge-base has been studied for many years without serious consideration about sensing and action. A huge amount of human power has been spent to model human knowledge, its representation, and input of data into knowledge-bases. However, the resulting systems are hard to use, hard to maintain, hard to keep up to date, and are brittle for a lack of understanding of what has been stored.

What the field of knowledge representation and knowledge-base has experienced is a natural stage that we humans must go through on our way toward understanding ourselves, machines, our environments, and their relationships. However, it is about time that we tried a fundamentally different approach, an approach that is much closer to the way a human acquires knowledge. Not too surprisingly, the living machine approach seems to require much less manpower and costs less than many conventional knowledge-base projects, since human is relieved from the tremendous task of building rules for human knowledge and spoon feeding human knowledge. For knowledge-base construction, we want to move from manual labor toward automation.

7.5 Is it a formidable endeavor?

To consider whether the proposed living-machine project is formidable, it is helpful to compare the nature of the proposed domain-independent approach with that of task-specific ones.

Task-specific approach: With a task-specific approach, humans manually model knowledge for

each task. Thus, each task is very hard because the knowledge required in each task is too vast in amount and too complicated in nature. Accustomed with task-specific approaches, it is hard for us to believe that any algorithm can handle general-purpose learning because each task is already too hard.

The living machine approach: With the proposed living machine approach, humans are relieved from the tasks of developing knowledge-level representation, manually modeling knowledge, and programming knowledge-level rules into programs. Instead, we develop a systematic, unified method to model system's self-organization scheme at the signal level (instead of the knowledge level). Thus, the development task tends to be less labor intensive because the algorithm is very systematic and domain independent. No knowledge needs to be programmed into the program and one program is meant for various sensing and action modalities.

Developing living machines is certainly not easy. Many unknowns need to be explored and studied. It is naive to think that challenging tasks such as vision, speech and language are easy to meet once we have an AA-learning framework that probably will eventually work. However, the development of AA-learning systems appears more tractable than many task-specific projects which use intensive task-level knowledges in each area, such as vision, speech, hand-written and mix-printed document recognition, autonomous robots, knowledge base development, and language understanding.

8 The Future of the Living Machines

The success of Phase 2 may require coordinated efforts from many research groups in various areas. It is expected that the learning algorithm developed in Phase 2 must undergo many versions of improvement before the goals of Phase 2 can be realized. Such improvements would continue even

in Phase 3. Although it is now too early to predict the time table for Phase 2, the implication of the developmental approach and the living machine concepts have opened a new way of thinking about intelligent machines and new array of future possibilities. Here we discuss some of those possibilities.

8.1 Smart toys

Before the living machines have reached a mature level, the first possible mass-market application is probably a new generation of smart living toys. They are trainable, smart toys that live with human beings. Such a toy has its sensors (visual, audio, ultrasound, tactile, text-input, etc) and its effectors (speaker, hand, steerable wheels, text-output etc), depending on the chosen sophistication level of the toy. The most basic, low cost version of these smart living toys does not need a mechanical robot body. It is a piece of software in a multimedia computer. A joystick connected to the computer feeds human trainer reward or punishment into the computer, e.g., 1.0 representing a reward (like food) and -1.0 representing a punishment (like hunger). The human trainer can also guide actions using a joystick or a graphical interface, very much like the case where a human teacher trains a dog how to reach a ball by holding dog's legs to touch the ball. Each living toy is loaded with a learning program which controls the toy to learn and to act to maximize the expected reward. Since the trainer trains the toy by playing with it, instead of programming knowledge-rules into it, the learning of the living toy is automatic. Thus, any person is able to train such a living toy with fun. Various toy competitions at various sophistication levels can be held once such "living toys" have reached the mass-market, which in turn further promotes wide ownership and upgrade of such toys.

8.2 A revolutionary way of developing intelligent software

After completion of Phase 3 in the above plan, continued education for living machines is more or less like teaching a human child. A large number of living machines will be made. Each newly made machine is loaded with the Phase 3 brain (software and network as a database) as the starting point. Computer experts work with educational experts to teach professional skills to living machines, very much like the way human students are taught. Each professional field or subject has a living-machine school, in which several living-machine robots are taught to master the professional knowledge. What each school does is to train a generic living machine to become a professional expert. A living machine can be trained to become a space-craft pilot, a fighter plane pilot, a deep-sea diver, a waste-site cleaner, a security-zone monitor, an entertainer, a nursing-home care-taker, a tutor, or a personal assistant. Each living machine is a software factory. This represents a revolutionary way of making intelligent software: Hand programming is no longer a primary mode for developing intelligent software, but rather, school teaching is. Teaching and learning are carried out through visual, auditory, and tactile communications. Two types of products will be sold, expert software brains and expert living machines.

8.3 Expert software brains as product

The expert brain (software and network) of each specially trained living machine is down-loaded and many copies are sold as software brain. The software-only brain is useful and cheap, but it does not have a full learning capability because of its loss of most sensors and effectors. For example, a software-only brain running on the Internet can only learn from those available from the Internet, based on its knowledge it learned when it had a full array of sensors and effectors. Depending on the richness of the information available from the Internet and how his owner instructs its learning

from the Internet, this software brain's knowledge may gradually become obsolete. This is very similar to the case where a human's knowledge becomes obsolete, once he or she does not keep learning. However, software brain buyers can always get upgrades regularly from software brain makers, very much like the way commercial software gets upgraded now.

8.4 Expert living machines as product

Each living machine school also sells expert "brains" with a physical body as a full living machine with sensors and effectors. Such complete living machines can continue to learn at their application sites to adapt to the new environment and to learn new skills. This type of full living machines are used to extend three types of human capabilities (1) physical capability, such as operating a machine or driving a car, (2) thinking capability, such as finding the most related literature from a library or replying piled-up electronic mails, (3) learning capability, such as learning to use computers for the handicapped or learning to predict economic ups and downs better. Humans can do things that they enjoy doing, leaving living machines to do other things. Further more, the living machines can work round-the-clock without fatigue and loss of concentration, which is something that no human being can match.

8.5 The longevity of living machine

The living machines eventually *outlive* human individuals because once their hardware is broken or worn, the brain can be down-loaded and then up-loaded to a new hardware. They can also learn faster and learn more hours everyday than each human individual. This longevity may lead to tremendous creativities. For example, Albert Einstein passed away and human lost his creativity. The creativity and the knowledge that each living machine learned can be preserved for future

human generations for more innovations and better service.

8.6 Social issues with living machines

Just like the case with human children or pets, each human teacher is responsible for his or her living machine. If a living machine happens to have learned something that we do not want it to learn, we can always replace its brain with the backed-up copy of, say, yesterday, to make it back to the yesterday's status. We do not have such a convenience with a human child or a pet.

8.7 A new industry

Now, the automobile industry is huge because every household needs at least one automobile. The computer industry is huge now because the computer is general purpose in extending human's computational needs (and all the functionalities that accompany the computation). Unlike all the special-purpose robots that have been built so far, the living machine is general purpose. Therefore, the market for the living machine is huge. With the decreasing cost that accompanies advances in robotic technology and computer technology and the ever increasing volume of mass-market sale, the living machines will not only enter every business and industrial sector but also millions of American households to serve as intelligent personal assistants.

9 Conclusions

A new approach — the developmental approach — has been introduced here with an algorithmic description [17]. The developmental approach is motivated by human cognitive development from

[17] Our earlier account for the concept and motivation of the living machines appeared in a report dated Dec. 1996 [113].

infancy to adulthood. The key for success of this approach is to realize automation of general-purpose learning (AA-learning) and to make it effective. The developmental approach does not just mean growing a network from small to big, from simple to complex [29] [114], although these concepts are related to the cognitive development. More importantly, it means AA-learning: free of manual design for task space, free of manual design for behavior-level modules or decomposition, closed brain, autonomous learning and learning while performing. Since learning by the living machines is automated and the training process is similar to the way human teaches children, training for living machines is natural and widely practicable by nonprogrammers. The future challenge is to further improve the basic framework described here and realize the design by real systems (see [117] [118] [119] for some preliminary results of AA-learning using the algorithmic design outlined here). The research on the living machines may help us to further understand sensorimotor learning, visual and speech learning, language acquisition, thinking, as well as how the mind works. Although it is now too early to promise a success, the future of this direction seems very exciting.

Acknowledgements

The SHOSLIF work was supported in part by NSF Grant IRI 9410741 and ONR grant No. N00014-95-1-0637. The author would like to thank Daniel L. Swets, Yuntao Cui, Shaoyun Chen, Sally Howden & Wey-Shiuan Hwang, and Jason Brotherton for discussions and development of SHOSLIF-O, SHOSLIF-M, SHOSLIF-N, SHOSLIF-R and SHOSLIF-S, respectively. Thanks also go to an anonymous reviewer who made constructive comments which are very useful for the improvement of the presentation here.

Bibliography

[1] I. R.. Alexander, G. H. Alexander, and K. F. Lee. Survey of current speech technology. *Communications of the ACM*, 37(3):52–57, March 1994.

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

[3] J. R. Anderson. Cognitive and psychological computation with neural models. *IEEE Trans. Systems, Man and Cybernetics*, 13(5):799–815, 1983.

[4] J. R. Anderson. *Cognitive Psychology and Its Implications.* Freeman, New Worky, third edition, 1990.

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

[6] A. D. Baddeley, N. Thompson, and M. Buchanan. Word length and the structure of short-term memory. *Journal of Verbal Learning and Verbal Behavior*, 14:575–589, 1975.

[7] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman. Eigenfaces vs fisherfaces: Recognition using class specific linear projection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(7):711–720, July 1997.

[8] C. Blakemore and G. F. Cooper. Development of the brain depends on the visual environment. *Nature*, 228:477–478, Oct. 1970.

[9] A. Bobick and A. Wilson. A state-based technique for the summarization and recognition of gesture. In *Proc. 5th Int'l Conf. Computer Vision*, pages 382–388, Boston, 1995.

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

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

[12] R. Brooks and L. A. Stein. Building brains for bodies. Technical Report 1439, MIT AI Lab Memo, Chambridge, MA, August 1993.

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

[14] J. Brotherton and J. Weng. HEARME: Speaker independent word recognition using SHOSLIF. Technical Report CPS 96-33, Department of Computer Science, Michigan State University, East Lansing, MI, Oct. 1996.

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

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

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

[18] K. W. Church and L. F. Rau. Commercial applications of natural language processing. *Communications of the ACM*, 38(11):71–79, 1995.

[19] A. J. Colmenarez and T. S. Huang. Face detection with information-based maximum discrimination. In *Proc. IEEE Conf. Comp. Vision Pattern Recognition*, pages 782–787, 1997.

[20] T. M. Cover. Estimation by the nearest neighbor rule. *IEEE Trans. Information Theory*, 14:50–55, Jan. 1968.

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

[22] Y. Cui, D. Swets, and J. Weng. Learning-based hand sign recognition using SHOSLIF-M. In *Proc. IEEE Int'l Conf. Comptuer Vision*, pages 631–636, Cambridge, MA, June 1995.

[23] Y. Cui and J. Weng. Hand segmentation using learning-based prediction and verification for hand-sign recognition. In *Proc. IEEE Conf. Comp. Vision Pattern Recognition*, pages 88–93, 1996.

[24] T. Darrell and Alex Pentland. Space-time gesture. In *Proc. IEEE Conf. Comp. Vision Pattern Recognition*, pages 335–340, New York, NY, June 1993.

[25] E. D. Dickmanns and A. Zapp. A curvature-based scheme for improving road vehicle guidance by computer vision. In *Proc. SPIE Mobile Robot Conf.*, pages 161–168, Cambridge, MA, Oct. 1986.

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

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

[28] R. Duda and P. Hart. *Pattern Classification and Scene Analysis.* Wiley, New York, NY, 1973.

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

[30] 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.

[31] S. Franklin. *Artificial Minds.* MIT Press, Cambridge, MA, 1997.

[32] J. H. Friedman. A recursive partition decision rule for nonparametric classification. *IEEE Trans. on Computers*, 26:404–408, April 1977.

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

[34] W. E. L. Grimson and J. L. Mundy. Computer vision applications. *Communications of the ACM*, 37(3):45–51, 1994.

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

[36] 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.

[37] H. Hendriks-Jansen. *Catching ourselves in the act: Situated activity, interactive emergence, evolution and human thought.* MIT Press, Cambridge, MA, 1996.

[38] T.S. Huang and V. I. Pavlovic. Hand gesture modeling, analysis, and synthesis. In *Proc. Int'l Workshop on Automatic Face- and Gesture-Recognition*, pages 73–79, 1995.

[39] D. H. Hubel. *Eye, Brain, and Vision.* Scientific American Library, Distributed by Freeman, New York, 1988.

[40] W. Hwang, S. J. Howden, and J. Weng. Performing temporal action with a hand-eye system using the SHOSLIF approach. In *Proc. Int'l Conference on Pattern Recognition*, Vienna, Austria, Aug. 25-30 1996.

[41] W. Hwang and J. Weng. Vision-guilded robot manipulator control as learning and recall using SHOSLIF. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, Albuquerque, NM, April 20-25 1997.

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

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

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

[45] F. Jelinek. Self-organized language modeling for speech recognition. In A. Waibel and K. Lee, editors, *Readings in Speech Recognition*, pages 450–506. Morgan Kaufmann, San Mateo, CA, 1990.

[46] I. T. Jolliffe. *Principal Component Analysis.* Springer-Verlag, New York, 1986.

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

[48] S. B. Kang and K. Ikeuchi. A robot system that observes and replicates grasping tasks. In *Proc. Int'l Conf. Comp. Vision*, pages 1093–1099, Cambridge MA, 1995.

[49] K. Karhunen. Uber lineare methoden in der wahrscheinlichkeitsrechnung. *Ann. Acad. Sci. Fennicae, ser. A1, Math. Phys.*, 37, 1946.

[50] R. Kjeldsen and J. Kender. Visual hand gesture recognition for window system control. In *Proc. Int'l Workshop on Automatic Face- and Gesture-Recognition*, pages 184–188, Zurich, Switzerland, June 1995.

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

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

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

[54] T. Kuniyoshi, M. Inaba, and H. Inoue. Learning by watching: Extracting reusable task knowledge from visual observation of human performance. *IEEE Trans. on Robotics and Automation*, 10(6):799–822, Dec. 1994.

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

[56] A. Lanitis, C. J. Taylor, T. F. Cootes, and T. Ahmed. Automatic interpretation of human faces and hand gestures using flexible models. In *Proc. Int'l Workshop on Automatic Face- and Gesture-Recognition*, pages 98–103, Zurich, Switzerland, June 1995.

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

[58] 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.

[59] Q. Li and J. Weng. SHOSLIF convergence properties. Technical Report CPS 96-26, Department of Computer Science, Michigan State University, East Lansing, MI, May 1996.

[60] J. Mao and A. K. Jain. Artificial neural networks for feature extraction and multivariate data projection. *IEEE Trans. Neural Networks*, 6(2):296–317, March 1995. Outstanding paper award.

[61] M. J. Mataric. A distributed model for mobile robot environment - learning and navigation. Technical Report AI-TR-1228, MIT Artificial Intelligence Laboratory, Cambridge, MA, 1990.

[62] M. J. Mataric. Integration of representation into goal-driven behavior-based robots. *IEEE Trans. on Robotics and Automation*, 8(3):304–312, June 1992.

[63] J. M. McInnes and J. A. Treffry. *Deaf-Blind Infants and Children.* University of Toronto Press, Toronto, Ontario, 1982.

[64] M. Meng and A. C. Kak. Mobile robot navigation using neural networks and nonmetrical environment models. *IEEE Control Systems*, pages 31–42, Aug. 1993.

[65] R. Michalski, I. Mozetic, J. Hong, and N. Lavrac. The multi-purpose incremental learning system AQ15 and its testing application to three medical domains. In *Proc. Fifth Annual National Conf. Artificial Intelligence*, pages 1041–1045, Philadelphia, PA, 1986.

[66] G. A. Miller. The magical number seven, plus or minus two: Some limits on our capability for prcessing information. *Psychological Review*, 63:81–97, 1956.

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

[68] M. Minsky. A selected descriptor-indexed bibliography to the literature on artificial intelligence. In E. A. Feigenbaum and J. Feldman, editors, *Computers and Thought*, pages 453–523. McGraw-Hill, New York, NY, 1963.

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

[70] B. Moghaddam and A. Pentland. Probabilistic visual learning for object detection. In *Proc. Int'l Conf. Comp. Vision*, pages 84–91, Cambridge, MA, 1995.

[71] H. P. Moravec. The Stanford Cart and the CMU Rover. *Proceedings of the IEEE*, 71(7):872–884, 1982.

[72] 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.

[73] S. K. Murthy. Automatic construction of decision trees from data: A multidisciplinary survey. *Data Mining and Knowledge Discovery*, 1998.

[74] A. Newell. *Unified Theories of Cognition.* Harvard University Press, Cambridge, MA, 1990.

[75] N. Nilsson. SRI A.I. center technical note. Technical Report 323, Stanford Research Institute, April 1984.

[76] J. Yamato J. Ohya and K. Ishii. Recognizing human action in time-sequential images using hidden Markov model. In *Proc. IEEE Conf. Comp. Vision Pattern Recognition*, pages 379–385, 1992.

[77] T. Pavlidis. Why progress in machine vision is so slow. *Pattern Recognition Letters*, 13:221–225, April 1992.

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

[79] M. I. Posner and M. E. Raichle. *Images of Mind.* Scientific American Library, New York, 1994.

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

[81] J. Quinlan. Introduction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

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

[83] L. R. Rabiner, L. G. Wilpon, and F. K. Soong. High performance connected digit recognition using hidden Markov models. *IEEE Trans. Acoustics, Speech and Signal Processing*, 37:1214–1225, Aug. 1989.

[84] V. S. Ramachandran. Perceiving shape from shading. In I. Rock, editor, *The Perceptual World*, pages 127–138. Freeman, San Francisco, CA, 1990.

[85] M. Rosenblum and Larry S. Davis. An improved radial basis function network for visual autonomous road following. *IEEE Trans. Neural Networks*, 7(5):1111–1120, 1996.

[86] H. A. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 20(1):23–38, 1998.

[87] J. Rubner and K. Schulten. Development of feature detectors by self-organization. *Biological Cybernetics*, 62:193–199, 1990.

[88] J. Rubner and P. Tavan. A self-organizing network for principal component analysis. *Europhysics Letters*, 10:693–698, 1989.

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

[90] O. Sacks. To see and not see. *The New Yorker*, pages 59–73, May 1993.

[91] 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.

[92] Wei-Min Shen. *Autonomous Learning from the Environment*. Computer Science Press, New York, 1994.

[93] P. Sinha and T. Poggio. I think I know that face ... *Nature*, 384:404, December 1996.

[94] T. Starner and A. Pentland. Visual recognition of American sign language using hidden Markov models. In *Proc. Int'l Workshop on Automatic Face- and Gesture-Recognition*, pages 189–194, Zurich, Switzerland, June 1995.

[95] K. K. Sung and T. Poggio. Example-based learning for view-based human face detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 20(1):39–51, 1998.

[96] 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.

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

[98] D. L. Swets and J. Weng. Hierarchical discriminant analysis for image retreival. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 21(5):386–401, 1999.

[99] P. Thompson. Margaret thatcher: a new illusion. *Perception*, 9:483–484, 1980.

[100] C. Thorpe, M. H. Hebert, T. Kanade, and S. Shafer. Vision and navigation for the Carnegie-Mellon Navlab. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 10(3):362–373, 1988.

[101] C. Thorpe, M. Herbert, T. Kanade, and S. Shafer. Toward autonomous driving: The CMU Navlab. *IEEE Expert*, 6(4):31–42, August 1991.

[102] J. Triesch and C. von der Malsburg. Robust classification of hand posture against complex background. In *Proc. Int'l Conf. on Automatic Face- and Gesture-Recognition*, pages 170–175, Killington, Vermont, Oct. 1996.

[103] A. M. Turing. On computable numbers with an application to the Entscheidungsproblem. *Proc. London Math. Soc.*, 2(42):230–265, 1936.

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

[105] M. A. Turk, D. G. Morgenthaler, K. D. Gremban, and M. Marra. VITS-a vision system for autonomous land vehicle navigation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 10(3):342–361, May 1988.

[106] A. Waibel and K. Lee. *Readings in Speech Recognition.* Morgan Kaufmann, San Mateo, CA, 1990.

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

[108] J. Weng. On comprehensive visual learning. In *Proc. NSF/ARPA Workshop on Performance vs. Methodology in Computer Vision*, pages 152–166, Seattle, WA, June 1994.

[109] J. Weng. SHOSLIF: A framework for object recognition from images. In *Proc. IEEE International Conference on Neural Networks*, pages 4204–4209, Orlando, FL, June 28-July 2 1994.

[110] J. Weng. SHOSLIF: A learning system for vision and control. In *Proc. IEEE Annual Workshop on Architectures for Intelligent Control Systems*, Columbus, Ohio, August 16 1994.

[111] J. Weng. SHOSLIF: A framework for sensor-based learning for high-dimensional complex systems. In *Proc. IEEE Workshop on Architectures for Semiotic Modeling and situation analysis in Large Complex Systems*, pages 303–313, Monterey, CA, Aug. 1995.

[112] J. Weng. Cresceptron and SHOSLIF: Toward comprehensive visual learning. In S. K. Nayar and T. Poggio, editors, *Early Visual Learning*, pages 183–214. Oxford University Press, New York, 1996.

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

[114] 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.

[115] J. Weng and S. Chen. Autonomous navigation through case-based learning. In *Proc. IEEE Int'l Symposium on Computer Vision*, pages 359–364, Coral Gables, FL, Nov. 1995.

[116] 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.

[117] J. Weng and W. Hwang. Toward automation of learning: The state self-organization problem

for a face recognizer. In *Proc. 3rd International Conference on Automatic Face- and Gesture-Recognition*, pages 384–389, Nara, Japan, April 1998.

[118] J. J. Weng and W. Hwang. Sensorimotor action sequence learning with application to face recognition under discourse. In *Proc. Int'l Conf. on Pattern Recognition*, Brisbane, Australia, August 1998.

[119] J. J. Weng and W. Hwang. A sensorimotor system for spatiotemporal learning with application to face recognition under discourse. Technical Report CPS 98-9, Department of Computer Science, Michigan State University, East Lansing, MI, March 1998.

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

[121] R. Wright. Can machines think? *Time*, pages 50–56, March 25 1996.

[122] C. Yoken. *Living with deaf-blindness: nine profiles.* National Academy of Gallaudet College, Washington, 1979.

[123] S. R. Young, A. G. Hauptmann, W. H. Ward, E. T. Smith, and P. Werner. High-level knowledge sources in usable speech recognition systems. *Communications of the ACM*, 32(2):183–194, 1989.

[124] S. Zeki. The visual image in mind and brain. *Scientific American*, 267(3):69–76, 1992.