Adaptive Behavior

http://adb.sagepub.com

A Hierarchical Autonomous Robot Controller for Learning and Memory: Adaptation in a Dynamic Environment

Fady Alnajjar, Indra Bin Mohd Zin and Kazuyuki Murase

Adaptive Behavior 2009; 17; 179
DOI: 10.1177/1059712309105814

The online version of this article can be found at:
http://adb.sagepub.com/cgi/content/abstract/17/3/179

Published by:

http://www.sagepublications.com

On behalf of:

International Society of Adaptive Behavior

Additional services and information for *Adaptive Behavior* can be found at:

Email Alerts: http://adb.sagepub.com/cgi/alerts

Subscriptions: http://adb.sagepub.com/subscriptions

Reprints: http://www.sagepub.com/journalsReprints.nav

Permissions: http://www.sagepub.co.uk/journalsPermissions.nav

Citations http://adb.sagepub.com/cgi/content/refs/17/3/179

Downloaded from http://adb.sagepub.com at Katholieke Universiteit Leuven on March 15, 2010

Adaptive **Behavior**

A Hierarchical Autonomous Robot Controller for Learning and Memory: Adaptation in a Dynamic Environment

Fady Alnajjar[1], Indra Bin Mohd Zin[2], Kazuyuki Murase[1,2,3]
[1] *Department of System Design Engineering, University of Fukui, Japan*
[2] *Department of Human and Artificial Intelligence Systems, Graduate School of Engineering, University of Fukui, Japan*
[3] *Research and Education Program for Life Science, University of Fukui, Japan*

This work concerns practical issues surrounding the application of learning and memory in a real mobile robot with the goal of optimal navigation in dynamic environments. A novel hierarchical adaptive controller that contains two-level units was developed and trained in a physical mobile robot "e-*Puck*." In the low-level unit, the robot holds a number of biologically inspired *Aplysia*-like spiking neural networks that have the property of spike time-dependent plasticity. Each of these networks is trained to become an expert in a particular local environment(s). All the trained networks are stored in a tree-type memory structure that is located in the high-level unit. These stored networks are used as experiences for the robot to enhance its navigation ability in both new and previously trained environments. The robot's memory is designed to hold memories of various lengths and has a simple searching mechanism. Forgetting and dynamic clustering techniques are used to control the memory size. Experimental results show that the proposed model can produce a robot with learning and memorizing capabilities that enable it to survive in complex and highly dynamic environments.

Keywords *Aplysia*-like spiking neural network · clustering · forgetting · learning · memory · spike time-dependent plasticity · tree-type memory

1 Introduction

Many real world robotic problems are dynamic and require dynamic algorithms capable of adapting over time. It is obvious that frequent changes in the physical environment are usually very difficult to model. An optimal behavior for one instance might not be optimal in the next instance. Therefore robots with the ability to learn, memorize, and deal with different environments are needed.

In recent decades, many groups of researchers have argued that to build a sufficiently autonomous mobile robot capable of interacting with real world applications, both learning and memorizing algorithms should be combined and concurrently introduced to the robot (Moore, 1990). This is the only way in which the robot can frequently use its history (knowledge) to increase the effectiveness of its adaptive ability and to guarantee its stability. The development of such a system, therefore, is one of the major trends in current robotic

Correspondence to: Dr. Kazuyuki Murase, Department of Human and Artificial Intelligence Systems, Graduate School of Engineering, University of Fukui, 3-9-1 Bunkyo, Fukui 910-8507, Japan.
E-mail: murase@synapse.his.fukui-u.ac.jp. *Tel.*: +81(0)776 27 8774, *Fax*: +81(0)776 27 8420

Copyright © 2009 International Society for Adaptive Behavior
(2009), Vol 17(3): 179–196.
DOI: 10.1177/1059712309105814

Downloaded from http://adb.sagepub.com at Katholieke Universiteit Leuven on March 15, 2010

research that is focusing on moving autonomous robotics closer to interfacing real world applications (Floreano & Mattiussi, 2008).

The ability to learn has been achieved using various types of well-known learning algorithms such as: genetic algorithm (Xiao, Michalewicz, Zhang, & Trojanowski, 1997), reinforcement learning (Lin, 1992), and case-based learning (Vasudecan & Ganesan, 1994). These approaches, however, can only survive at the local level (Nowostawski, Epiney, & Purvis, 2005), that is, they do not adapt to changes in the environment, or they can only operate on a predefined and/or fixed fitness landscape. They are therefore very difficult or even impossible to use in multi-task dynamic environments.

Some evolutionary algorithm (EA) variants have been shown to offer a robust and effective optimization for solving dynamic optimization problems (Branke, 2002) by using a population of candidate solutions. However, training the whole or the majority of each population in the environment can be time consuming and even harmful if the environment changes dynamically, especially when we are dealing with real robotic applications. In addition they have no comparative memory. The only way in which selection operates in such algorithms is in the present (Floreano & Mattiussi, 2008).

Self-organization by Hebbian-rule has also been used to solve dynamic problems, because the robot can adapt within a short time and follow changes in the environment (Nielsen & Lund, 2003). The problem with such a mechanism is that it could produce runaway processes of potentiation and it cannot maintain the stability of neural function (McClelland, 2005). Simultaneous localization and mapping (SLAM) has worked very well for addressing the problem of building a map of an unknown environment and controlling a robot in a dynamic environment (Dissanayake, Newman, Clark, Durrant Whyte, & Csorba, 2001). However, in this mechanism, there were no clear clues for developing any kind of memorizing algorithms.

The case-based reasoning module has been used for adapting robots to dynamic environments using a kind of memory (Likhachev, Kaess, & Arkin, 2002). The problem with this approach is that the forgetting mechanism could negatively affect the robot's performance in some domains (Kira & Arkin, 2004). Cost sensitive reinforcement learning (Tan, 1991) could be viewed as a system with a kind of memory. The robot remembers the results of several perceptual actions to define its internal state. In another sense, however, with this memory the agent has no mechanism for maintaining state information from one overt action to the next. The agent no longer has to determine what to remember and when to forget. It remembers the results of all its perceptual actions and then throws all its memory away when it makes an overt action (McCallum 1996).

The biologically inspired *Aplysia*-like spiking neural network (ASNN), or its expanded version, has proven to be an effective and quick adaptive mechanism for performing robot navigation in a dynamic environment (Alnajjar & Murase, 2006a, 2008a, 2008b). It uses the minimum network structure required for performing the task and it modulates its synaptic weights online during the robot's navigation. The main drawback of this circuitry is that it has no memory, that is, the advantage of the adaptation history (the past knowledge) is not taken into account by the robot. The robot forgets the trained environment immediately after it adapts to a new one. A stable behavior and/or an advance action are, however, hard to create. Working to develop a dynamic memory that can lead to such advantages may therefore give a new direction to the future of robotics research.

Verschure, Voegtlin, and Douglas (2003) have developed a neural model called distributed adaptive controller (DAC), which includes a mechanism for perceptional and behavioral learning. The controller design was based on the assumption that adaptive behavior is mainly the result of three tightly coupled layers: reactive, adaptive, and contextual control. Each of these layers was assigned to deal with a specific task: sensory-motor interaction, learning, and memory, respectively. The work has proven the importance of preserving the robot's behavior to build a robust perceptual learning, and achieve a stable behavior. Verschure and Althaus (2003) have also tried to extend the work to show that the perspectives on intelligent systems offered by old and new AI can be unified by assuming that the knowledge level describes the competence of an intelligent system.

Following this line of research, in this study we intend to develop a hierarchical neural system that includes two layers. The first is to support sensorimotor interaction and behavioral learning; the second is to observe the interrelationship between these behav-

Downloaded from http://adb.sagepub.com at Katholieke Universiteit Leuven on March 15, 2010

iors to shape the robot's memory. In this article we focus on designing a memory that could enhance a traditional robot's control system to enable it to face real world problems.

This article is an extension of the ASNN presented by Alnajjar and Murase (2006a, 2008b), which is used as the first layer of the proposed controller and is augmented by a dynamic memory as the second layer. We have called the entire system a dynamic memory ASNN (DMASNN). We believe that the memory proposed in this study can be applied to any type of network architecture or learning algorithm. However, in this study we examined its ability with ASNN, since ASNN has been proven to be an appropriate quick adaptive controller for physical mobile robots (Alnajjar & Murase, 2006a, 2008b).

The article is organized as follows. Section 2 gives a brief background to the work. Section 3 describes the working mechanism of the low- and high-level units in the DMASNN. Section 4 describes the robot and the environments. Section 5 describes the experimental setup and results. A brief discussion and comparison with earlier reported works are given in Section 6. Finally, Section 7 offers some general conclusions and directions for future research.

2 Background

We believe that the ability of any organism to survive in a dynamic environment depends mainly on the number of experiences that the agent has learned and retained in its memory (Dempsey, Vaidya, & Cheng, 2003), which obviously controls its behavior (Noice & Noice, 2006) and/or may even be responsible for its survival (Glickman, Balthrop, & Forrest, 2005). The way an agent learns the behavior, organizes it, and stores it in its memory could therefore be a measure that reflects the level of the intelligence of the agent (Craik & Lockhart, 1972).

Recently, the importance of developing an autonomous mobile robot capable of learning and organizing its own experiences in a virtual memory space has been of interest to many groups of researchers (Gantovnik, Anderson-Cook, Gurdal, & Watson, 2003; Kira & Arkin, 2004). Although the ability to learn has been well developed, there has been relatively little work combining learning and memory in one control system (Floreano & Mattiussi, 2008). To date there is no clear idea as to how the actual biological memory works, therefore this area of investigation is still open.

In this article, we propose a novel quick self-adaptation system with both learning and memorizing abilities that can build an autonomous mobile robot capable of surviving in real dynamic environments. The novelty of our work relies mainly on the proposed high-level memory. In this work, the robot can build online a number of biologically inspired ASNNs, each of which allows it to adapt to one or a group of similar environments and the entire network can remember and manage past experiences and switches between them for optimal navigation.

To achieve the maximum capacity of the working mechanism of the proposed algorithm, our framework is constructed as a hierarchical network architecture. The advantage of this type of architecture, its ability to cope with complex robot behaviors, has been verified by many researchers (Albus, 1999; Kumar & Stover, 1998; Paine & Tani, 2005; Stening, Jacobsson, & Ziemke, 2005; Tani & Nolfi, 1999). It works by decomposing complicated behaviors into many simple behavior modules and solving each model independently in each layer (Tani & Nolfi, 1998). The higher layers in such a network architecture usually supervise the activation of lower layer networks which have only local tasks to perform.

The network model used in this study is represented by two-level units. The low-level (learning level) contains a number of ASNNs each of which adapts to a local environment. The high-level (memory level) keeps tracking the functionality of the robot's behavior in each environment and supervises the activation of the networks in the low-level.

The experimental results show the utility of our adaptive controller. The robot could design and control its memory (behavior) based on its experience to speed up its adaptation to new environments. We believe that our model has self-adaptability, because it works without direct user interaction.

3 Dynamic Memory *Aplysia*-Like Spiking Neural Network (DMASNN)

In this section we describe the details of the two-level units in the hierarchical network architecture and the utilities of each unit.

Downloaded from http://adb.sagepub.com at Katholieke Universiteit Leuven on March 15, 2010

3.1 Low-Level Unit for Robot's Local Tasks

The low-level unit is designed to handle the robot's local tasks in a local open environment; in this study, the task is to perform navigation and obstacle-avoidance behavior. This level consists of a number, *N*, of ASNNs, where the value of *N* depends on the amount of experience the robot acquired. Each of these networks is trained and becomes an expert in a particular environment(s). Each environment is identified by its surrounding brightness, obstacle color, and sound, which can be detected by the high-level unit sensors. Building a new network or switching between existing networks in this level is managed by the high-level unit.

The ASNN used in this level is an effective adaptive controller for an autonomous mobile robot (Alnajjar & Murase, 2006a, 2008b). We have demonstrated the validity of this circuitry as an adaptive controller in a real mobile robot that successfully performed obstacle-avoidance behavior in an open environment (Alnajjar & Murase, 2006a, 2008b). Here we briefly describe the structure and function of this network.

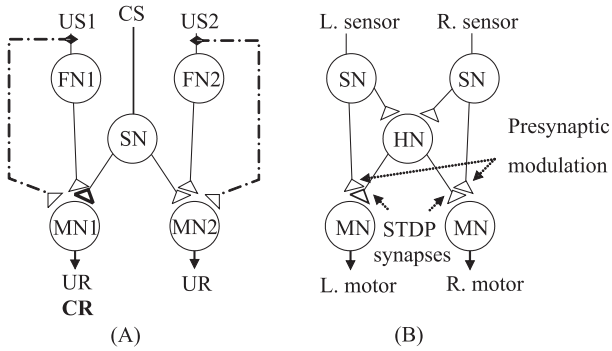


Figure 1 (a) The schematic model of heterosynaptic facilitation of sensorimotor connection in *Aplysia*. (b) The implementation to an adaptive mobile robot controller. CR, conditioned response; CS, conditioned stimulus; FN, facilitatory neuron; HN, hidden neuron; MN, motor neuron; SN, sensory neuron; UR unconditioned response; US unconditioned stimulus. (Adapted from Alnajjar & Murase, 2008b.)

3.1.1 ASNN The structure of ASNN was inspired by the siphon-gill and tail-siphon withdrawal reflex circuits of *Aplysia* (Squire et al., 2007); more specifically, by the associative facilitation mechanism of the sensorimotor connection that underlies the short- and long-term sensitization of *Aplysia*.

Aplysia is a marine snail with a minimal structure containing a basic mechanism for learning and memory. The schematic model of heterosynaptic facilitation of sensorimotor connection in *Aplysia* is shown in Figure 1a (modified from Squire et al., 2007). A sensory neuron (SN) makes synaptic contacts on two motor neurons, MN1 and MN2. When US1, one of the unconditioned stimulus pathways, is activated by stimulation of the animal, the corresponding motor neuron MN1 is activated and a withdrawal reflex is elicited (unconditioned response, UR). The unconditioned stimulus US1 also activates a facilitatory neuron FN1 that makes a presynaptic contact (synapse-on-synapse contact) onto the SN's synapse on MN1.

The facilitation of the sensorimotor reflex is based on an activity-dependent heterosynaptic neuromodulation. The presynaptic terminals from FNs release presynaptic neuromodulator(s), such as serotonin. When a conditioned stimulus (CS) is paired with the US1, the SN's terminal on MN1 is selectively sensitized because of the coincidence of its activity with the neuromodulator action. Now, CS alone is sufficient to activate MN1 by releasing a larger amount of neurotransmitter, which elicits the conditioned response CR. In contrast unconditioned stimulation US2, which is unpaired with the CS, produces no change in the SN's synapse on the corresponding motor neuron MN2. The coincidence of CS and US is thus detected by a presynaptic mechanism. The detection by a postsynaptic mechanism that is also present in *Aplysia* is omitted here for simplicity.

To fabricate such a network in a real mobile robot, several schemes are possible. The simplest circuitry is shown in Figure 1b. The sum of both sensors activities, represented by the activity of the SN, plays the role of the CS. The coincidence with the left/right sensor's activity enhances the synaptic transmission from the SN to the MN1/MN2, thus the signal to the left/right motor is potentiated. Notice that the pathways that activate MNs by USs (dash-dot lines in Figure 1a) are now present as the paths from the SNs to the MNs through the hidden neuron (HN) in Figure 1b. The left and right sensory signals in the robot thus have to take on the role of conditioning stimuli for the signals to left and right motors, respectively.

Downloaded from http://adb.sagepub.com at Katholieke Universiteit Leuven on March 15, 2010

3.1.2 Spike Timing-Dependent Plasticity (STDP) With the circuitry described above, we used a model of synaptic plasticity for the heterosynaptic facilitation at the sensorimotor connections called the spike timing-dependent plasticity (STDP).

STDP is a form of synaptic modification rule found recently in the natural synapses (Song, Miller, & Abbott, 2000). STDP differs from EAs by modifying synaptic weights. Instead of using evolutionary operators such as crossover and mutation, and retraining each individual in the environment, STDP modifies the synaptic weights online during the robot's navigation based on the pre- and post-neuron firing rates. Inputs that fire the postsynaptic neuron with short latency or that act in a coherent manner develop strong synaptic connections, and vice versa (Song et al., 2000). In other words, synapses modifiable with STDP compete for control of the timing of postsynaptic action potentials to be more sensitive to the presynaptic action potentials.

There are two types of computational models of STDP (Song et al., 2000). The type that based on neuron firing rates is used in this study. The synaptic strengths of the postsynapses are modified in accordance with the difference in average spike-time intervals at the presynaptic sites and presynaptic modulator input from the input-layer neurons. Here, the STDP takes place in accordance with $\Delta t_{\text{pre-post}}$, Equation 1, where Δt_{pre} referrers to the firing rate of the hidden-layer neuron and Δt_{post} to that of the output-layer neuron; N is the number of spikes in each time period. The postsynaptic weight is modified by (–/+0.1), to drive $\Delta t_{\text{pre-post}}$ to a minimum value (see Figure 2). More details can be found in Alnajjar and Murase (2008b).

$$\Delta t_{\text{pre-post}} = \frac{\sum_{i=0}^{N} \Delta t_{i(\text{pre})}}{N-1} - \frac{\sum_{i=0}^{N} \Delta t_{i(\text{post})}}{N-1} \tag{1}$$

After the synaptic modification, the postsynaptic neuron tends to respond more quickly to the presynaptic spiking. In our early work STDP was successfully used in adapting the ASNN to an open environment and to following the changes in that environment (Alnajjar & Murase, 2006a, 2008b).

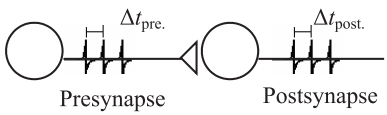


Figure 2 A type of STDP based on the difference in firing rates in pre- and postsynaptic sites.

3.1.3 Spike Response Model (SRM) We have chosen the spiking response model (SRM), a type of SNN model**,** to present ASNN. This model is easy to understand and to implement, especially when it combines with the mechanism of STDP. More details about this model can be found in Alnajjar and Murase (2006b) and Floreano, Zufferey, and Nicoud (2005).

In SRM, a single variable υ_i (Equation 2) that describes the state of a neuron is defined. In the absence of a spike, the variable υ_i is in its resting value, zero in this study. Each incoming spike generates a postsynaptic potential that takes time to return to zero (see Figure 3).

$$\upsilon_i(t) = \sum_j \omega_j^t \sum_f \varepsilon_j(s_j) + \sum_f \eta_i(s_i) \tag{2}$$

$$\varepsilon(s) = \exp[-(s-\Delta)/\tau_m](1 - \exp[-(s-\Delta)/\tau_s]) \tag{3}$$

$$\eta(s) = -\exp[-s/\tau_m] \tag{4}$$

The function $\varepsilon(s)$ describes the time course of the postsynaptic response generated by an incoming spike. If the summation of the effects of several incoming spikes reaches a threshold (θ), an output spike is triggered. Once the neuron has emitted a spike, its membrane potential is set to a very low value period, called the refractory period $\eta(s)$, to prevent an immediate second spike, and then it gradually recovers to its resting potential. Notice that during the refractory period spikes are hardly evoked by the input.

In Equation 3, the effect of ε of an incoming spike on the neuron membrane is a function of the differ-

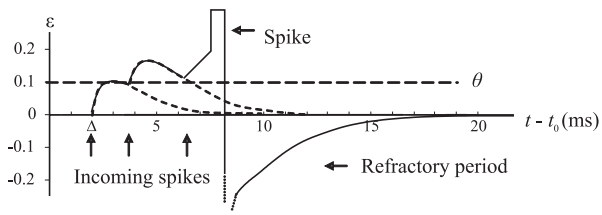


Figure 3 A schematic drawing of a spiking response model (SRM). (Adapted from Alnajjar & Murase, 2006b.)

Downloaded from http://adb.sagepub.com at Katholieke Universiteit Leuven on March 15, 2010

ence ($s = t - t_0$) between the current time t and the firing time t_0 of the neuron. The properties of the function are determined by: (a) the delay Δ between the generation of a spike at the pre-synaptic neuron and the time of arrival at the synapse; (b) a synaptic time constant τ_s; and (c) a membrane time constant τ_m. The function $\eta(s)$ in Equation 4 calculates the refractory period. Finally, ω_j^t is the weight value from the jth synapse on the neuron. In practice, each synaptic potential has to be terminated in a finite time. The values of exponents were truncated for 20 sampling periods in our experiments (Alnajjar & Murase, 2006b).

3.2 High-Level Unit for Learning and Memory

It is well known that as humans learn and remember, the brain does not change its overall structure or grow new neurons; instead the connections between existing neurons are just restructured in response to the current experience (Lehman & Carey, 1986). The memory introduced in this level is developed based on the above phenomenon.

This level is represented by a tree-type memory structure inspired by McCallum's (1996) work. This kind of memory structure has several properties of great practical value, one of which is that the data can be retrieved, modified, and inserted relatively quickly (Gantovnik et al., 2003; Lehman & Carey, 1986). We have intentionally limited the maximum size of the memory to 2^5 nodes (tree's leafs), where the robot can store its experiences (see Figure 4). Each node presents a network connection for one or a group of similar environments. The activation of any node depends on the appearance of its related environment. The robot constructs the tree online, that is, builds a new branch, within its size limit, only when a new environment arises. The memory is divided into four parts. The first three parts are used to identify the current environment based on the camera, light, and microphone sensory inputs of the robot. The last part is used to store each of the network's data (e.g., trained synaptic weights, fitness, and training time).

The working mechanism of DMASNN can be briefly summarized in two main aspects (see the flowchart in Figure 5 for further details):

1. When the robot identifies a new environment, it builds a new hidden node with random pre- and postsynaptic weights, places it into ASNN, trains the network in the environment, and stores its data in its related node. If the environment is changed before the network is completely trained or if the network is not called by the robot for a certain period of time, the connection to its node will gradually fade and the network will be signed as an un-completed trained network (UCTN). Using the above two conditions, we can ensure that most of the time there will be at least a candidate network in UCTN. This phenomenon could be equivalent to the short-term memory in the biological brain because it is not given much time for memorizing and is therefore easily forgotten.
2. When the robot faces an environment that it has already been trained in, the memory searches the tree, beginning at the root and successively moving down, following the sequence of the high-level sensor's input, until it reaches the required leaf with the trained hidden neuron and simply replaces it in the network and continues the training.

We have applied two useful techniques that were incorporated into DMASNN to enhance the memory ability. These techniques were based mainly on tracking the adaptation history of the robot and using it in the future. The first technique, called *environment's similarity marker* (ESM), was designed to speed up the learning mechanism during the robot's navigation. The second, called *forgetting and dynamic clustering mechanisms* (FDCM), was designed to forget and to

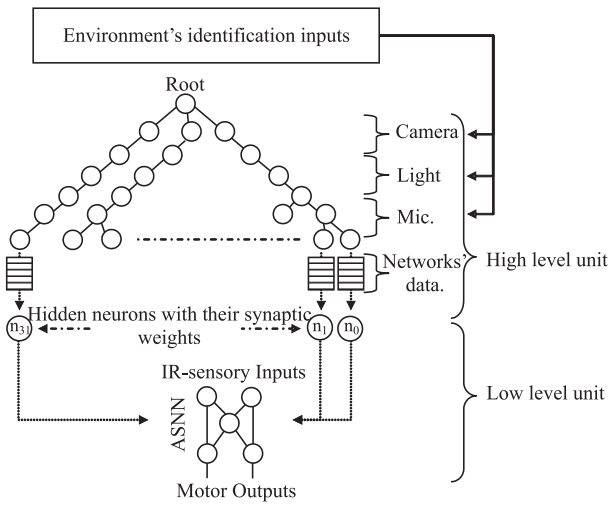


Figure 4 DMASNN layout. Networks' data are stored in the nodes (n_0–n_{31}).

Downloaded from http://adb.sagepub.com at Katholieke Universiteit Leuven on March 15, 2010

cluster some of the nodes inside the memory to guarantee within its maximum. The next two subsections will introduce ESM and FDCM, respectively.

3.2.1 ESM In the belief that similar problems could have similar solutions (Leake & Wilson, 1999), it may be worthwhile using networks that have similar synaptic weights to deal with similarly structured environments, or at least to initialize newly created network synaptic weights from the point where the old trained one has stopped, instead of initializing it randomly. In other words, generate a solution on the basis of already acquired knowledge and experience in the memory.

In this study, therefore, ESM is introduced to the memory. This technique gradually learns how to predict organized synaptic weights based on the current available knowledge when a new ASNN needs to be built. ESM evolves by observing the trained network's behaviors, that is, it evolves based on the correlation between the trained ASNNs, in the low-level, and its related environments. Learning these relationships helps the high-level unit to gradually build managed synaptic weights in the newly created ASNN, which can speed up the robot's adaptation time in new environments. For instance, if a network X_1 trained and performed well in an environment Y_1, and if there is an environment Y_2 which is 80% similar to the environment Y_1, then it may be worth creating network X_2, which is a copy of network X_1, to perform the task in environment Y_2, instead of creating a network Z with initial random synaptic weights and starting the learning from the initial state. In this study, ESM's synaptic weight copy conditions were programmed based on our observation during the experiments as follows:

- If ESM's value ≥ 80%, then the old synaptic weights found will all be copied to the new network and continue the training in the new environment.
- If ESM's value ≥ 60%, then the less activated side of the old synaptic weights found will be randomly regenerated and copied with the other remaining weights to the new network.
- If ESM's value ≥ 40%, then the postsynaptic weights of the old existing network will be randomly regenerated and copied with the other remaining weights to the new network.
- If ESM's value < 40%, then the whole synaptic weight will be randomly generated for the new network.

Figure 5 (dotted boxes) illustrates the working mechanism of ESM. When the environment is changed, ESM stores its maximum fitness, synaptic weights, and the training time in a particular node. Thereafter it compares the synaptic weights with the networks that share the same branch in the memory, and a similarity value is then assigned to this branch's level as in Equations 5 and 6.

$$\mathrm{ESM}_n = \frac{\left\{\begin{array}{l}\text{current}[|\text{left.Pre\_syn.} - \text{left.post\_syn}| \\ \quad + |\text{right.Pre\_syn.} - \text{right.post\_syn}|]\end{array}\right\}}{\left\{\begin{array}{l}\text{old}[|\text{left.Pre\_syn.} - \text{left.post\_syn}| \\ \quad + |\text{right.Pre\_syn.} - \text{right.post\_syn}|]\end{array}\right\}} * 100$$

$$\text{If (current} < \text{old)} \tag{5}$$

$$\mathrm{ESM}_n = \frac{\left\{\begin{array}{l}\text{old}[|\text{left.Pre\_syn.} - \text{left.post\_syn}| \\ \quad + |\text{right.Pre\_syn.} - \text{right.post\_syn}|]\end{array}\right\}}{\left\{\begin{array}{l}\text{current}[|\text{left.Pre\_syn.} - \text{left.post\_syn}| \\ \quad + |\text{right.Pre\_syn.} - \text{right.post\_syn}|]\end{array}\right\}} * 100$$

$$\text{If (current} > \text{old)} \tag{6}$$

where "current" represents the last trained network, while "old" represents the previously trained networks that shared the same node with the "current" one. Further explanation is provided by the result of the second experiment (Section 5.2).

3.2.2 FDCM Although it is unfair to limit the amount of experience that the robot can learn by limiting the memory size, it is, however, not practical to keep adding nodes to the memory for any incoming experience (the memory could exceed the maximum). The robot therefore needs to control its memory size by knowing what to forget, what to remember, and how to manage its experiences by clustering similar ones to minimize its need for storage space. Information that has greater correlation with others can be mentally connected to the existing related information in the memory. This correlation can be measured by ESM (Equations 4 and 5).

Downloaded from http://adb.sagepub.com at Katholieke Universiteit Leuven on March 15, 2010

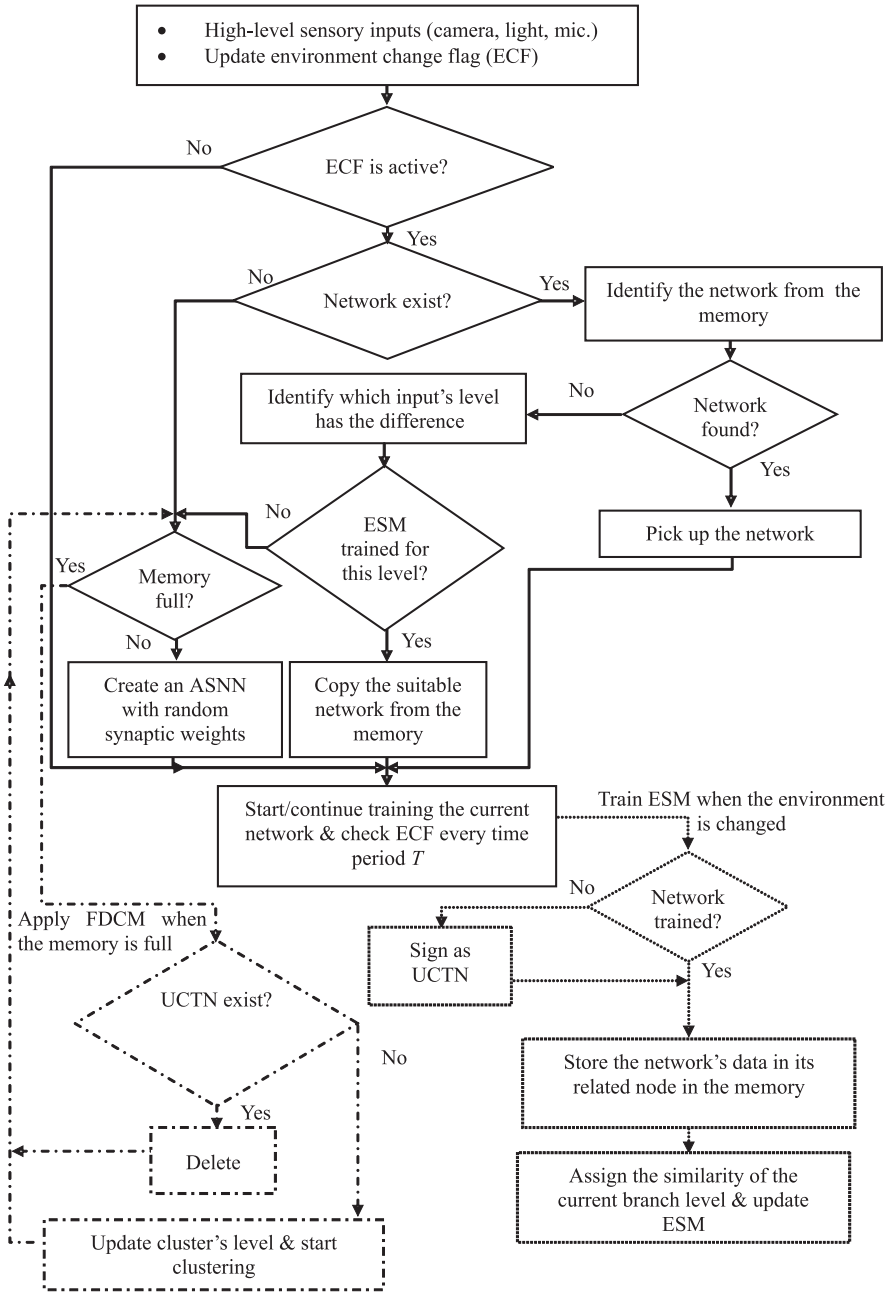


Figure 5 DMASNN's working mechanism. Dotted and dashed boxes represent the working of ESM and FDCM respectively.

In this section, we introduced a FDCM to control DMASNN memory size. See the dashed boxes in Figure 5. The novelty of this technique is that it combines forgetting and clustering mechanisms (Kira & Arkin, 2004; Zimmermann, 2003). When the memory becomes full, the networks that are marked by UCTN become candidates for gradual removal from the memory until free space is made available for more new experiences. The main reasons for deleting existing UCTN from the memory are:

1. These networks have not obtained much experience in the environment, that is, the robot has not spent much time training them. Therefore, retrain-

Downloaded from http://adb.sagepub.com at Katholieke Universiteit Leuven on March 15, 2010

ing them if their environments are reintroduced will not be very time consuming.

2. Some of these networks could have been initialized with very complex synaptic weights during training. Therefore, regenerating the network at a later time could give the robot a chance to develop a better initial synaptic weight.

If after deleting all the existing UCTN the memory is still full, the dynamic clustering mechanism starts to operate, that is, the connections between the nodes are reorganized. It starts clustering all similar networks in the level ESM_1. The network that has a longer experience time and better fitness than others will survive. The clustering range gradually grows wider (ESM_{n+1}) as the memory gets full, that is, more trained networks with less correlated environments could cluster together to form a group of networks with a wider range of adaptation. Note that increasing the clustering range of the less correlated networks can decrease the robot's performance in some domains, because one network needs to survive in a wide range of environments.

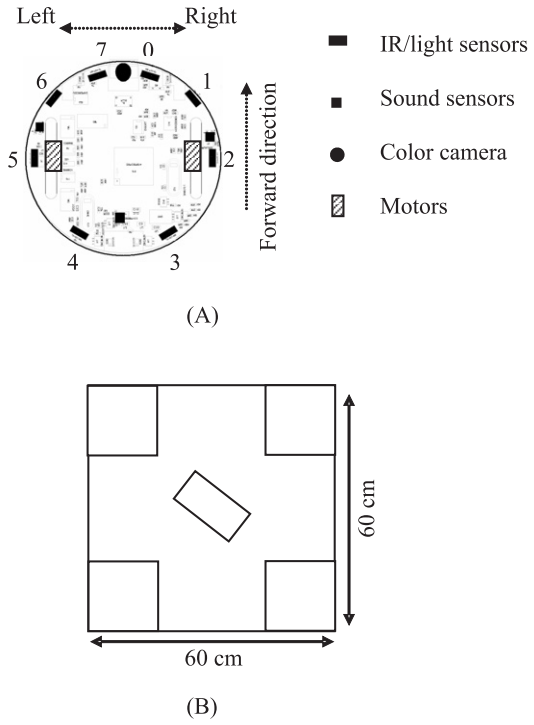


Figure 6 The locations of input sensors and output motors in e-Puck (a). The environment's layout (b).

4 The Robot and the Environment

4.1 The Physical Robot e-Puck

The miniature mobile robot e-Puck was originally developed for educational purposes (http://www.e-puck.org). As illustrated in Figure 6a, it is equipped with a large number of sensors: eight infrared (IR) proximity sensors with a detection distance of 3–4 cm, eight light sensors, three omnidirectional microphones and a color camera with a resolution of 640×480 pixels. As output, in addition to the two wheels e-Puck is equipped with a speaker and LEDs. The communication links supported by e-Puck are the standard RS232, an infrared remote control, and Bluetooth. On Bluetooth, there is a serial line emulation supported by any PC, making communication and the development of PC software simple. In all the experiments reported here, we used the Bluetooth connection to transfer data to and from the robot.

4.2 The Dynamic Environments

In this study, 2^6 different types of physical environments can be built with fixed position obstacles as shown in Figure 6b. Each environment is identifiable by the obstacles' colors, surrounding brightness, and sound. These identifiable marks can be easily recognized by the high-level unit sensory inputs, such as the color camera, light sensors, and microphones, respectively. Each of these sensors were set to give four different binary readings based on its value (see Table 1), and produce an impact to the IR sensors' readings as illustrated in Figure 7. For instance, if the environmental has white obstacles, maximum surrounding light, and maximum sound level, then the high-level's camera sensors, light sensors, and microphones will reflect the values 00, 11, 11, respectively; the robot will identify this environment as (00-11-11). Notice that the obstacles' color, the surrounding light, and the sound level that we use are easy recognizable by the high-level unit sensors, that is, the recognition error is very low.

5 Experiments and Results

In this section, we show the adaptation process of the low-level unit in the local environment, the validity of

Downloaded from http://adb.sagepub.com at Katholieke Universiteit Leuven on March 15, 2010

Table 1 Extracting a binary reading from the high-level robot's sensors.

High-level sensors	Environment's setup	Sensor reading	Sensors representations
Color camera	White obstacles	> 230	00
	Light gray obstacles	[140, 190]	01
	Dark gray obstacles	[80, 120]	10
	Black obstacles	< 50	11
Light sensors	Normal light	> 3950	00
	Extra 40 Volt	[3800, 3900]	01
	Extra 60 Volt	[3400, 3600]	10
	Extra 80 Volt	< 3200	11
Microphones	No sound	< 3%	00
	10 KHz	[5, 20]%	01
	5 KHz	[30, 70]%	10
	300 Hz	> 80%	11

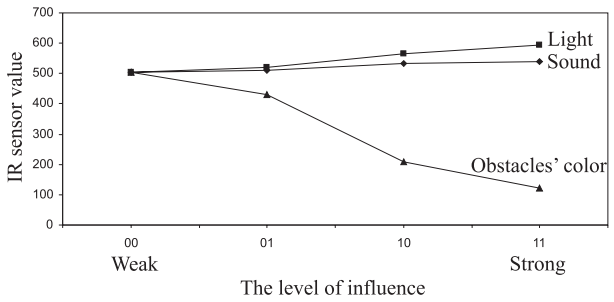


Figure 7 The impact of other sensory inputs on IR sensor readings. Each data point represents an average of five readings of IR sensors that are 2 cm away from an obstacle.

the memory coping with different environments, and the working mechanisms of both *ESM* and *FDCM*.

5.1 Adapting in a Local Environment with a Single Run

This experiment briefly summarizes the adaptation mechanism of ASNN for the robot to perform a navigation task in a local environment with a single run. This mechanism takes place in the low-level unit (Figure 1b).

In such a task, the robot is required to at least have left and right proximity sensors and left and right motors. When the robot recognized a new environment with its high-level sensory inputs, the high-level unit built an ASNN with random synaptic weights unique to this environment. During the robot's life, these synaptic weights were gradually modified by the mechanism of STDP whenever the robot performed any undesired behavior. The modification in synaptic weights depended on the sensitivity between the pre- and postsynaptic firing rates and the effect of presynaptic modulation in the activated sensorimotor side. This modification autonomously generated the obstacle avoidance behavior in the mobile robot. Therefore, while the robot was moving, the activation of the left/right sensor resulted in an increase in the left/right motor activity and a decrease in the right/left motor activity to avoid obstacles on the left/right side of the robot and keep moving forward in the environment (see the demonstration in Figure 8). Further details of this experiment can be found in Alnajjar and Murase (2008b). Figure 9 illustrates the adaptation process of the robot's performance in environment (00-00-00). The performance was

Downloaded from http://adb.sagepub.com at Katholieke Universiteit Leuven on March 15, 2010

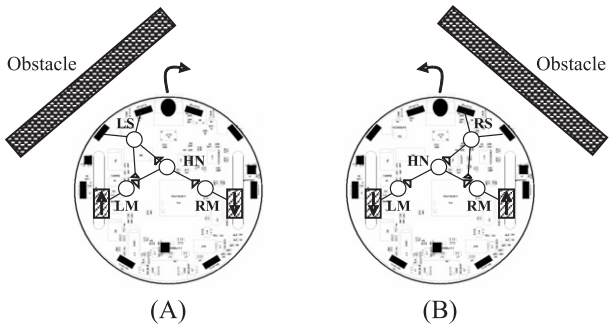


Figure 8 The final robot's behavior for avoiding left (a) or right (b) obstacles. Regarding the synaptic weight modifications done by STDP, in (a)/(b), the firing rate of the left/right motor neuron LM/RM is larger than the firing rate of right/left motor neuron RM/LM when the left/right sensors are activated LS/RS, respectively. The arrows inside the motors illustrate the motors' movement direction.

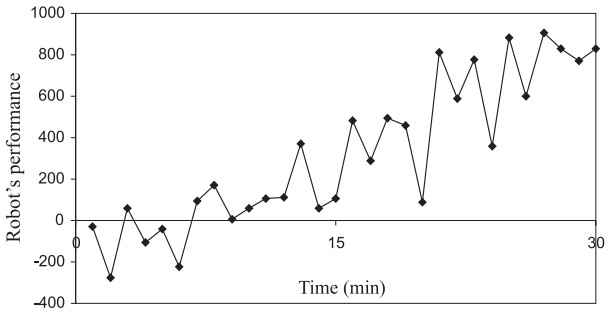


Figure 9 The adaptation process of the robot's performance in environment (00-00-00) for 30 min. The performance was measured every 60 s for a single trial.

measured by the rewards that the robot earned during its training as follows:

- +30 points, if both wheels were moving forward;
- +15 points, if the robot was turning right (left) to avoid a left (right) obstacle;
- –30 points, if the robot was turning right (left) to avoid a right (left) obstacle;
- –10 points, if robot was turning itself around or moving backward.

5.2 Memorizing, Learning, and Switching Mechanisms

This experiment was conducted to evaluate the performance of the high-level unit (robot's memory) in memorizing different experiences and its ability to switch between these experiences based on the current situation. It was also designed to examine whether or not the learning performance of ESM can speed up the adaptation time of the robot in new environments.

In this experiment, seven randomly generated physical environments were introduced sequentially to the robot for 30 min each. Each environment was identified by its obstacle's color (O), surrounding brightness (B), and sound (S), (see Table 2). In each environment, the robot was left to navigate freely until the desired behavior was obtained and then it was moved directly to the next environment. The latest memory structure is shown in Figure 10. Notice that from those introduced environments, the levels of ESM_2, ESM_4, and ESM_6 were trained by Equations 5 and 6. ESM_6 was trained after the training in environment 2 was completed, ESM_4 was trained by environment 4, and ESM_2 was trained by environments 3, 5, and 7.

In this experiment, we also monitored the robot's performance in each environment. The results are shown in Figure 11. From the figure, we can clearly see that the proposed algorithm can substantially improve the performance of the robot in a dynamic environment. In other words, the robot used its past knowledge to deal with future situations: (a) For repeated environments the robot simply switched to the desired network and continued the training (e.g., environment 6); (b) For new environments, after some of the ESM_i levels were trained, the robot used these trained levels to predict networks having weights that were close to the optimal ones (e.g., environments 5

Table 2 Seven environments introduced to the robot.

Env. No	1 O B S	2 O B S	3 O B S	4 O B S	5 O B S	6 O B S	7 O B S
Env. ID	00 00 00	10 11 10	00 00 10	00 10 00	00 10 10	00 00 10	10 11 00

Downloaded from http://adb.sagepub.com at Katholieke Universiteit Leuven on March 15, 2010

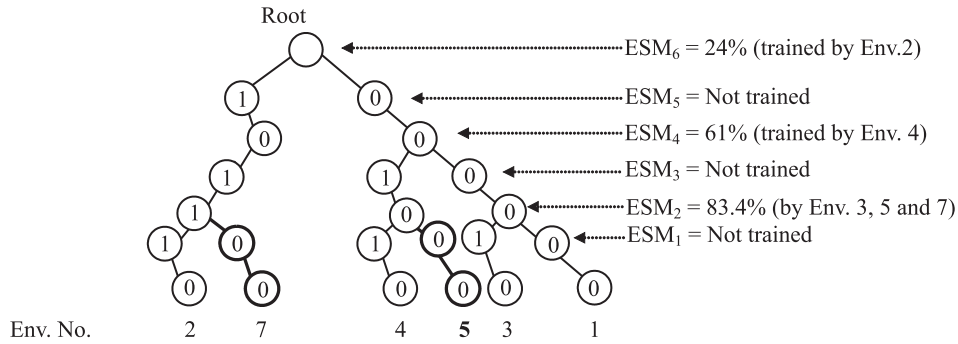


Figure 10 Memory structure after applying seven random environments to the robot. The seven new environments were recognized successfully by the robot. Six were new and one environment was replicated. Each environment was represented by one node at the latest leaf of the tree. Thick lines show the networks that were created by ESM: networks 5 and 7 were copied from networks 4 and 2 respectively, after ESM_2 was trained. When environment 6, which is a duplicate of environment 3, was reintroduced, the robot directly switched to network 3, without building a new network. Latest ESM values are shown at the right side of the figure.

and 7). In both cases the robot could therefore shorten the adaptation time.

We then introduced the same sequence of physical environments (Table 2) to the robot that has ASNN without a memory (Alnajjar & Murase, 2008b). Figure 12 illustrates the results of the robots' adaptation times in both situations. It can be seen from the figure that the robot adapted by DMASNN had used its memory to decrease its adaptation time in the new environments. The network that was generated to navigate in environment 3, for instance, was also used to navigate in environment 6 and, therefore, the adaptation time was very short. In contrast, the robot which had no memory trained in each environment independently, without considering its past knowledge to enhance its ability. For instance, even though environment 6 was a copy of environment 3, which the robot had trained in before, the adaptation times for both environments were almost equal.

5.3 Memory Controlling

In this experiment, we examined the ability of FDCM to control the memory size. We continued the prior experiment by introducing an additional eight random physical environments to the robot. Therefore, in total, 15 physical environments were introduced sequentially to the robot, two of them were replicated and the other 13 were different. With these 13 different environments, all ESM levels were trained by Equations 5 and 6. The trained ESM values are shown in Table 3. Thereafter, because of the difficulty of filling up the memory with real environments, and because of our robot's limited ability, we built a memory simulation and fed it with the trained ESM values (Table 3). For this simulation, we introduced a random number of simulated environments between (00-00-00) and (11-11-11), as well as a random training time for each environment. Table 4 shows the memory performance in the physical and the simulated environments.

From Table 4, we can see that the memory does not delete or cluster any networks if it still has free space (the first three rows in Table 4). When the amount of experience exceeds the memory capacity, forgetting and clustering mechanisms operate in sequence. The robot forgot three environments from the 35 introduced environments without any clustering operation (the fourth row in Table 4). In the fifth row, eight

Table 3 ESM's values.

ESM_n	1	2	3	4	5	6
ESM value	91.5%	82.9%	76%	59.4%	42.5	26.8

Downloaded from http://adb.sagepub.com at Katholieke Universiteit Leuven on March 15, 2010

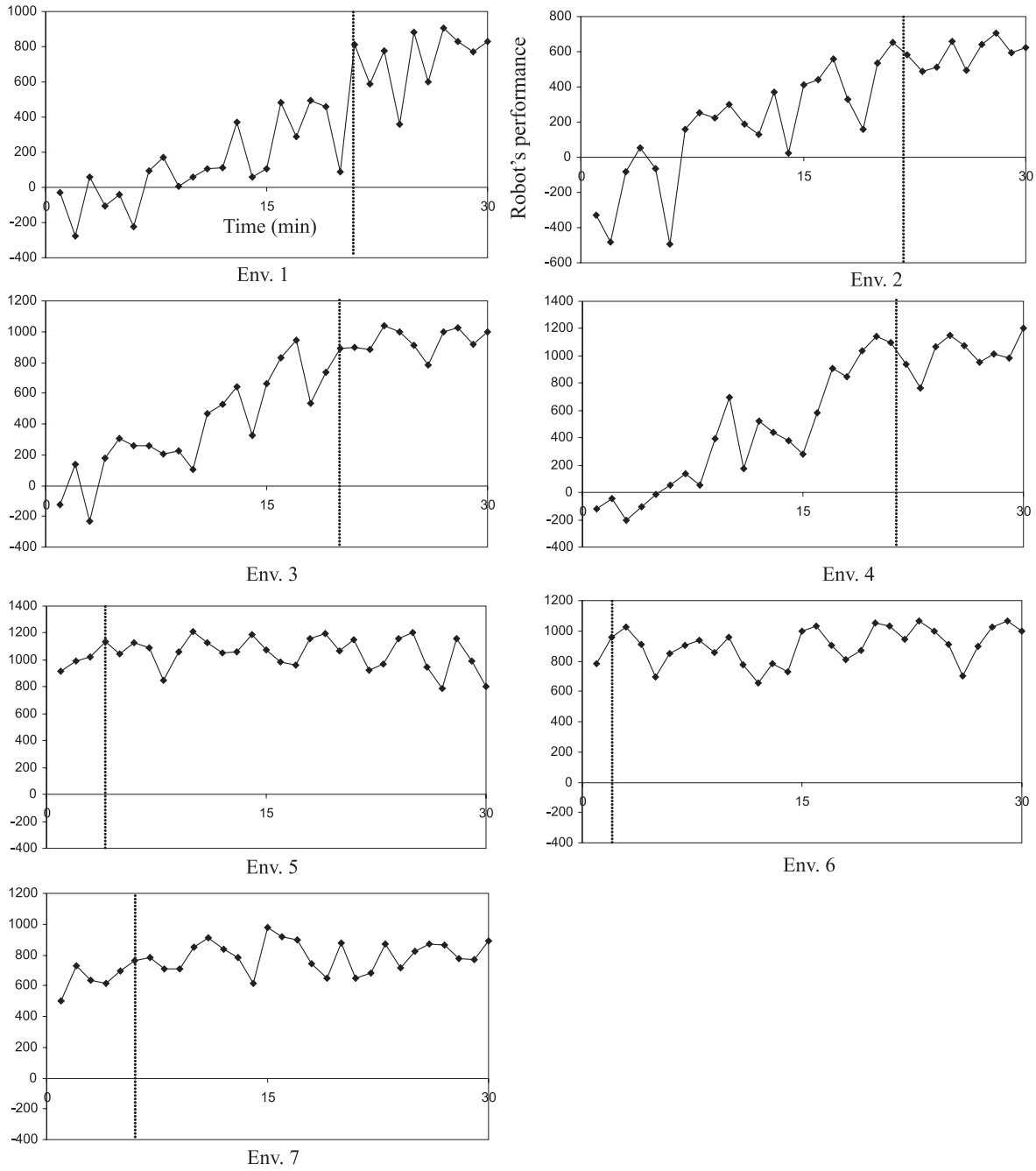


Figure 11 The robot performances in the seven environments addressed in Table 2. Vertical dotted lines illustrate the time needed for the robot to reach the desired behavior.

environments were forgotten and 15 were clustered at (ESM_1 = 91.5%). In the last row, when 61 new environments were introduced, 11 environments were forgotten and 18 were clustered at (ESM_2 = 82.9). The clustering level gradually modified itself based on ESM to overcome the new incoming experiences.

5.4 Validity

Experimental results prove the validity of the proposed controller. Experiment 1 examined the performance of SNN at the low-level. The synaptic weights were adapted online whenever the robot performed unde-

Downloaded from http://adb.sagepub.com at Katholieke Universiteit Leuven on March 15, 2010

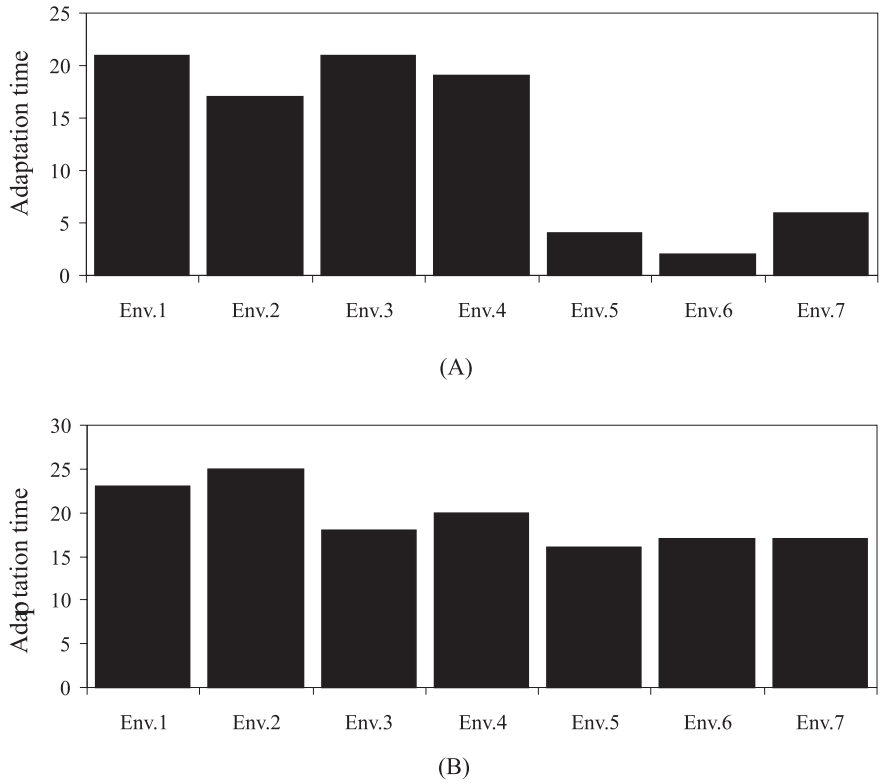


Figure 12 The adaptation time of the robot in the series of seven environments by: (a) DMASNN, (b) ASNN without memory.

Table 4 FDCM and memory performance.

	No. of Env.	New Env.	Repeated Env.	Extra Env.	Forgotten networks	Clustered networks	Clustered threshold level (by ESM's)	Number of trained environments	Number of used nodes
Real	7	6	1	–	0	0	100%	6	6
	15	13	2	–	0	0	100%	13	13
Simulation	32	27	5	–	0	0	100%	27	27
	50	35	15	3	3	0	100%	35	32
	100	55	45	23	8	15	91.5%	55	32
	150	60	90	28	9	19	91.5%	60	32
	200	61	139	29	11	18	82.9%	61	32

sired behavior. The robot adapted to the given environment, navigating smoothly while avoiding obstacles, within a short time.

Experiment 2 showed the performance of the proposed memory at the high-level. The memory constructed its branches online based on the new environ-

Downloaded from http://adb.sagepub.com at Katholieke Universiteit Leuven on March 15, 2010

ments that were identified by the high-level robot's sensors. ESM was also examined at this stage. The correlations between the introduced environments based on their identification and the network behavior, were taken into account. The newly added network was used to update the value of ESM at the related level, which represents the correlations between this network and all networks that share the same ESM level.

Experiment 3 examined the validity of the FDCM. Networks with less performance or training time were more likely candidates to be forgotten from memory when the memory required extra space. If the forgetting process did not fulfill the memory requirement, a dynamic classifying mechanism started to operate. This operation relied on the value of ESM_i, starting from $i = 1$ (minimum), that is, environments with high correlations. Thereafter, it gradually increased to $i =$ maximum, that is, environments with fewer correlations, depending on how much space was needed by the memory. Note that although forcing the memory to cluster a larger number of environments, that is, increased i in ESM_i, could lead to more space in the memory, it could also lead to lower performance and thus may cause confusion in the robot.

6 Discussion

Xiao et al. (1997), Lin (1992), and Vasudecan and Ganesan (1994) have all tried to solve robot navigation tasks with various degrees of success. In these works the mechanisms were implemented at a local level and did not guarantee a stable representation if the input distribution changed over time. Dissanayake et al. (2001) tried to solve a dynamic environment without any memory algorithm. Using this approach, the robot forgot everything it had learned from the environment immediately after it moved to a new one.

Likhachev et al. (2002) used a kind of memory to solve dynamic problems. Although it seems to be successful in their simulation, in order to control the memory size they have applied a kind of forgetting mechanism that is activated when the memory gets full. It is easy to argue that forgetting some cases can have adverse effects on the performance of the robot in some domains. In work reported by Kira and Arkin (2004), the memory stopped accepting any new incoming information if there were no candidate cases to remove. This, of course, limits the robot's learning ability. Zimmermann (2003) applied a kind of memory for dynamic problems with a clustering mechanism to control its size. However, the clustering threshold is fixed and initially predefined. Verschure et al. (2003), built DAC, a hierarchical adaptive controller represented by three main levels to address a new idea for learning and memory. His work proved the importance of attaching a memory to the robot control system to guarantee its stability.

In contrast to the above approaches, in our model we developed a hierarchical adaptive controller with dynamic memory that can: (a) remember the maximum possible experiences that the robot has gone through; (b) learn from the relation between stored experiences to predict advance synaptic weights that are close to the optimal ones, to speed up the adaptation time for new environments; (c) forget only the networks that are not well trained or that initially have bad synaptic weights and take a long time to train; (d) cluster the trained networks that are, to some degree, performing similar behaviors with an online changeable threshold value (ESM) that is based on the current memory capacity. Although our algorithm guarantees to find a space for huge numbers of new environments, it cannot guarantee that the robot will perform well in all of these environments; neither can human memory because one network needs to survive in a wide range of different environments. FDCM has a reverse relationship with the number of introduced environments with respect to the robot's memory size and the correlation between these environments.

We believe that such a consideration can be reasonably accepted in respect of how our brain works. The more the robot increases the clustering level (i.e., ESM_{n++}), the lower performance it can achieve and the more confusion may occur.

7 Conclusion and Future Direction

In this article we described DMASNN, an adaptive controller with dynamic memory for an autonomous mobile robot in dynamic environments. Two-level units were used in the controller: a high-level unit that regulated the robot's memory using a tree-type memory structure and a low-level unit that made the robot adaptable to a local environment by ASNNs.

In this study, two key aspects that enhance the ability of DMASNN were studied: (a) whether the

Downloaded from http://adb.sagepub.com at Katholieke Universiteit Leuven on March 15, 2010

robot was able to learn from its past knowledge to speed up its learning ability in a series of dynamic environments using the environment similarity marker ESM, and (b) how the memory size could be controlled without limiting the robot's learning ability by using the forgetting and dynamic clustering mechanisms FDCM. Our results showed that the proposed DMASNN could consistently adapt well to all the environments tested in a very short amount of time. In addition, it could control the memory size by managing and classifying its experiences. In this model, the robot frequently restructured its memory (behavior) based on its available knowledge.

We believe that the memory component presented in this study is simple and general and can be successfully applied to a broad range of tasks or environments. The low-level unit (learning level), therefore, can be replaced by any type of network architecture or learning algorithm, for example, the work in Zin, Alnajjar, and Murase (2009). In contrast with earlier reported work, which had no clear virtual memory, this model offers a unique ability for memorizing different environments and managing different networks to deal with each environment independently, as well as controlling the memory size.

More experimental data needs to be collected to test the capability of DMASNN, and more formal comparisons with existing similar algorithms are planned for the future. We are also planning to improve decoding and encoding data to and from the memory. The nodes that are repeatedly firing in the memory can improve the robot's skill in some domain, and therefore, build strong experiences which are hard to forget and vice versa for those which are non-periodically firing. We are also planning to implement this algorithm on a large office-like robot (e.g., Robovie-R) with various types of sensory inputs and motor outputs, similar to the network introduce in Alnajjar and Murase (2008a), to evaluate the validity of the proposed adaptive controller.

Acknowledgments

This work was supported by grants to KM from the Japanese Society for Promotion of Sciences, Yazaki Memorial Foundation and by the University of Fukui.

References

Albus, J. S. (2000). 4-D/ RCS Reference model architecture for unmanned ground vehicles. In *Proceedings of IEEE International Conference on Robotics and Automation* (pp. 3260–3265). New York: IEEE.

Alnajjar, F., & Murase, K. (2006a). An autonomous mobile robot controlled by a spike neuron network with one hidden-layer neuron having spike timing-dependent plasticity. In *Proceedings of 3rd International Conference on Soft Computing and Intelligent Systems and 7th International Symposium on Advanced Intelligent Systems* (pp. 207–212), Tokyo, Japan.

Alnajjar, F., & Murase, K. (2006b). Self organization of spiking neural network that generates autonomous behavior in a real mobile robot. *International Journal of Neural Systems*, *16*, 229–239.

Alnajjar, F., & Murase, K. (2008a). Sensor-fusion in spiking neural network that generates autonomous behavior in real mobile robot. In *Proceedings of International Joint Conference on Neural Networks* (pp. 2201–2207).

Alnajjar, F., & Murase, K. (2008b). A simple adaptive controller for autonomous mobile robot: An *Aplysia*-like spiking neural network with one hidden-layer neuron and spike timing-dependent plasticity. *Adaptive Behavior*, *16*, 306–324.

Branke, J. (2002): Evolutionary optimization in dynamic environments. In *Series of genetic algorithms and evolutionary computation*, *Vol.3*. Norwell, MA: Kluwer Academic Publishers.

Craik, F. I. M., & Lockhart, R. S. (1972). Levels of processing: A framework for memory research. *Journal of Verbal Learning and Verbal Behavior*, *11*, 671–684.

Dempsey, P. W., Vaidya, S. A., & Cheng, G. (2003). The art of war: Innate and adaptive immune responses. *Cellular and Molecular Life Sciences*, *60*, 2604–21.

Dissanayake, M. W. M. G., Newman, P., Clark, S., Durrant Whyte, H. F., & Csorba, M. (2001). A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Transactions on Robotics and Automation, 17*, 229–241.

Floreano, D., & Mattiussi, C. (2008). *Bio-inspired artificial intelligence: Theories, methods, and technologies*. Cambridge, MA: MIT Press.

Floreano, D., Zufferey, J. C., & Nicoud, J. D. (2005). From wheels to wings with evolutionary spiking neurons. *Artificial Life, 11*, 121–138.

Gantovnik, V. B., Anderson-Cook, C. M., Gurdal, Z., & Watson, L. T. (2003). A genetic algorithm with memory for mixed discrete-continuous design optimization. *Computers & Structures*, *81*, 2003–2009.

Glickman, M., Balthrop, J., & Forrest, S. (2005). A machine learning evaluation of an artificial immune system. *Evolutionary Computation*, *3*, 179–212.

Downloaded from http://adb.sagepub.com at Katholieke Universiteit Leuven on March 15, 2010

Kira, Z., & Arkin, R. C. (2004). Forgetting bad behavior: Memory management for casebased navigation. *In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems* (Vol. 4, pp. 3145–3152).

Kumar, R., & Stover, J. A. (1998). A behavior-based intelligent control architecture. In *Proceedings of the 1998 IEEE International Symposium on Intelligent Control* (pp. 549–553), Gaithersburg, MD.

Leake, D., & Wilson, D. (1999). When experience is wrong: Examining CBR for changing tasks and environments. In *Proceedings of the 3rd International Conference on Case-Based Reasoning* (pp. 218–232). Springer: Berlin.

Lehman, T. J., & Carey, M. J. (1986). A study of index structures for main memory database management systems. In *Proceedings of the 12th International Conference on Very Large Data Base Endowment* (pp. 294–303).

Likhachev, M. Kaess, M., & Arkin, R. C. (2002). Learning behavioral parameterization using spatio-temporal case-based reasoning. In *Proceedings of IEEE International Conference on Robotics and Automation* (Vol. 2, pp. 1282–1289).

Lin, L. J. (1992). Self-improving reactive agents based on reinforcement learning and teaching. *Machine Learning*, *8*, 293–321.

McCallum, M. K. (1996). *Reinforcement learning with selective perception and hidden state*. Unpublished doctoral dissertation, Doctoral dissertation, Department of Computer Science, Rochester University, New York, USA.

McClelland, J. (2005). How far can you go with Hebbian learning, and when does it lead you astray? In Y. Munakata & M. H. Johnson (Eds.), *Attention and performance XXI: Processes of change in brain and cognitive development*. Oxford: Oxford University Press.

Moore, A. W. (1990). *Efficient memory-based learning for robot control*. Ph.D. Thesis, Technical Report No. 229, Computer Laboratory, University of Cambridge.

Nielsen, J., & Lund, H. H. (2003). Spiking neural building block robot with Hebbian learning. In *Proceedings of International Conference in Intelligent Robots and Systems* (pp. 1363–1369). Las Vegas, Nevada.

Noice, H., & Noice, T (2006). What studies of actors and acting can tell us about memory and cognitive functioning. *Current Directions in Psychological Science*, *15*, 14–18.

Nowostawski, M., Epiney, L., & Purvis, M. (2005). Self-adaptation and dynamic environment experiments with evolvable virtual machines. In: S. Brueckner, G.Di. Marzo Serugendo, D. Hales, & F. Zambonelli (Eds.), *Proceedings of the 3rd International Workshop on Engineering Self-Organizing Applications* (pp. 46–60). Berlin: Springer-Verlag.

Paine, R. W., & Tani, J. (2005). How hierarchical control self-organizes in artificial adaptive systems. *Adaptive Behavior*, *13*, 211–225.

Song, S., Miller, K. D., & Abbott, L. F. (2000). Competitive Hebbian learning through spike-timing dependent synaptic plasticity. *Nature Neuroscience*, *3*, 919–926.

Squire, L., Bloom, F., McConnell, S., Roberts, J., Spitzer, N., & Zigmond, M. (2007). *Fundamental neuroscience* (2nd ed.) (pp. 1277–1283). San Diego: Academic Press.

Stening, J., Jacobsson, H., & Ziemke, T. (2005). Imagination and abstraction of sensorimotor flow: towards a robot model. In *Proceedings of the Symposium on Next Generation Approaches to Machine Consciousness* (pp. 50–58), UK.

Tan, M. (1991). Cost-sensitive reinforcement learning for adaptive classification and control. In *Proceedings of the 9th National Conference on Artificial Intelligence* (pp. 774–780). San Jose, CA: AAAI Press.

Tani, J., & Nolfi, S. (1999). Learning to perceive the world as articulated: an approach for hierarchical learning in sensory-motor systems. In R. Pfeifer, B. Blumberg, J. A. Meyer, & S. W. Wilson (Eds.), *Proceedings of 5th International Conference on Simulation of Adaptive Behavior* (pp. 270–279). Cambridge, MA: MIT Press. (The revised version is in *Neural Networks*, *12*, 1131–1141, 1999.)

Vasudecan, C., & Ganesan, K. (1994). Case-based path planning for autonomous underwater vehicles. In *Proceedings of IEEE International Symposium on Intelligent Control* (pp. 160–165).

Verschure, P. F. M. J., & Althaus, P. (2003). A real-world rational agent: Unifying old and new AI. *Cognitive Science*, *27*, 561–590.

Verschure, P. F. M. J., Voegtlin, T., & Douglas, R. J. (2003). Environmentally mediated synergy between perception and behavior in mobile robots. *Nature*, *425*, 620–624.

Xiao, J., Michalewicz, Z., Zhang, L., & Trojanowski, K. (1997). Adaptive evolutionary planner/navigator for mobile robots. *IEEE Transactions of Evolutionary Computation*, *1*, 18–28.

Zimmermann, A. (2003). Context-awareness in user modeling: Requirements analysis for a case-based reasoning application. In K. D. Ashley, & D. G. Bridge, (Eds.), Case-based reasoning research and development. *Lecture Notes in Artificial Intelligence*, 2689/2003, (pp. 718–732). Berlin: Springer-Verlag.

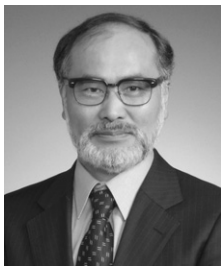
Zin, I., Alnajjar, F., Murase, K. (2009) Adaptation of real autonomous mobile robot in complex environment using pattern association network controller (PAN-C). *Journal of Advanced Computational Intelligence and Intelligent Informatics*, *13*(3).

Downloaded from http://adb.sagepub.com at Katholieke Universiteit Leuven on March 15, 2010

About the Authors

Fady S. K. Alnajjar is a Ph.D. student in the department of System Design Engineering, University of Fukui, Japan. He received a master degree in the Department of Human and Artificial Intelligence Systems in the same university (2007). His main research interest is developing a biologically inspired adaptive controller with dynamic memory for a physical mobile robot. Address: University of Fukui, 3-9-1 Bunkyo, Fukui 910-8507, Japan. E-mail: fady@synapse.his.fukui-u.ac.jp

Indra Bin Mohd Zin obtained a BE degree in the Department of Human and Artificial Intelligence Systems at the University of Fukui in March 2008. He continues his study for Master degree in the same institution. He is interested in building autonomous behavior of physical robots.



Kazuyuki Murase has been a Professor at the Department of Human and Artificial Intelligence Systems, Graduate School of Engineering, University of Fukui, Fukui, Japan, since 1999. He received an M.E. in electrical engineering from Nagoya University in 1978, and a Ph.D. in biomedical engineering from Iowa State University in 1983. He became a research associate at the Department of Information Science of Toyohashi University of Technology in 1984, an associate professor at the Department of Information Science of Fukui University in 1988, and a professor in 1992. He is a member of the Institute of Electronics, Information and Communication Engineers (IEICE), the Japanese Society for Medical and Biological Engineering (JSMBE), the Japan Neuroscience Society (JSN), the International Neural Network Society (INNS), and the Society for Neuroscience (SFN). He serves on the Board of Directors of Japan Neural Network Society (JNNS), as a Councilor of Physiological Society of Japan (PSJ) and as a Councilor of Japanese Association for the Study of Pain (JASP).

Downloaded from http://adb.sagepub.com at Katholieke Universiteit Leuven on March 15, 2010