https://doi.org/10.20965/jaciii.2009.p0312

Paper:

Adapting Real Mobile Robots to Complex Environments Using a Pattern Association Network Controller (PAN-C)

Indra Bin Mohd Zin*, Fady Alnajjar, and Kazuyuki Murase**,*****

*Industrial Computing Research Group, Graduate School of Information Science and Technology, University Kebangsaan Malaysia, Malaysia
**Department of Human and Artificial Intelligence System, Graduate School of Engineering, University of Fukui
3-9-1 Bunkyo, Fukui 910-8507, Japan
E-mail: {fady, murase}@synapse.his.fukui-u.ac.jp
***Research and Education Program for Life Science, University of Fukui, Bunkyo, Fukui, Japan
[Received November 17, 2008; accepted February 25, 2009]

Adapting real mobile robots to complex or dynamic environments is just one of the many challenges robotics researchers face. The difficulty in such environments is in developing a simple, quick adaptive controller that adapts robots to patterns in these environments, especially when individual patterns require unique behavior from the robot. Although most standard evolutionary algorithms attempt to obtain optimal networks for such environments, this is difficult to attain due to network confusion in adapting and readapting patterns. We propose a simple adaptive controller able to learn and remember. It simplifies environments into simple groups of patterns, each of which the robot can independently learn and memorize. The memory introduced in the controller enhances the robot's ability to track its own experience and to cope with upcoming events. Experimental results show that the controller handles general complexity and gives the robot more adaptability, stability, and autonomy.

Keywords: adaptive controller, learning and memory, symmetrical neural network, pattern association network controller

1. Introduction

Real-world robotic environments are complex and change over time. Behavior optimal at one time may not be so in the next moment. The challenge of such environments is in developing an adaptive controller that copes with patterns usually existing in environments. If individual patterns required totally different behavior from the robot, for example, a controller could find difficulty adapting and readapting among patterns.

Classical learning algorithms such as genetic algorithms (GA) [1–3] and reinforcement learning (RL) [4–7] are powerful in adapting robots to local environments, but do not adapt automatically to changes in the environment or operate only in a predefined or fixed landscape [8]. To avoid these drawbacks, researchers have explored combining learning and memory that copes with real-world complexity [9, 10]. Despite some success, however, efforts remain limited.

We introduce a quick, adaptive controller with learning and memory for mobile robots in complex, highly changeable environments. Our proposed model's novelty lies in 1) simplifying complex environments into simple patterns and 2) training, storing, and recalling individual patterns independently through a dynamic memory.

The controller consists of a pulse differentiation learning algorithm (PDL) and a pattern association network controller (PAN-C), where the memory takes place.

This paper is organized as follows: Section 2 gives a brief background on learning algorithms, then details controller principles. Section 3 reviews the robot and environment. Section 4 discusses the experimental setup and results, and Section 5 presents conclusions and projected work.

2. Learning and Memory

After summarizing learning algorithms commonly used to control mobile robots, we detail our proposed algorithm (**Fig. 1**).

2.1. Background

Most learning algorithms used in adaptive robot controllers rely on a single neural network with a mechanism for synaptic plasticity. During the robot's life, the network adjusts weights to learn behavior that meets most of the patterns repeated in a given environment, i.e., the network shapes itself to fit within introduced patterns [16]. These algorithms tend to ignore pattern complexity in environments and attempt to deal with problems as a whole. In highly complex situations, uncorrelated patterns arise that require behavior totally different from the network to survive, limiting the success of these algorithms [16].

Most researchers dealing with mobile robots work to enhance learning algorithms by either presenting a hierarchical network topology [17], by developing new synaptic plasticity [18], or by attaching memory [19]. These

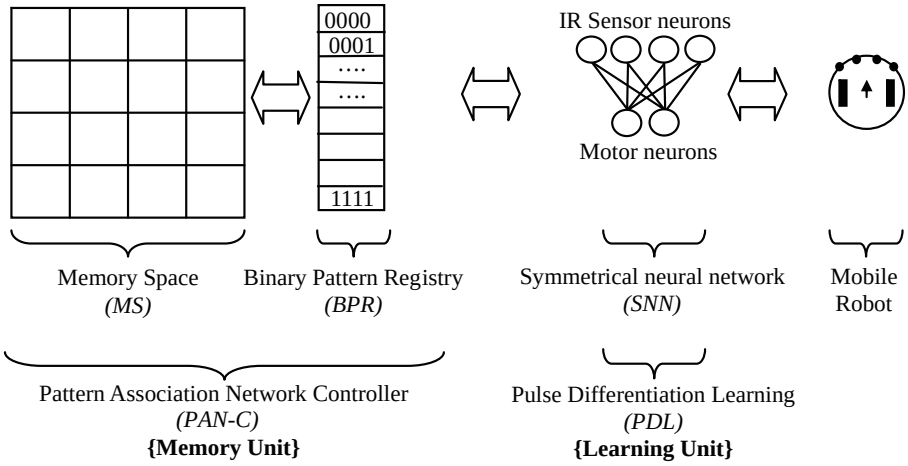

Fig. 1. Proposed controller.

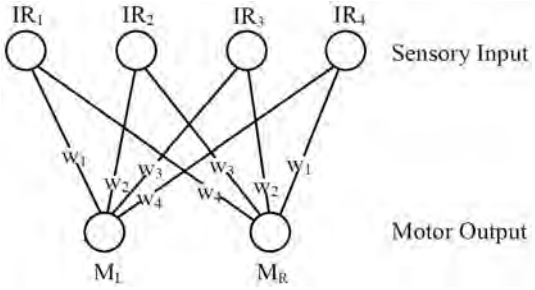

Fig. 2. Two-layer symmetrical neural network.

approaches attempt to maintain trained patterns as long as possible to increase network stability [16].

The novel controller we propose breaks problems down into parts, or patterns, to be dealt with independently. Our controller uses 1) symmetrical neural networks (SNN) with *PDL* for synaptic plasticity (Section 2.2) and 2) a *PAN-C*, which takes into account the degree of correlation between environmental patterns so that the robot can create networks online to ensure its stability and that are generated based on the robot's infrared (IR) sensors during navigation and adjusted by the network's final behavior (Section 2.3).

2.2. Learning Units

2.2.1. Symmetrical Neural Network

A two-layer symmetrical neural network (*SNN*) represents sensorimotor connections (**Fig. 2**) [2, 11]. IR sensory neurons IR_1-IR_4 represent the input layer and directly contact motor neurons M_L and M_R in the output layer. For simplicity, we call synaptic connections from sensory neurons to the left/right motor neurons the left/right synaptic connection $(LSC)/(RSC)$. *LSC* and *RSC* are programmed to hold symmetrical synaptic weights in reverse, i.e., *LSC* from (IR_1-IR_4) is a copy of *RSC* (IR_4-IR_1). The advantage of such symmetrical connection is that changes in synaptic weight on one side cause similar changes on the other, so both sides evolved simultaneously, speeding up adaptation [11].

2.2.2. Pulse Differentiation Learning

Pulse differentiation learning (*PDL*) is a synaptic plasticity model we developed and introduced to the *SNN* (**Fig. 1**). *PDL* updates synaptic weights based on current and a short past history of robot sensory values (x_t), (x_{t-1}, x_{t-2}). We believe that these values impact on what the robot's next time step (x_{t+1}) situation will be. A gradually increase in the robot's sensory values near the side of a wall over time, for example, could lead to a collision with that wall. To avoid this, a certain synaptic weight should be modified, e.g., by increasing motor speed on the wall side and decreasing it on the opposite side, so that the robot will turn away from it [12]. *PDL* is programmed to gradually increase or decrease synaptic weights based on acceleration or deceleration of the short sensor history as follows:

$$W_{t+1} = \begin{cases} W_t + \alpha & : \ddot{x}_i > 0 \\ W_t & : \ddot{x}_i = 0 \\ W_t - \alpha & : \ddot{x}_i < 0 \end{cases} \quad \ldots\ldots\ldots (1)$$

where $\ddot{x}_i$ is the short history of sensor i [$\ddot{x} = (x_t - x_{t-1}) - (x_{t-1} - x_{t-2})$], and the update rate ($\alpha = 0.25$). The synaptic weight range is $[-15, +15]$.

To demonstrate robot network behavior, we calculate motor neuron output to robot motors as follows:

$$M_L = \sum_{i=1}^{n/2} (IR_i.w_i) - \sum_{i=(n/2)+1}^{n} (IR_i.w_i) \quad \ldots\ldots (2)$$

$$M_R = -\sum_{i=1}^{n/2} (IR_i.w_{n-i+1}) + \sum_{i=(n/2)+1}^{n} (IR_i.w_{n-i+1}) \quad (3)$$

where M_L represents the left motor, M_R the right motor, n the number of neurons in the input layer, and w_i synaptic weight.

To evaluate network performance in obstacle avoidance, we introduce a fitness function (Φ) (Eq. (4)). Φ measures the robot's ability to navigate in environments as far away as possible from obstacles within a specific period = 6 seconds. Higher fitness results from high mo-

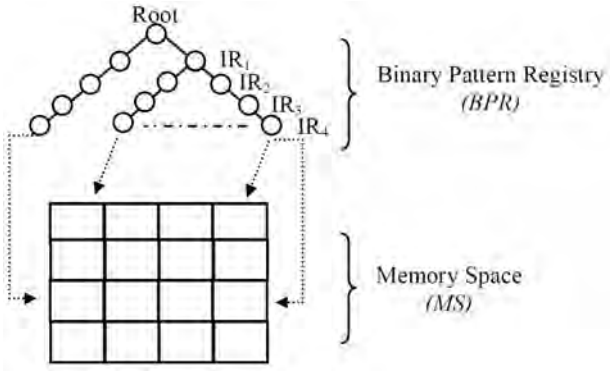

Fig. 3. Pattern association network controller (*PAN-C*). Tree memory represents *BPR*. Each limb ends up by a node that refers to an address or group of addresses in memory space.

tor values and low sensor activity.

$$\Phi = \sum((IR_{max} - IR_i).Distance\ traveled)/time\ . \quad (4)$$

where IR_{max} is a constant representing the maximum IR sensor value ($IR_{max} = 3800$) and IR_i the current sensor value.

2.3. Pattern Association Network Controller

The memory-based pattern association network controller (PAN-C) abstracts patterns from the environment based on sensory input sets, arranges them, and stores them in a binary pattern registry (BPR) (**Fig. 3**). Each of these sets represents a pattern or a group of similar patterns, each connected to a particular set of synaptic weights created and stored in memory space (*MS*). These sets are trained independently in the environment to attain optimal behavior for corresponding patterns.

The controller we propose operates as follows (**Fig. 4**):

1. The robot identifies the current pattern by its sensory input value set.
2. If a pattern was experienced previously, i.e., is in memory space *MS*, the *PAN-C* recalls its corresponding set of synaptic weights and applies it directly to the network.
3. The robot trains the network using *PDL*.
4. Memory saves new fitness values and deletes the untrained network.
5. Since similar sets of synaptic weights lead to similar network behavior, clustering with a threshold (θ) runs offline to cluster similar patterns stored in *MS*. Networks with higher fitness survive.
6. If the current pattern is new to the robot, i.e., does not exist in *MS*, a new address is reserved in *MS* holding a set of synaptic weight specific to this pattern, which then will be run on the network. Go to Step 3.

(1) The *binary pattern registry (BPR)* is binary tree memory at the top of *MS* (**Fig. 3**), inspired by previous

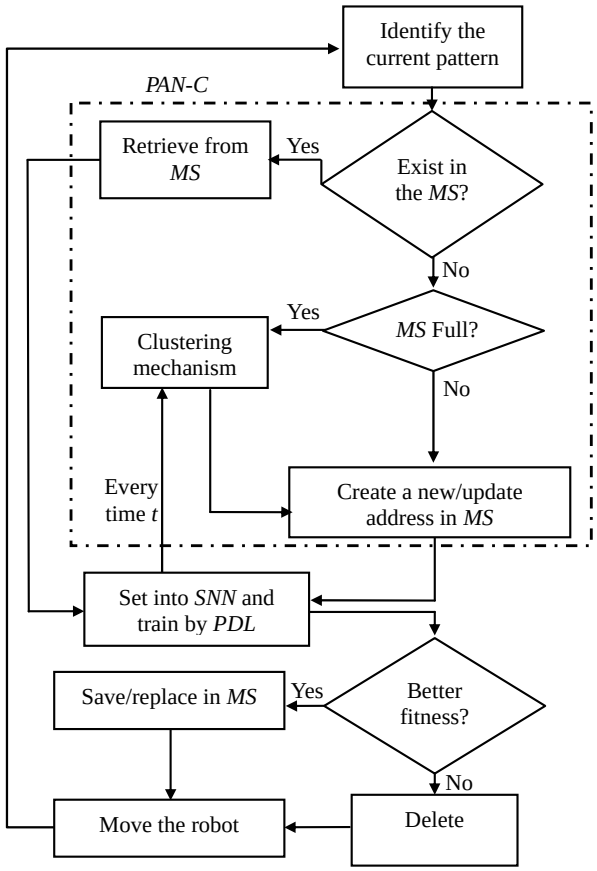

Fig. 4. Controller performance. Dotted line: *PAN-C*.

work [13]. It contains 4 digits, each representing an IR sensory value. The *BPR* is programmed to i) search *MS* for sets of synaptic weights suitable to the current pattern, ii) locate space in MS for new incoming patterns, and iii) cluster similar patterns.

Two-level clustering is applied simultaneously by the *BPR* to control robot memory and give more freedom. Level 1 is applied to *MS*, operating each period ($t = 1.6$ seconds) or whenever *MS* gets full. Networks with similar synaptic weights after training are clustered ($\theta = +/-3$). Frequently called synaptic weights with the highest fitness survive.

Level 2 applies to the *BPR* itself, clustering similar networks using one-step hamming distance for sensory input sets [14].

(2) Memory space (MS) is where the robot stores its experience (**Fig. 3**). The size of *MS* is initially defined by a fixed number of 16 slots, but its capacity is uncertain due to the clustering we use. One slot in memory addresses a variable number of similar patterns.

3. Robot and Environment

To test controller feasibility, we used the differential wheeled mobile robot *e-Puck* (**Fig. 5A**), originally developed for educational purposes [15]. Of the robot's many sensory input devices, we used only the four front IR sen-

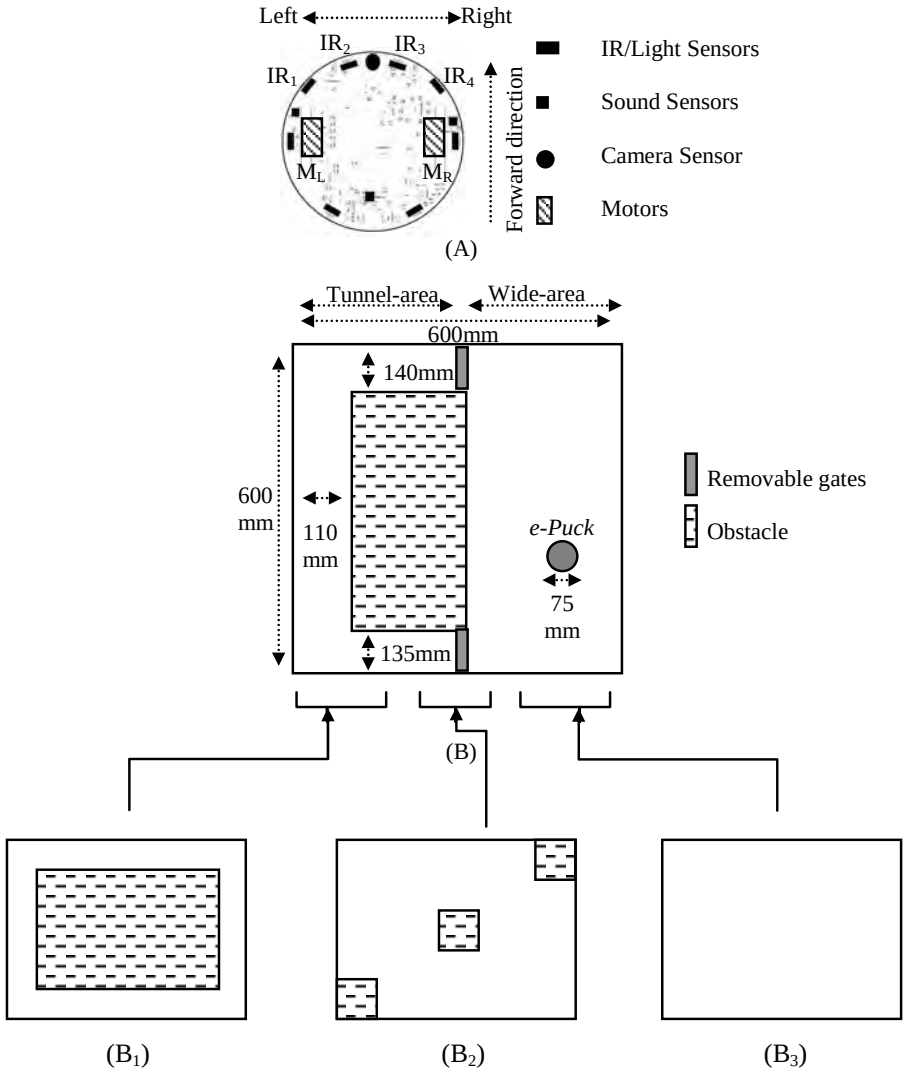

Fig. 5. (A) *e-Puck* sensory input and motor output locations. (B) Complex environment layout combining three environments (B_1, B_2, and B_3).

sors and two wheel motors, since they were sufficient for the robot to manage simple navigation and obstacle avoidance.

A unique complex design environment requiring different behavior from robots (**Fig. 5B**) combines three environments – a simple wide area, a section with different-shaped obstacles, and a set of narrow corridors (**Fig. 5B**$_{1,2,3}$). We divided the environment in two using two removable gates – one containing the simple wide area and the other the remaining areas. Note that removing gates provided a complex environment requiring different skills from the robot (**Fig. 5B**).

Our task was to determine the robot's ability to navigate quickly and consistently in a complex environment.

4. Experimental Results

4.1. Evolution with a Classic Genetic Algorithm

Experiment 1 examined the robot's ability to evolve in a complex environment (**Fig. 5B**), using a classic *GA*.

A simple *GA* program was run on the robot for 60 min, with the *GA* designed to handle 16 individuals per generation. Each individual coded a set of synaptic weights applied directly to the network and trained by the robot in the environment for 6 seconds. Two-point crossover and one mutation were used to build new generations. To evaluate individual, the fitness function (Φ) was calculated each 6 seconds (Eq. (4)).

Behavior fluctuated between adaptations to each pattern (**Fig. 6**). Initially, the robot tried to adapt its network to wide area A, but when it entered tunnel areas B and C, the fitness value suddenly decreased, requiring readaptation, and vice versa when the robot left the tunnel area.

We repeated this experiment with different setups. Gates to the tunnel were closed and the robot started evolving within the wide area for 30 min, then we removed gates and continued the training another 30 min. This scenario was repeated 3 times.

In average behavior for the three runs (**Fig. 9**, thin line), the robot determined the best individual in the wide area within less than 30 min, but after gates were removed and

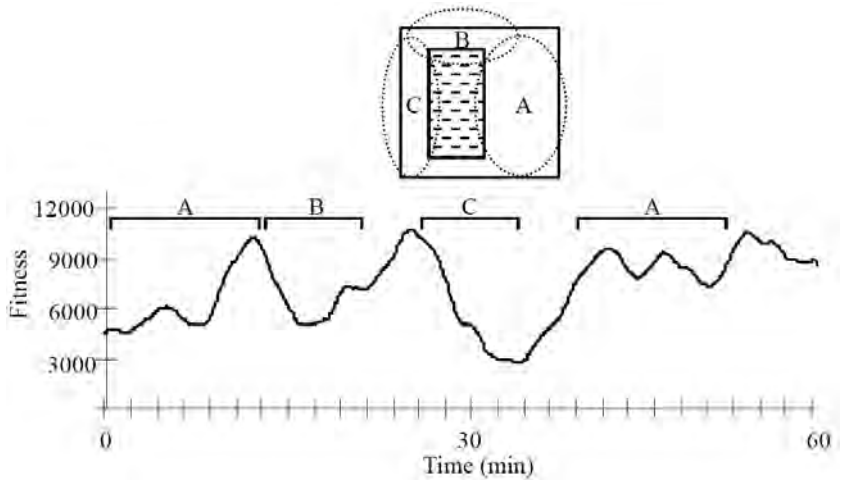

Fig. 6. Single *e-Puck* run evolved by standard *GA* in complex environment.

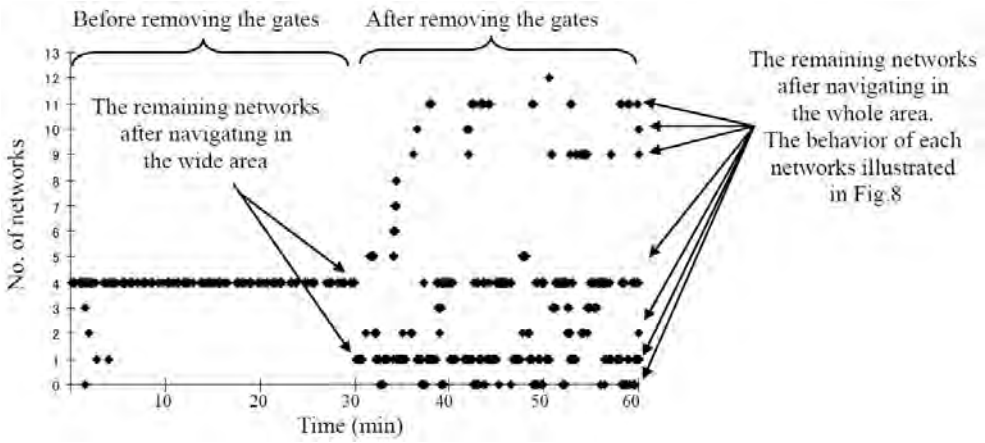

Fig. 7. Networks activity in robot *MS* during complex environment navigation. Each point represents time when a network was activated.

the robot was moved to the tunnel area, the best individual found failed to survive because the new pattern required totally different behavior from the robot. Extra training was therefore required for the robot to fit within the new situation.

We thus concluded most classic known evolutionary algorithms, e.g., *GAs*, were inappropriate for complexity problems, and such algorithms would make it difficult to achieve an optimal network.

4.2. Adaptation with Proposed Controller

Experiment 2 examined the controller's ability to deal with complex environments (**Fig. 5B**).

In a scenario similar to experiment 1, the robot was initialized in the wide area (**Fig. 5B**) with gates to the tunnel area closed and left to move around for 30 min. At the initial time, the robot found 5 patterns based on the set of IR sensor values and corresponding networks were registered sequentially in *MS*. These 5 networks were then clustered into 2 (**Fig. 7**). The robot adapted to navigating smoothly in the wide area, avoiding obstacles within a very short time.

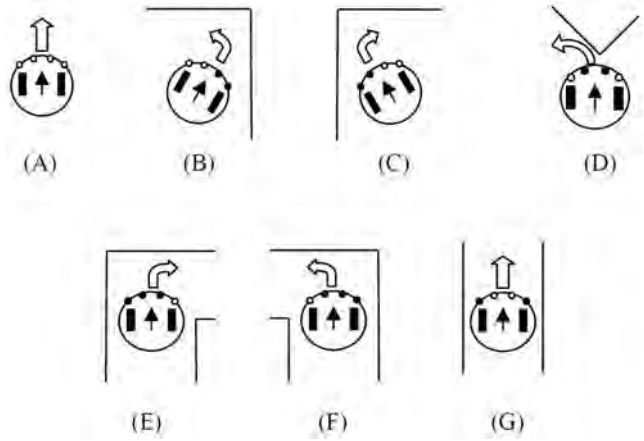

Fig. 8. Final patterns found in environment. Black circles indicate activated IR sensory input.

After 30 min, we removed gates and continued training using the whole area. The robot had learned extra patterns and its networks were sequentially added to *MS* (**Fig. 7**). Each network could cope with one or more behaviors in the environment. Network activation relied on

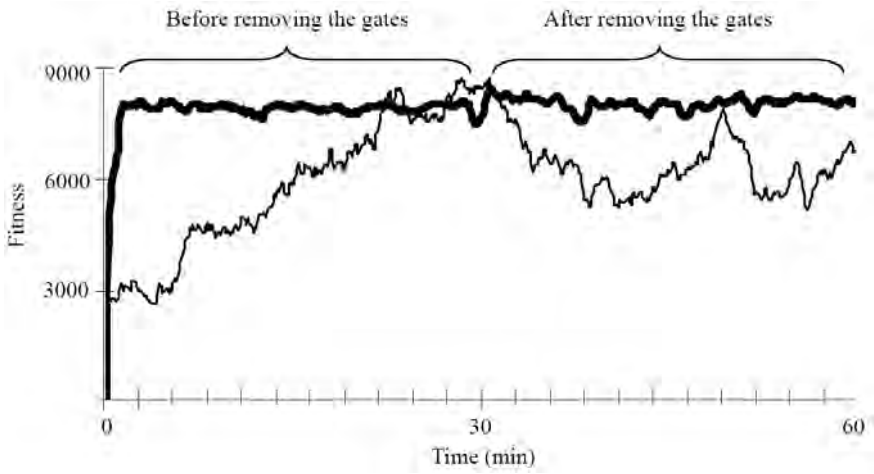

Fig. 9. *PAN-C* adaptation (thick line) and *GA* evolution (thin line). Gates to narrow space were removed after 30 min.

the appearance of the corresponding pattern. Note that 13 patterns were initially generated and clustered into 7 networks. Each network represented a behavior. The latest behaviors in the (**Fig. 8**) resulted eventually in seven networks – (A) for straight navigation, (B) for right and front obstacle avoidance, (C) for left and front obstacle avoidance, (D) for front obstacle avoidance, (E) for turning right in a tunnel, (F) for turning left in a tunnel, and (G) for moving straight in a tunnel.

We repeated the above scenario 3 times. In the average for the 3 runs (**Fig. 9**, thick line), fitness became highest very quickly and remained stable even though situations the robot faced during navigation changed often.

The robot thus found different patterns and learned them independently, storing corresponding networks in *MS* and recalling them as needed.

4.3. Controller vs. GA

As indicated, although the *GA* is powerful for generating autonomous behavior in a mobile robot, its ability to navigate in complex or dynamic environments is limited. Behavior optimal in the wide area failed to survive in the tunnel area, i.e., fitness became low, engendering unstable behavior and finding no optimal network to cope automatically with both environments (**Figs. 6** and **9**).

Our proposed controller, in contrast, abstracted, learned, and saved a number of networks in memory based on IR sensory sets. Each network represented a behavior, and these networks were recalled and retrained whenever the corresponding pattern occurred. The robot thus learned to deal with different patterns independently instead of dealing with the whole environment as one, and no longer had to rely on environment complexity. Instead, it simplified problem complexity into simple patterns and dealt with each independently, gaining for itself a high degree of adaptability, stability, and autonomy.

To reaffirm *PAN-C* efficiency, we divided the environment into two equal areas and represented a series of different environments simultaneously in each. We introduced two equally trained robots, one by a *GA* and the other by the *PAN-C*, into this series of environments for 210 seconds (**Fig. 10**).

We found that:

1. Regardless of synaptic plasticity used in the network, it appeared difficult for a single network to maintain stability and cope rapidly with changes in a series of different environments. The robot trained by the *GA*, for example, kept hitting the wall and reevolved its network whenever changes occurred in the environment. The robot appeared to be confused between adaptations in each situation. In contrast, the robot trained by the *PAN-C* was free to switch among activation of a number of networks that it had previously learned and stored in memory to maintain stability.
2. The robot trained previously by the *GA* appeared to try to avoid entering narrow areas or those it had not previously trained for to avoid losing its fitness, so the areas it covered were very limited. In contrast, the robot trained by the PAN-C tried to cover the whole area by switching between networks and generating new networks as needed, and invariably did its best to keep a certain distance from the wall.

5. Conclusion

We have built a simple adaptive robot controller that handles general complexity, simplifying a given environment into simple patterns, learning these patterns, and dealing with individual patterns independently.

Patterns are recognized by sets of IR sensory input and represented by particular networks stored in dynamic memory space. Networks with similar behavior or close sensory input sets are clustered in two-level clustering. Calling and retraining networks depends on the appearance of their corresponding patterns. Networks are trained independently in the environment and do not affect other network behavior.

Experiments 2 and 3 demonstrate the performance of the proposed controller. Obstacle avoidance and smooth,

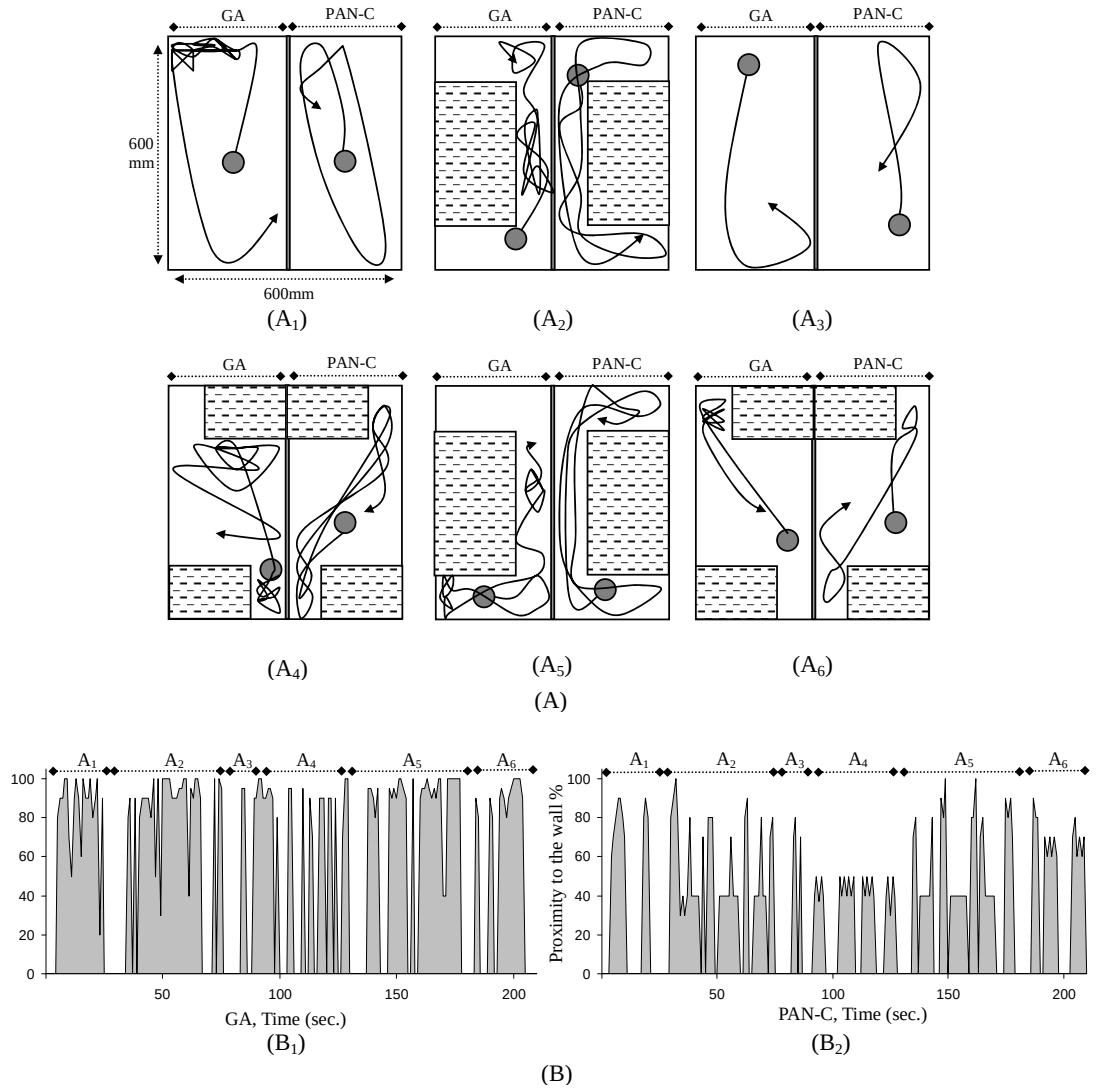

Fig. 10. (A) Robot behavior when introduced to different environments, using the GA (left)/*PAN-C* (right). (B) Wall proximity in robot navigation. 100: Robot collision with wall.

stable navigation occurred more quickly compared to when a GA was used. Optimal behavior was distributed among seven networks so that each represented one behavior and was switched with the other.

Note that our controller both shortened learning and generated overall performance stability while minimizing the number of networks needed to complete a task through simple clustering.

The work presented here is applicable to a broad range of tasks and complex environments and provides an attractive alternative for coping efficiently and quickly with complexity problems.

Although our experiments were simple, however, the basic of experiments 2 and 3 generally clarifies the efficacy of our proposal.

We now plan to expand sensory input patterns to include a video camera for fast, stable navigation based on image processing for a real robot in a dynamic office-like environment. *PDL* is also to be tested to determine its compatibility with other network architectures.

Acknowledgements

This work was supported in part by grants from the Japanese Society for Promotion of Science (JSPS), the Yazaki Memorial Foundation for Science and Technology, and the University of Fukui.

References:

[1] J. Xiao, Z. Michalewicz, L. Zhang, and K. Trojanowski, "Adaptive Evolutionary planner/Navigator for Mobile Robots," IEEE Transactions on Evolutionary Computation, 1-1, pp. 18-28, 1997.

[2] K. Murase, Md. Monirul Islam, T. Hirata, A Matsumoto, H. Akita, T. Asai, K. Sakai, T. Sasajima, M. Naruse, S. Terao, and M. Okura, "Chapter 3: Analysis and production of organized behavior in real environment with a mini-robot Khepera," Evolutionary Robotics III (Applied Artificial Intelligence Book), Ontario, Canada, pp. 81-108, 2000.

[3] S. Nolfi and D. Floreano, "Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-organizing Machines," The MIT Press, 2000.

[4] G. G. Yen and T. W. Hickey, "Reinforcement learning algorithms for robotic navigation in dynamic environments," ISA Transactions, 43-2, pp. 217-230, 2004.

[5] L. J. Lin, "Self-Improving Reactive Agents Based on Reinforcement Learning, Planning and Teaching," Machine Learning, 8, pp. 293-321, 1992.

[6] M. Rodríguez, R. Iglesias, C. V. Regueiro, J. Correa, and S. Barro,

"Autonomous and fast robot learning through motivation," Robotics and Autonomous Systems, 55-9, pp. 737-740, 2007.
[7] S. Bhatnagar and K. M. Babu, "New algorithms of the Q-learning type," Automatica, 44-4, pp. 1111-1119, 2008.
[8] M. Nowostawski, L. Epiney, and M. Purvis, "Self-Adaptation and Dynamic Environment Experiments with Evolvable Virtual Machines," Engineering Self-Organizing Systems, pp. 46-60, Springer-Verlag, 2005.
[9] M. Likhachev, M. Kaess, and R. C. Arkin, "Learning Behavioral Parameterization Using Spatio-Temporal Case-Based Reasoning," Proc. of the IEEE Int. Conf. on Robotics and Automation, pp. 1282-1289, 2002.
[10] M. Tan, "Cost-sensitive reinforcement learning for adaptive classification and control," Proc. of the Ninth National Conf. on Artificial Intelligence, pp. 774-780, 1991.
[11] N. Jakobi, "Minimal Simulations for Evolutionary Robotics," Ph.D. thesis, School of Cognitive and Computing Sciences, University of Sussex, 1998.
[12] F. Alnajjar and K. Murase, "Self organization of spiking neural network that generates autonomous behavior in a real mobile robot," Int. Journal of Neural Systems, 16-4, pp. 229-239, 2006.
[13] F. Alnajjar, I. B. M. Zin, and K. Murase, "A Spiking Neural Network with Dynamic Memory for a Real Autonomous Mobile Robot in Dynamic Environment," Proc. of Int. Joint Conf. on Neural Networks, pp. 2207-2213, 2008.
[14] W. Hamming, "Error Detecting and Error Correcting Codes," Bell System Technical Journal, 29, pp. 147-150, 1950.
[15] Developed by Ecole Polytechnique Fédérale de Lausanne (EPFL) Switzerland 2005. Available: http://www.e-puck.org.
[16] D. Floreano and C. Mattiussi, "Bio-Inspired Artificial Intelligence: Theories, Methods, and Technologies," The MIT Press, 2008.
[17] S. Kim, M. Spenko, S. Trujillo, B. Heyneman, V. Mattoli, and M. R. Cutkosky, "Whole body adhesion: Hierarchical, directional and distributed control of adhesive forces for a climbing robot," In Proc. of the IEEE Int. Conf. on Robotics and Automation, Rome, pp. 1268-1273, 2007.
[18] X. Wang, Z. Hou, A. Zou, M. Tan, and L. Cheng, "A behavior controller based on spiking neural networks for mobile robots," Neurocomputing, 71, 4-6, pp. 655-666, 2008.
[19] Y. Adachi, H. Saito, Y. Matsumoto, and T. Ogasawara, "Memory-Based Navigation using Data Sequence of Laser Range Finder," Proc. of IEEE Int. Symposium on Computational Intelligence in Robotics and Automation, 1, pp. 479-484, 2003.

Name:
Indra Bin Mohd Zin

Affiliation:
Industrial Computing Research Group, Faculty of Information Science and Technology, University Kebangsaan Malaysia

Address:
43650, Bangi, Malaysia

Brief Biographical History:
2008 Received B.E. in Human and Artificial Intelligent System from University of Fukui, Japan

Main Works:
• "Vision-sensorimotor Abstraction and Imagination towards Exploring," in: Proc. of IJCNN2008, pp. 2418-2424, 2008.

Membership in Academic Societies:
• IEEE Student Member

Name:
Fady Alnajjar

Affiliation:
Department of System Design Engineering, Graduate School of Engineering, University of Fukui

Address:
3-9-1 Bunkyo, Fukui 910-8507, Japan

Brief Biographical History:
2004 Received B.E. degree in Computer Since and Engineering from Ittihad University, UAE
2007 Received M.E. degree in Human and Artificial Intelligent System from University of Fukui, Japan

Main Works:
• "A simple Aplysia-like spiking neural network to generate adaptive behavior in autonomous robots," Adaptive Behavior, Vol.14(5), pp. 306-324, 2008.
• "Self organization of spiking neural network that generates autonomous behavior in a real mobile robot," Int. Journal of Neural Systems, Vol.16(4), pp. 229-239, 2006.

Membership in Academic Societies:
• IEEE Student Member
• Computational Intelligence Society (CIS)

Name:
Kazuyuki Murase

Affiliation:
Professor, Department of Human and Artificial Intelligence Systems, Graduate School of Engineering, University of Fukui

Address:
3-9-1 Bunkyo, Fukui 910-8507, Japan

Brief Biographical History:
1978 Received M.E. degree in Elec. Eng. from Nagoya Univ.
1983 Received Ph.D. in BME from Iowa State Univ.
1984 Joined Dept. Information Science at Toyohashi Univ. Tech. as a Research Associate
1988 Joined Dept. Information Science at Fukui Univ. as an Associate Professor, and as a Professor since 1992

Main Works:
• "Chaotic Dynamics of a Behavior-Based Miniature Mobile Robot: Effects of Environment and Control Structure," Neural Networks, Vol.18, pp. 123-144, 2005.
• "A Constructive Algorithm for Training Cooperative Neural Network Ensembles," IEEE Trans. Neural Networks, Vol.14, pp. 820-834, 2003.
• "A New Algorithm to Design Compact Two-Hidden-Layer Artificial Neural Networks," Neural Networks, Vol.14, pp. 95-108, 2001.

Membership in Academic Societies:
• The Institute of Electronics, Information and Communication Engineers (IEICE)
• Japanese Society for Medical and Biological Engineering (JSMBE)
• Japan Neuroscience Society
• International Neural Network Society (INNS)
• Society for Neuroscience
• Board of Directors in Japan Neural Network Society (JNNS)
• Councilor of Physiological Society of Japan (PSJ)
• Councilor of Japanese Association for the Study of Pain