Applying Hybrid Reinforcement and Unsupervised Weightless Neural Network Learning Algorithm on Autonomous Mobile Robot Navigation

Yusman Yusof[1], H. M. Asri H. Mansor[2], H. M. Dani Baba[3]
[1]*Industrial Automation Section, Universiti Kuala Lumpur Malaysia.*
[2]*France Institute, Bandar Baru Bangi, Selangor, Malaysia.*
[3]*Faculty of Electrical Engineering, Universiti Teknologi MARA, Shah Alam, Selangor, Malaysia.*
yusman@unikl.edu.my

Abstract**—An autonomous system constructed using written computer programs based on human expert knowledge only handles anticipated and verified states. On the other hand, a self-learning algorithm allows an autonomous system to instinctively acquire knowledge, learn from experience and be more prepared to expect the unexpected. A novel hybrid self-learning algorithm which combines reinforcement and unsupervised weightless neural network algorithm learning was formulated. The self-learning algorithm was applied to an autonomous mobile robot navigation system in simulation and physical world. The result shows that the simulated and physical robot possesses the ability to self-learn by acquiring knowledge, learn and record experience without having prior information on the environment. The mobile robot was able to distinguish different types of obstacles i.e. corners and walls; and generate complex control sequences of actions in order to avoid these obstacles.**

Index Terms**—Reinforcement Learning; Q-learning; AutoWiSARD; Autonomous Navigation; Unsupervised Learning; Weightless Neural Network; LeJOS; Lego Mindstroms.**

I. INTRODUCTION

This paper describes the research, formulation and implementation of a novel self-learning algorithm. The algorithm was derived from the combination of unsupervised weightless neural network learning, which employs AutoWiSARD [1] and reinforcement learning algorithm, which employs Q-learning [12]. By integrating both algorithm, a system will be able to acquire knowledge, learn, record and recall past experience thus achieving self-learning state.

In contrast to developing an autonomous system using pre-defined expert knowledge which was converted into computer program or by utilizing exhaustively trained and tested Artificial Intelligence (AI) algorithm; self-learning algorithm allows a system to instinctively acquire knowledge, learn from experience and better prepared to expect the unexpected.

In order to study its effectiveness, the formulated self-learning algorithm will first be applied to a simulated mobile robot, then implemented in physical mobile robot. Both in simulation and physical world, the mobile robot will wander in an unknown environment while avoiding obstacles. The mobile robot application was selected because the method of acquiring knowledge and learning from experience while wandering and navigating to avoid obstacles is similar to human learning process [2].

II. LITERATURE REVIEWS

A. Other Researches

Since 2005, several relevant studies were conducted on developing self-learning systems such as by Kamath [3], Guo [4] and others [5]–[7] which employs various AI algorithms such as Experts System, Fuzzy Logic and others. Out of these researches, only Yousif et al [2] develop a self-learning system for an autonomous system. They developed a mobile robot simulation with self-learning capabilities but the technique employs combination of rules based and path planning algorithm that were hard coded into the system thus making the system inflexible.

B. Autonomous System Control Algorithm

An autonomous system that controls a mobile robot navigation can be classified as reactive system which reacts to the changes in the external environment. Reactive system gathers information about the current state/situation by sensing its surrounding environment, and reacts by performing finite number of actions [8].

Based on characteristics of control algorithm which governs reactive systems behaviors, it can be concluded that states of environment will determine sequences of actions to be taken. Therefore, in order to autonomously learn, the formulated algorithm must be able to independently: a) classify new and differentiate existing states; b) determine sequences of actions to be taken.

Close examinations of various autonomous systems developed by others [9]–[17] reveals that both WiSARD [18], a Weightless Neural Network (WNN) algorithm; and Q-learning, reinforcement learning algorithm were applicable for self-learning algorithm formulation. Both algorithm were fast and efficient, can be implemented in resource constrained embedded systems and the combination of these two algorithms are able to classify and create complex control algorithm for robot navigation.

WiSARD algorithm requires exhaustive training of anticipated states as demonstrated by Nurmaini [12] and Mcelroy [11]. As an alternative, AutoWisARD, an unsupervised learning version of WiSARD will be used to autonomously classify states. As highlighted by Sahin et al [19] it is expected that the combinations of two or more intelligent technologies are able to support generalizations and can handle incomplete cases.

III. Self-learning Algorithm

A. Q-learning Algorithm

Figure 1 summarizes the Q-learning learning loop when implemented in a system, whereby:

i. Q(s, a) – component of Q table;
ii. *s* – current state;
iii. *s'* – next state;
iv. *a* – current action;
v. *a'* – next action;
vi. r – reward;
vii. α – learning rate;
viii. and *γ* – discount factor.

```
Initialize $Q(s,a)$ arbitrarily
Repeat (for each episode):
    Initialize $s$
    Repeat (for each step of episode):
        Choose $a$ from $s$ using policy derived from $Q$ (e.g. ε-greedy)
        Take action $a$, observer $r$, $s'$
        $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$
        $s \leftarrow s'$
    Until s is terminal
```

Figure 1: Q-learning algorithm [15]

As time progresses, *Q(s, a)* values stored in Q table shown in Figure 2 will be updated. Correct action taken by the system will be rewarded or otherwise it will be penalized.

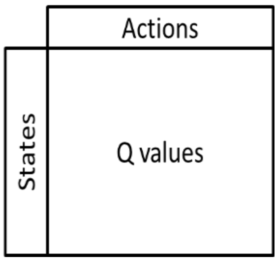


Figure 2: Q-learning's state action table

The progressively updated stored *Q(s, a)* values located in the table will provide information about the action to be taken in what state in the future. For a system which utilizes Q-learning algorithm, prior to the system development, knowledge expert will provide information about states, s and actions, *a*.

Shown in Figure 3, in contrast to the self-learning algorithm, the information about the state, *s* will be generated automatically by AutoWiSARD as learned information when the system autonomously classifies the environment.

B. AutoWisard (Unsupervised WiSARD) Algorithm

As stated earlier, the newly formulated self-learning algorithm was derived from the combinations of Q-learning and AutoWiSARD. AutoWiSARD is an unsupervised learning version of WiSARD algorithm. Previous research performed by others had successfully demonstrated that WiSARD, can positively classifies the states or different types of obstacles or environments when applied in reactive system such as mobile robot applications [9], [12] but requires comprehensive training and retraining of anticipated situations or states.

The presumptions that AutoWiSARD can successfully be used to automatically classify states without supervision was based on:

i. AutoWiSARD is an extended version of WiSARD algorithm, therefore it retains similar traits and capabilities as WiSARD and with the improvements of unsupervised learning;
ii. AutoWiSARD was successful in classifying optical recognition of handwritten digits [1];

Figure 4 illustrates the WiSARD WNN structure. In summary, WiSARD inputs will be mapped as 1 or 0 value onto row by column table which represents a pattern. This pattern will then be converted into RAM type discriminator during training. The input will be compared with every RAM discriminators and their *R* value will be calculated during recognition process. The input will belong to a particular class represented by the RAM discriminator if it has the highest *R* value.

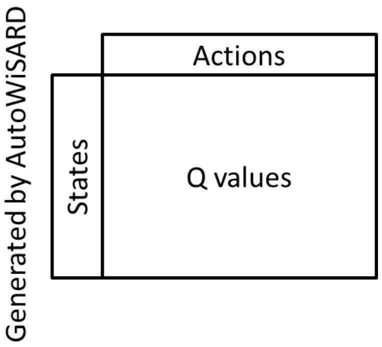


Figure 3: Self-learning algorithm

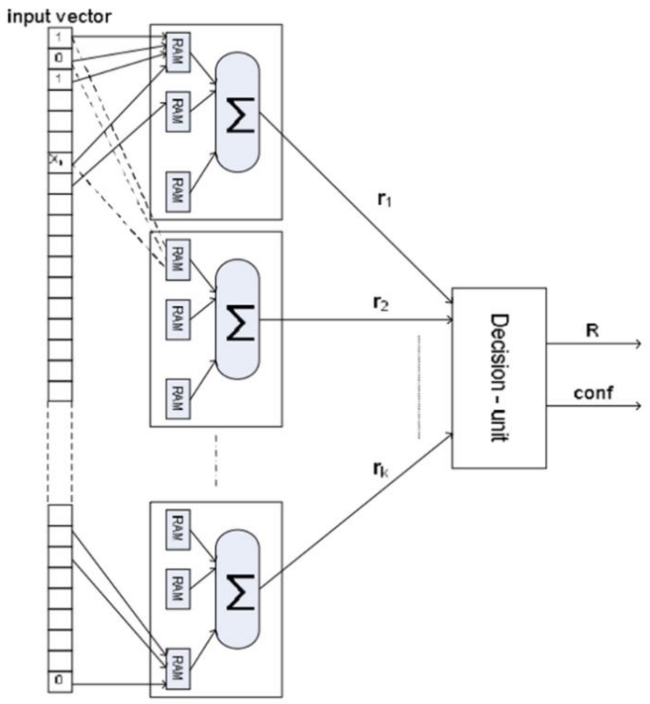


Figure 4: WiSARD WNN structure [10]

By manipulating the outcome *R*, the supervised learning version of WiSARD can be transformed into unsupervised learning AutoWiSARD as demonstrated by Wickert [1]. The automated classifier was realized when the *R* value was be mapped to a learning window shown in Figure 5 and then applied with the learning policy shown in Figure 6.

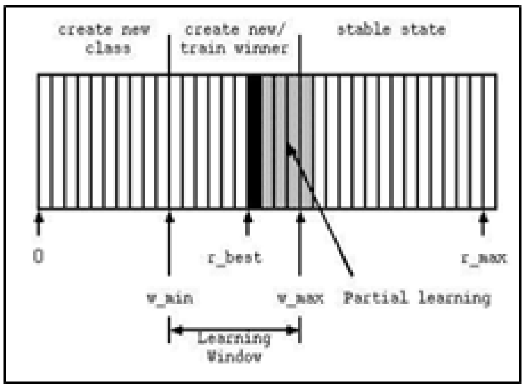
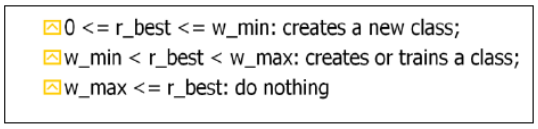


Figure 5: AutoWiSARD *R* learning window [1]



0 <= r_best <= w_min: creates a new class;
w_min < r_best < w_max: creates or trains a class;
w_max <= r_best: do nothing

Figure 6: AutoWiSARD learning policy [1]

C. Self-learning Algorithm Formulation

The formulated hybrid self-learning algorithm which derived from the combination of AutoWiSARD and Q-Learning does not require prior knowledge. During implementation, the algorithm will gradually differentiate the states and eventually learn to react. Both algorithm were chosen due to the speed, size and efficiency which is suitable when implemented in resource constrained embedded system. Figure 7 describes the self-learning algorithm loop.

- ***System***: *Read the sensors.*
- ***AutoWiSARD***: *Classifies and determines in what state the robot is.*
- ***Q-learning***: *Select the action according to the state and values of the corresponding action.*
- ***Q-Learning***: *execute the selected action.*
- ***System***: *Read the sensors*
  - ***Q-Learning***: *Computes the reward, calculates and updates the state-action pair(s) Q values.*

Figure 7: Self-learning algorithm loop

IV. Mobile Robot Simulation

A. Simulation Environment and Key Parameters

The self-learning algorithm was simulated in an open-source Simple 2D Robot Simulator in Python+Pygame simulator developed by M. Agapie [20]. Figure 8 depicts the simulated mobile robot.

The mobile robot was equipped with thirteen sonar sensors that will provide information about the distance of an object. Data collected from the sensor will then be mapped onto a 6x5 array for AutoWiSARD input as shown in Table 1.

Referring to Table 1, S_x represents thirteen sensor inputs attached to the robot shown in Figure 8. The t, denotes distance threshold. When $S_x \leq t$, then column value will be set to 1 or otherwise 0. Figure 9 shows two different pattern sensed by the robot while wandering in the simulation environment. From visual inspection it is clear that both pattern can be classified into different classes because they look totally different.

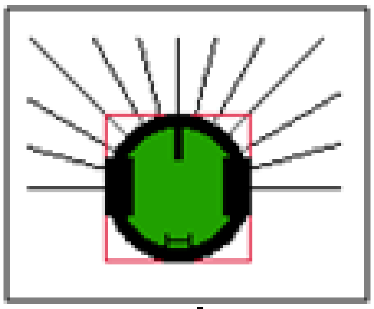


Figure 8: Mobile robot with 13 sonar sensors

Table 1
Sensor pattern mapping rules represented by 6x5 array for AutoWiSARD input

$S_5 \leq t$	$S_0 > t$	$Ss_6 \leq t$	$S_7 \leq t$	$S_{12} > t$
$S_4 \leq t$	$S_1 > t$	0	$S_8 \leq t$	$S_{11} > t$
$S_3 \leq t$	$S_2 > t$	0	$S_9 \leq t$	$S_{10} > t$
$S_2 \leq t$	$S_3 > t$	0	$S_{10} \leq t$	$S_9 > t$
$S_1 \leq t$	$S_4 > t$	0	$S_{11} \leq t$	$S_8 > t$
$S_0 \leq t$	$S_5 > t$	$S_6 > t$	$S_{12} \leq t$	$S_7 > t$

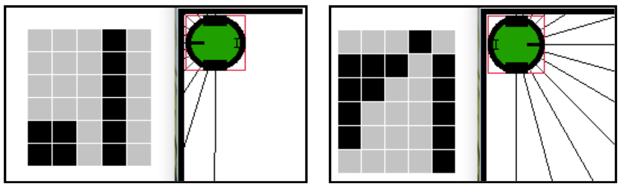


Figure 9: Sensor patterns

Throughout the simulation, the key parameters such as reward value and learning parameters values were set as according to Table 2 and Table 3 respectively.

Table 2
Q-learning reward function

Actions	Reward
Move forward and hit obstacle	-0.7
Move forward and did not hit obstacle	1
Spin in same direction	-0.1
Spin in opposite direction	-0.3

Table 3
Q-learning parameters

Parameters	Value
α – learning rate	0.1
γ – discount factor	0.9
ε-greedy exploration algorithm (ε value)	0.1

B. Result and Discussion

The robot was set to wander in two different simulation environment and results were recorded. Prior to this simulation, the robot did not have any information about the obstacles and methods to evade them. First, the robot was set to wander in the environment shown in Figure 10.

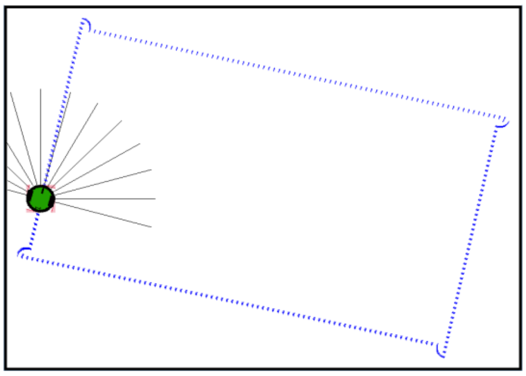


Figure 10: Traced path in environment 1

Shown in Figure 11, during the first 500 simulation cycles the robot discovers 17 different types of obstacles and tries to determine best sequences of action to evade them. The final path for the robot is shown in Figure 10.

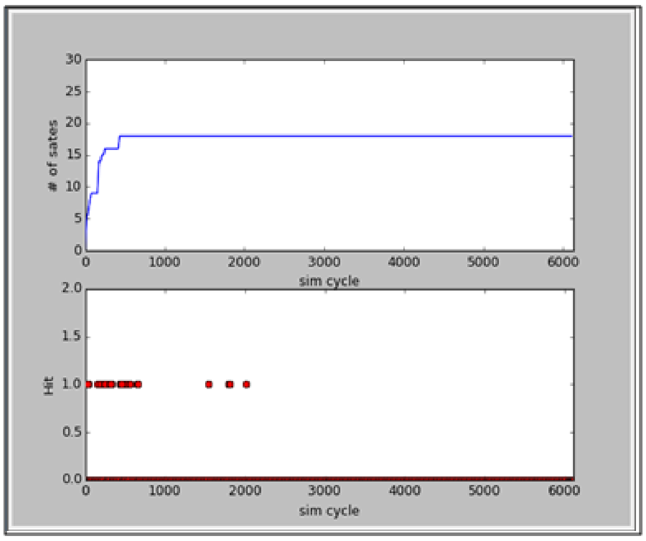


Figure 11: Observed result for environment 1

The final path taken when introduced to the second environment is shown in Figure 12. In this environment, the robot found three new obstacles as shown in Figure 13.

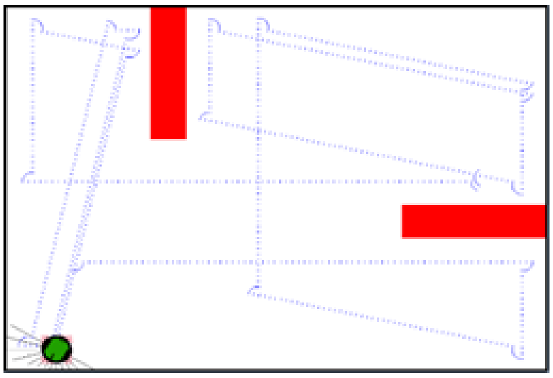


Figure 12: Traced path in environment 2

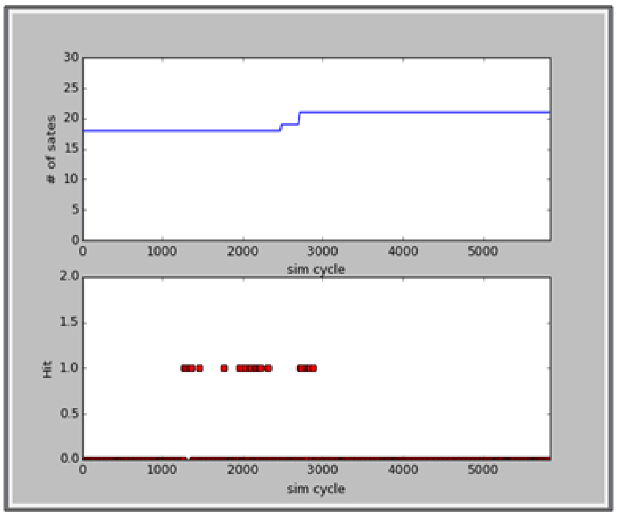


Figure 13: Observed result for environment 2

V. Physical Implementation

A. *Mobile Robot Configurations and Key Parameters*

The robot shown in Figure 15 was constructed using Lego Mindstorms EV3 Kit and having the followings configurations:

i. EV3 IntelliBrick, programmed using Java on LeJOS EV3 0.9.0-beta JVM platform;
ii. Touch sensors on the input ports 1 and 2, 2 servos for the wheels on output ports A and D; and 1 ultrasonic sensor for avoiding obstacles/walls connected to input port 4.

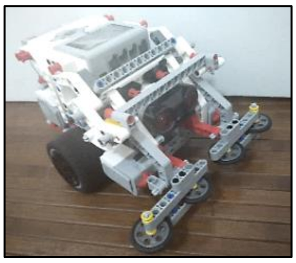


Figure 14: Lego EV3 mobile robot

For AutoWiSARD inputs shown in Table 4: a) values *X* in columns will be set to 1 when the ultrasonic sensor sense an obstacles is less than 30 cm or; b) otherwise the values of *Y* will be set to 1; c) when the front-left touch sensor touches any object the *L* values will be set to 1; d) when the front-right touch sensor touches any object the *R* values will be set to 1.

Table 4
Sensor Mapping Pattern for AutoWiSARD Input

L	*L*	*L*	*0*	*0*
L	*0*	*0*	*X*	*X*
L	*X*	*0*	*X*	*R*
X	*X*	*0*	*0*	*R*
0	*Y*	*R*	*R*	*R*
Y	*Y*	*Y*	*Y*	*Y*

Table 5 and 6 shows the Q-learning parameters implemented in the mobile robot.

Table 5
Q-learning reward function

Actions	Reward
Hit obstacle	-3.0
Move forward	3.0
Spin in same direction	0.5
Spin in opposite direction	0.05

Table 6
Q-learning Parameters

Parameters	Value
α – learning rate	0.1
γ – discount factor	0.9
ε-greedy exploration algorithm (ε value)	0.2

VI. Result and Discussion

Figure 15 and 16 indicates that during the experiment the robot will start to classify the environment and spins few times at the start-up position then determine the best sequences of action in order to avoid both the static and randomly placed obstacles.

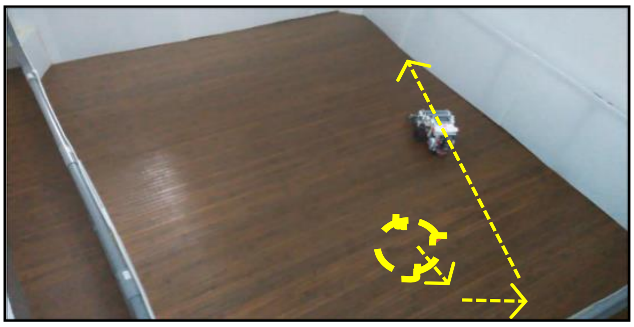


Figure 15: Start-up position

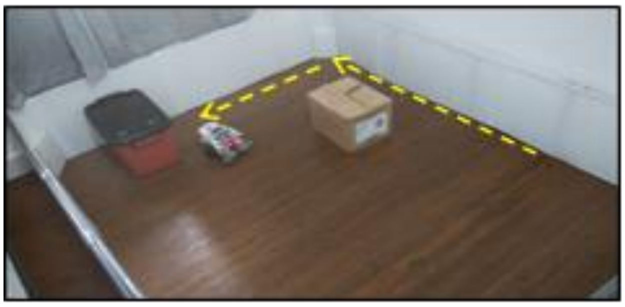


Figure 16: Random obstacles 1

Shown Figure 17, the physical robot classifies 6 different types of obstacles. Due to limited ultrasonic sensor usage compared to the simulated robot, the physical robot tries to find best possible way to avoid from hitting the obstacles. Result shown in Figure 18 indicates that the robot hits the obstacles 16 times during the 250 run cycles, which is 6.4% hit rate.

VII. Conclusion and Future Works

In this paper we present a novel hybrid self-learning algorithm derived from AutoWiSARD and Q-learning algorithm implemented in an autonomous mobile robot navigation. Both the simulation and physical implementation result verifies that the algorithm enables the robot to self-learn without having prior knowledge of its environment by differentiating various types of obstacles, keeps learning and correcting itself to avoid them with more efficiency while wandering in dynamically changing environment.
In the future the self-learning algorithm will be implemented in other types of application in order to demonstrate its adaptability and capability of solving other problems. The currently formulated self-learning algorithm will be extended to include an autonomous discovery of reward thus making the more truly independent from human intervention. Other approach of formulating self-leaning algorithm will be investigated and implemented utilizing other combination of unsupervised and reinforcement learning algorithm; and implemented in other system which requires autonomous learning.

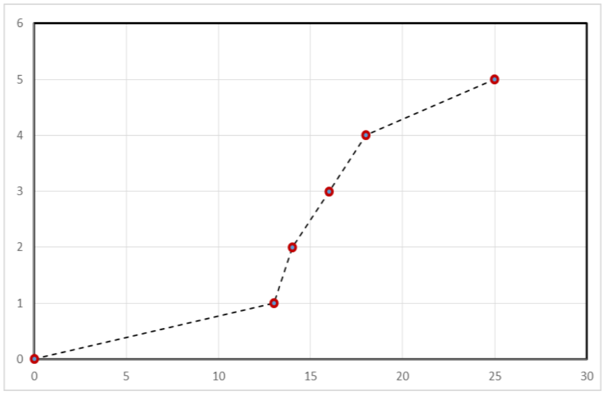


Figure 17: Number of states classified (States vs. Run cycles)

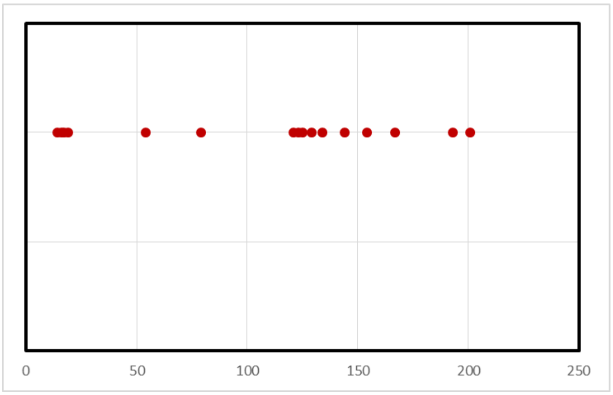


Figure 18: Robot hit walls (Hit vs Run cycles)

References

[1] Iuri Wickert and Felipe M. G. França. 2001. AUTOWISARD: Unsupervised Modes for the WISARD. 6th International Work-Conference on Artificial and Natural Neural Networks, IWANN 2001. Granada, Spain. 13-15 June 2001. 435-441.
[2] R. W. Yousif and M.A.H., Mansor. 2009. Design and Simulation of a New Self-Learning Expert System for Mobile Robot. International Journal of Computer, Electrical, Automation, Control and Information Engineering. 3(2).
[3] U., Kamath U. 2008. Self-learning expert systems using rule classifier in detection engines. International Conference on Artificial Intelligence and Pattern Recognition, AIPR 2008, Orlando, Florida, USA. 7-10 July 2008. 224–227.
[4] Y., Liu, Wang and M. Guo. 2005. The research and application of the self-learning expert system based on BP network. Fourth International Conference on Machine Learning and Cybernetics. 2005. China. 18-21 August 2005. 18–21.
[5] C., Kirby, A., Sadlier, C.,Wood and M.,Vinther. 2013. Filling the Experience gap in the Drilling Optimization Continuous Improvement Cycle Through a Self-Learning Expert System. SPE Middle East Oil

and Gas Show and Conference. 2013. Manama, Bahrain. 10-13 March 2013.

[6] E., Piga and A., Geschiere. 2009 Self learning expert system (SLES) for power transformers. 20th International Conference and Exhibition on Electricity Distribution, CIRED 2009, Prague, Czech Republic. 2009. 8-11 June 2009. 1-3.

[7] L., Chen and J., Li. 2012. Development and application of blast furnace expert system with self-learning function based on pattern recognition. Dongnan Daxue Xuebao (Ziran Kexue Ban)/Journal Southeast Univ. (Natural Sci. Ed.). 42(1):117–121.

[8] D., Harel and A., Pnueli . 1985, Logics and Models of Concurrent Systems. New York: Springer-Verlag New York.

[9] B., Mcelroy, M., Gillham, G., Howells, S., Spurgeon, S., Kelly, J., Batchelor and M.,Pepper. 2012. Highly efficient Localisation utilising Weightless neural systems. European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. 2012. Bruges, Belgium. 25-27 April 2012. 25–27.

[10] S., Nurmaini . 2009. Modular Weightless Neural Network Architecture for Intelligent Navigation. International Journal of Soft Computing Application. 1(1): 1–18.

[11] M., Gillham, B., McElroy, G., Howells, S., Kelly, S., Spurgeon and M., Pepper. 2012. Weightless Neural System Employing Simple Sensor Data for Efficient Real-Time Round-Corner, Junction and Doorway Detection for Autonomous System Path Planning in Smart Robotic Assisted Healthcare Wheelchairs. 2012 Third International Conference Emerging Security Technology. Lisbon, Portugal. 5-7 Sept. 2012. 161–164.

[12] S. Nurmaini and B. Tutuko. 2011. A New Classification Technique in Mobile Robot Navigation. TELKOMNIKA. 9(3): 453–464.

[13] Building a Light-seeking Robot with Q-learning: InformIT retrieved October, 18, 2015 from http://www.informit.com/articles/article.aspx?p=26423&seqNum=3.

[14] Z., Wang, Z., Shi , Y., Li and J., Tu . 2013. The Optimization of Path Planning for Multi-robot System using Boltzmann Policy based Q-Learning Algorithm. 2013 IEEE International Conference on Robotics and Biomimetics (ROBIO). Shenzhen, China. 12-14 December 2013. 1199–1204.

[15] H., Wicaksono. 2011. Q learning behavior on autonomous navigation of physical robot. 8th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI). 2011. Incheon, Korea. 23-26 November 2011. 50–54.

[16] H.,G.,A.,M., Víctor Ricardo Cruz-Álvarez and Enrique Hidalgo-Peña. 2012. A line follower robot implementation using Lego's Mindstorms Kit and Q-Learning. Acta University. 22. 113–118.

[17] S., Dini and M., Serrano. 2012. Combining Q-Learning with Artificial Neural Networks in an Adaptive Light Seeking Robot.

[18] I., Aleksander and T.J., Stonham. 1979. Guide to pattern recognition using random access memories. IEE Journal Computing and Digital Technology. 2(1):29.

[19] S., Sahin, M.R., Tolun and R., Hassanpour. 2012. Hybrid expert systems: A survey of current approaches and applications. Journal of Expert Systems and Applications. 39(4):4609–4617.

[20] Simple Python robot simulator 2D download | SourceForge.net retrieved September, 14, 2015 from http://sourceforge.net/projects/pyrobosim2d/.