Adaptive Behavior of Evolutionary Neurocontroller for Obstacle Avoidance Mobile Robot

Aloysius Aldo Gani, Sofyan Tan and Meiliayana

Abstract—**This research aims to analyze the adaptive behavior of an artificial neural network-based controller (neurocontroller) for an obstacle avoidance wheeled mobile robot. The neurocontroller of interest is the best of an ecosystem of neurocontrollers that evolved according to the genetic algorithm. The chromosome of each individual neurocontroller is simply the binary weights of the connections, and the fitness function is a simple rule for obstacle avoidance behavior. To limit the hardware requirements, the neurocontroller's evolution is simulated in a computer, and the best neurocontroller behavior is analyzed in the simulation and in an actual mobile robot. Albeit a simple chromosome, the resulting best neurocontroller showed an obstacle avoidance behavior in both simulation and in the actual mobile robot, regardless of the positions of obstacles in the environment. The analysis of the chromosome after evolution also found that not all of the sensors are actually needed for the obstacle avoidance behavior.**

Index Terms—**Evolution, genetic algorithm, neural network, neurocontroller, reinforcement learning**

I. Introduction

MANY computing systems are being modeled in the imitation of biological systems in nature for its adaptive behavior. In the context of robots, an adaptive behavior enables a robot to operate with minimum or no supervision by human in a changing environment. This behavior is important when the robot must operate in an environment that is unsafe for humans, an unclear or unknown environment, or one having limited communication link. Examples are the outer space and the deep sea.

Two popular algorithms modeling the biological epigenesis and phylogenesis are the artificial neural network (ANN) [1] and the evolutionary algorithm (EA) [2] respectively. The former models the connections of neuron in the brain, whereas the latter models the Darwin's evolutionary theory. Generally the ANNs are trained in the supervised learning scheme using the back propagation algorithm. The supervised learning scheme requires a comprehensive training set to train the ANN. Such training set may be difficult or even impossible to be completely obtained. The more complex the behavior and environment, the more difficult it is to obtain the training set. Moreover, the back propagation algorithm is a gradient-based optimization algorithm, which is prone to be trapped in local optimum of the search space.

The evolutionary ANN in [3] and [4] offers an alternative for an intelligent neurocontroller (an ANN for control purpose). The algorithm involves the evolution of an ecosystem of neurocontrollers using the genetic algorithm. The chromosome of each neurocontroller is the string of all connection weights. Instead of training a single neurocontroller, many neurocontrollers compete for survival for several generations based on their fitness value. Instead of specifying all the reactions for all situations, as in the supervised learning, the fitness function measures the performance of each neurocontroller in accordance to the desired behavior. At each generation the neurocontrollers are subjected to elimination of the lowest fitness individuals, crossover between individuals, and random mutation. The evolution continues for generations until a neurocontroller with fitness value exceed a threshold value is found.

This approach of learning is categorized as the reinforcement learning where the neurocontroller is just given a set of general rules to achieve. The actual strategy to achieve that desired behavior is learned by the neurocontroller through interaction with the environment. Furthermore this learning algorithm is less prone of being trapped in the local optimum.
This research is a variation of the evolutionary neurocontroller in [5] in which the proposed neurocontroller consisted of simply binary weights, hence the chromosome being binary. The proposed neurocontrollers are evolved in simulation, and the best neurocontroller is implemented in an actual mobile robot. Analysis of the adaptive behavior of the neurocontroller is carried out in the simulation and an actual mobile robot.

A. Related Works

The idea of neural network learning from its environment was introduced in [6]. It simulated an ecosystem of neurocontrollers that decide their next actions while learning to predict the future sensor reading during their lifetime. Many of the weight connections of each neurocontroller are trained

Manuscript received July 27, 2012. This work was supported in part by Universitas Pelita Harapan.

A. A. Gani was with the Computer Technology Department, Universitas Pelita Harapan, Tangerang, Indonesia. He is now with the School of Computer Engineering, Nanyang Technological University, Singapore (e-mail: aldo_gani@yahoo.com).

S. Tan was with the Computer Technology Department, Universitas Pelita Harapan, Tangerang, Indonesia. He is now with the Computer Engineering Department, Binus University, Jakarta, Indonesia. (e-mail: sofyan@binus.edu).

Meiliayana was with the Computer Technology Department, Universitas Pelita Harapan, Tangerang, Indonesia. She is now with PT. Sumatech, Tangerang, Indonesia (e-mail: meiliayana@yahoo.com).

using back-propagation to predict the next sensor reading during its lifetime, whereas some output weight connections for determining the next action are fixed. At the end of their lifetime the best neurocontrollers inherit their learned weight connections to their offspring which then undergo mutation. The paper emphasized that by learning to predict the next sensor reading, an ecosystem of neurocontrollers has a better chance of finding neurocontroller with the desired behavior. The paper only adopted the selection and mutation genetic operators, and it did not employ crossover operator. The evolutionary learning is further surveyed by [3] and [4] to discuss various combinations of neural networks and evolutionary algorithms, and various search operators. It shows that an evolutionary algorithm can be used to evolve not only the connection weights, but also the architecture and the learning rule of the neural network. Some applications of the evolutionary neurocontroller are demonstrated in [5] and [7-9] using a fixed learning rule or fixed architecture. The chromosome encodes the connection weights and the neuron thresholds of the neurocontroller as floating point numbers.

B. Methodology

Evolution of a population of neurocontrollers in hardware can be expensive and takes a very long time. Therefore it is more convenient to simulate the evolution in computer and then implement the best neurocontroller in the actual robot for evaluation. The simulation of the evolution requires modeling of the mobile robot, the environment, and the neurocontroller.

In this research, an actual mobile robot, the arena, and their models are constructed. The models are used to simulate the evolution in computer to find the best-performing neurocontroller after several generations. The best neurocontroller is then implemented in the microcontroller of the actual mobile robot to evaluate its behavior in the actual arena. The next subsections will describe the mobile robot, the arena, the neurocontroller, the evolution simulation, and the hardware implementation of the neurocontroller.

II. THE MOBILE ROBOT AND THE ARENA

The mobile robot to be controlled is a wheeled mobile robot having two wheels and a caster to balance the robot. Each wheel has a diameter of 6.5 cm and it is driven by a DC motor that can rotate in forward and backward directions. The robot is 15 cm in length and 15 cm wide, and the two wheels are aligned to the back of the robot so that the turning radius of the robot is 13.5 cm from the center of the wheel axis. Details of the components arrangements and the photograph of the actual mobile robot are shown in Fig. 1. The placement of components in the mobile robot are arranged to impose the same load to the two wheels.

The robot is equipped with eight ultrasonic sensors distributed around the robot to ensure than the behavior of the neurocontroller will not be restricted by the lack of sensors in some directions. Four sensors are aligned to the forward, backward, left, and right directions of the mobile robot, while the other four covers the four corners of the mobile robot in order to eliminate blind spot, as shown in Fig. 1c. Each sensor can measure the distance to an object at 2 cm to 3 m.

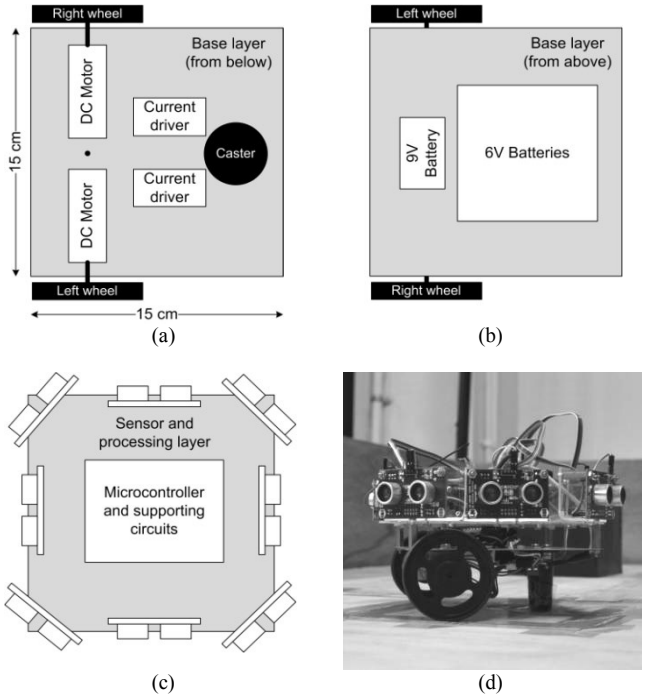


Fig. 1. Arrangement of components in the mobile robot (a) under the base layer, (b) above the base layer, (c) above the sensor and processing layer, and (d) the photograph of the actual mobile robot.

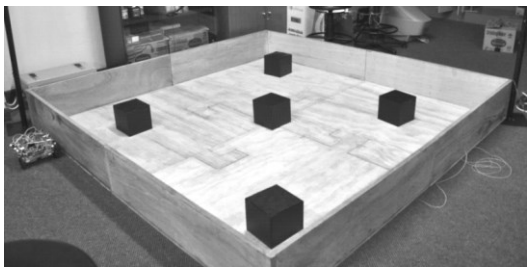


Fig 2. Photograph of the actual arena with five obstacles scattered in the arena.

An ATMega8 microcontroller of the Atmel AVR family is used in the mobile robot to collect information from all 8 ultrasonic sensors and to control the two DC motors. All the sensors are connected to the microcontroller through an I2C bus. The DC motors are controlled through the appropriate current drivers using pulse width modulation signal from the microcontroller. The best neurocontroller found from the simulation of evolution is implemented in the microcontroller as feed forward network without learning capability.

The arena is simply a 2×2 m wooden floor and bordered with 25 cm high wooden walls at the four edges. The obstacles are paper cubes, each sized 15 cm. These cubes can be moved to any location inside the arena during hardware evaluation of the best neurocontroller, as shown in Fig. 2.

III. THE NEUROCONTROLLER

Each neurocontroller is a single-layer perceptron consisted of eight input nodes and two output nodes. Each input node receives distance information from an ultrasonic sensor that measure the distance to the closest object. The two output

nodes control the speed of the two DC motors in a differential drive configuration. Configuration of the neurocontroller is depicted in Fig. 3.

The distance information from each sensor is converted to a single bit binary value by comparing the distance to a threshold of 16 cm. The threshold is chosen based on the response time of the sensor and the speed of the mobile robot. Therefore the input value S_i from sensor at node i of the input layer is expressed as

$$S_i = \begin{Bmatrix} 0 & dist_i > 16 \\ 1 & dist_i \le 16 \end{Bmatrix} \quad 1 \le i \le 8 \tag{1}$$

The weights W_{ji} are single-bit binary values that model the connectivity of synapses from input node i to output node j. The values of the weights are expressed as

$$W_{ji} = \{0,1\} \quad 1 \le j \le 2,\ 1 \le i \le 8 \tag{2}$$

The output of the neural network O_j is also a single-bit binary value that follow the following activation function

$$O_j = \begin{Bmatrix} 0 & net_j = 0 \\ 1 & net_j > 0 \end{Bmatrix} \quad 1 \le j \le 2, \text{ where } net_j = \sum_{i=1}^{8} W_{ji} S_i \tag{3}$$

The activation function is a threshold function that instructs the wheel to rotate forward when the internal activity (net_j) of the neuron is zero and rotate backward when it is larger than zero. The net_j cannot have a negative value since the inputs and weights are equal or larger than zero.

Combination the output value O_j of the two nodes in the output layer produces four possible movements of the mobile robot, listed in Table 1.

IV. The Evolution of the Neurocontroller

Simulation of the evolution is performed in computer. The mobile robot kinematic is modeled according to the dimensions of the actual mobile robot, whereas the dynamic is assumed to be ideal, such as zero mass and zero friction.

Each sensor in the simulation is assumed to have zero angle of view, hence it only measures the distance of obstacle straight in from of the sensor. The actual sensors however have field of view of a few tens of degrees. The arena is modeled as a two dimension plane with line walls and square obstacles. The mobile robot velocity vectors can only be parallel to the plane. The arena's walls and the obstacles are modeled as rigid static objects during evolution.

A. The Chromosomes

For each generation there are one hundred neurocontrollers which correspond to one hundred chromosomes. Each chromosome is consisted of 16 genes, which are all the weights in a neurocontroller, and as explained, these genes have binary value. The arrangement of weights in the chromosome is shown in Fig. 4a, where weights affecting the

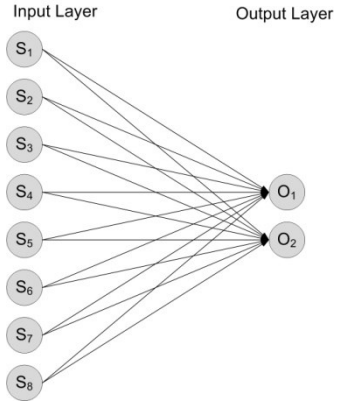


Fig 3. Each neurocontroller is consisted of eight input nodes connected to eight ultrasonic sensors and two output nodes connected to two motor drivers.

TABLE I. Possible Movements of the Mobile Robot

Left Wheel (O_2)	Right Wheel (O_1)	Movement
0	0	Forward
0	1	Rotate right
1	0	Rotate left
1	1	Backward

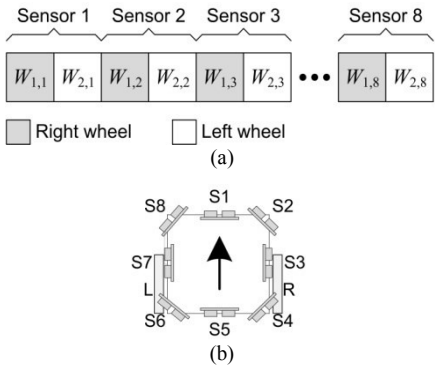


Fig 4. (a) Arrangement of weights in chromosome and (b) the corresponding sensors in mobile robot.

left and right wheels are arranged alternately. The corresponding placement of sensors is shown in Fig. 4b, where the arrow shows the direction of forward movement of the mobile robot.

B. The Fitness Function

Fitness function of the neurocontroller is designed to appreciate neurocontroller that stay away from obstacles (including walls), while consistently move forward or consistently move backward. Fitness of each individual neurocontroller at each generation is calculated according to

$$fitness = \frac{0.7\sum_{k=1}^{1000} C1_k + 0.3\left|\sum_{k=1}^{1000} C2_k\right|}{1000} \tag{4}$$

Eq. 4 shows that at each generation the neurocontroller must control the simulated mobile robot for 1000 steps of movement. There are two sub-behaviors in the fitness function

which are the obstacle avoidance sub-behavior and the straight movement sub-behavior, assessed as the $C1$ and $C2$ respectively. The two assessments are averaged so that the maximum fitness value is one, and weighted so that the $C1$ assessment is much more dominant than the $C2$ assessment. The main behavior of the mobile robot is obstacle avoidance, hence the larger weight of 0.7 for $C1$. The smaller weight for $C2$ makes it a lower priority assessment, such that in difficult conditions the robot may keep rotating or moving forward and backward alternately as long as it does not bump into obstacle.

The obstacle avoidance sub-behavior assessment is calculated according to

$$C1_k = \begin{Bmatrix} 1 & D_k = 1 \\ 0 & D_k < 1 \end{Bmatrix} \quad 1 \le k \le 1000 \tag{5}$$

where $D_k = \max(S_{1k}, \cdots, S_{8k})$. Whenever any sensor reports a distance of 16 cm or less from an object at step k, then $C1_k$ has a value of zero, otherwise $C1_k$ is one.

The straight movement sub-behavior assessment is calculated according to

$$C2_k = \frac{(2O_{1k} - 1) + (2O_{2k} - 1)}{2} \tag{6}$$

Whenever the two outputs O_{jk} are all ones or all zeros at a particular step k, then the magnitude of $C2$ is at its maximum, otherwise $C2$ is zero. $C2_k$ is +1 when the robot is moving forward, and −1 when backward. Otherwise, when the robot is rotating left or right then $C2_k$ is zero. Preservation of the sign in Eq. 6 ensures that neurocontroller that has a consistent forward or backward movement during a generation has a better fitness value when accumulated in Eq. 4.

C. The Genetic Algorithm

The simulated evolution lasts for 100 generations, starting with 100 neurocontrollers with random chromosomes. The square obstacles are sparsely placed at the beginning of the evolution and their positions are static throughout the evolution. The starting position of the mobile robot in each generation is alternating between four scattered starting positions to ensure adaptive behavior of the neurocontroller. At each generation neurocontrollers are running in the simulated arena in sequence, and at the end of the generation, after all neurocontroller has had their turn, the fitness values of all neurocontrollers are calculated. From one generation to the next generation there are elimination based on the neurocontroller's fitness values, crossover, and random mutation of the neurocontrollers.

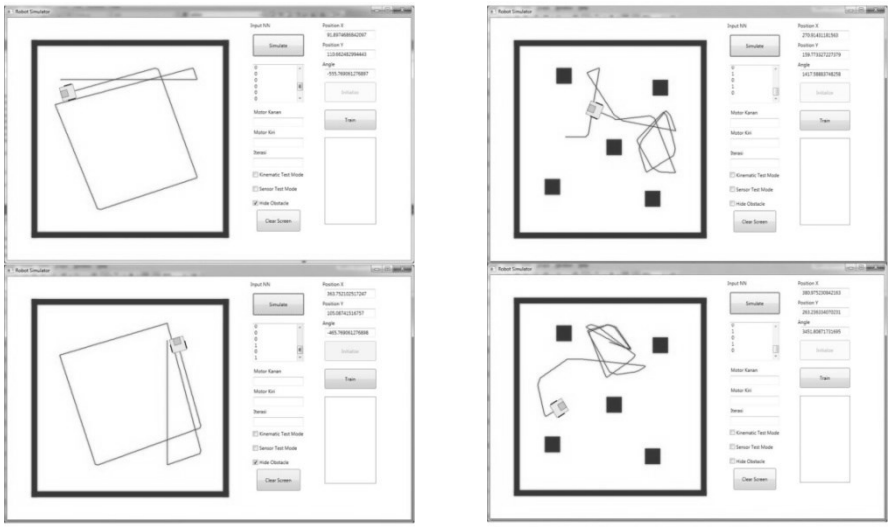
The elimination method used is the truncation method, where some of the neurocontrollers that have lower fitness values are eliminated and replaced with the same number of neurocontrollers with random chromosomes. The elimination rate in the simulation is 20%, which means that 20% lowest fitness neurocontrollers will be replaced with neurocontrollers with random chromosomes.

The resulting population of neurocontrollers will then go through the crossover process, using the one point crossover method. In the process, 24 randomly chosen pairs of parent neurocontrollers are exchanging part of their genomes/weights to form 24 new pairs of offspring neurocontrollers, which replace their parents.

Finally a random mutation is applied to the population of neurocontrollers. The mutation rate is 1%, where 16 out of 1600 genomes/weights in a population are flipped. The resulting population of neurocontrollers proceeds to the next generation, where their fitness will be calculated.

V. Results and Discussion

Fig. 5 shows the simulator GUI along with the simulated arena at the left side of the screen. The GUI supports the initialization, evolution, and testing of the neurocontrollers.



(a) (b)

Fig. 5 Trajectories of the best neurocontroller after evolution when (a) bounded by walls only and (b) by additional five square obstacles inside the arena.

In the simulated arena the last position of the mobile robot is pointed by the drawing of the mobile robot. The line behind the robot illustrates the past trajectory of the mobile robot, where the end of the line points to the starting position of the mobile robot. Square obstacles can be placed in any position inside the simulated arena.

In the first simulation evaluation, Fig. 5a, there is no other obstacle but the four walls at each side of the arena. Regardless of its initial position, the mobile robot developed the obstacle avoidance behavior. It can be seen from both evaluations that the mobile robot tends to move straight forward in order to increase its roaming distance. In the second simulation evaluation, Fig. 5b, obstacle avoidance behavior is also shown by the mobile robot when there are square obstacles inside the arena. The mobile robot rotates away from obstacles only when it is close to the obstacles. It is consistent with the thresholding function at the input layer and the fitness function. However it managed to develop moving forward behavior, differentiating the role of the front and rear sensors so that it will not rotate when only the rear sensors are obstructed. In both evaluations the mobile robot never move backward, this is because backward movement reduce the fitness obtained by forward movement, as described in Eq. 4 and 6.

To analyze more detail into the cause of the mentioned behaviors, we can look into the chromosome of the best neurocontroller in Fig. 6. It can be seen that the weights for sensor 4, 5, and 6 are all zero. These three sensors are the three rear sensors as shown in Fig. 4b. The evolution has founded that the three sensors are actually not important for the obstacle avoidance behavior, and they can be removed from the actual mobile robot without disrupting the obstacle avoidance behavior. The explanation is that the mobile robot does not have to move backward to avoid obstacle, it can always rotate away from obstacle.

Further inspection into the connections of the other five sensors shows that all connections to the right motor is zero and all connections to the left motor is one. The activation function in Eq. 3 and the programmed movements in Table 1 show that these values of weights will force the mobile robot to rotate left whenever any of the five sensors are obstructed. This behavior completes the previous analysis showing that the robot will always rotate left until no obstacle is obstructing any of the five sensors at the front and sides, and then continue with forward movement to move away from the obstacle.

One best neurocontroller found from the simulation with obstacles is evaluated in the actual mobile robot and arena shown in Fig. 2. The behavior of the best neurocontroller in the actual mobile robot in the arena is in accordance with the behavior in the simulation with some calibrations. The rotation speeds of the two wheels must be calibrated to assure that the robot's movement is reasonably straight during forward or backward movement. The movement speed of the robot must not be too fast so that during a distance reading period the robot does not move too far. In the evaluation the actual straight movements speed is 5 cm per second.

Sensor 1		Sensor 2		Sensor 3		Sensor 4	
0	1	0	1	0	1	0	0
R	L	R	L	R	L	R	L

Sensor 5		Sensor 6		Sensor 7		Sensor 8	
0	0	0	0	0	1	0	1
R	L	R	L	R	L	R	L

Fig 6. Example of the chromosome of the best neurocontroller.

The sequence of distance readings by the ultrasonic sensors must be controlled to minimize interference between the ultrasonic sensors, while minimizing delay between sensor readings. Furthermore the thresholds of the input neurons must be adjusted differently for different sensor directions. Front and rear sensors have larger threshold values compared to the side sensors in order to compensate the larger displacement of straight forward and backward movements.

VI. CONCLUSIONS

An approach of evolvable neurocontroller algorithm with binary weights is proposed and presented. The neurocontroller is a simple single layer perceptron with eight input neurons corresponds to eights distance sensors and two output neurons corresponds to two DC motors. The fitness function of the genetic algorithm is a simple rule to encourage obstacle avoidance sub-behavior and consistent straight movement sub-behavior.

Although the chromosome is simple and only encodes the weight connections, the simulated evolution of the neurocontrollers has produced a neurocontroller that possess the obstacle avoidance behavior. The best neurocontroller simply controls the mobile robot to rotate left whenever the front and sides sensors detect an obstacle. Only after the obstacle is at the back of the mobile robot then it moves straight forward away from the obstacle. The evolution also found that three rear distance sensors are not important for the mobile robot to have the obstacle avoidance behavior.

The best neurocontroller has been implemented to control an actual mobile robot producing equivalent obstacle avoidance behavior with some adjustments to the mobile robot and neurocontroller.

Future works of this research will involve evaluating the mobile robots for more types of environments such as labyrinth. Furthermore, expanding the research into swarm of neurocontrollers can provide more insights into the capability of evolutionary neural networks for swarm behaviors.

REFERENCES

[1] D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*, MIT Press , Cambridge, MA, 1986.

[2] J. H. Holland, *Adaptation in natural and artificial systems*. The University of Michigan Press, Ann Arbor, 1975.

[3] Xin Yao, *Evolving Artificial Neural Networks*, in Proceeding of the IEEE, 87(9), 1423-1447, 1999.

[4] Yong Liu, Xin Yao, *Evolving Neural Network Ensembles by Fitness Sharing*, in Proceeding of the IEEE Congress on Evolutionary Computation, 3289-3293, 2006.

[5] D. Floreano, F. Mondada, *Evolutionary Neurocontrollers for Autonomous Mobile Robots*, Neural Networks - Special issue on neural control and robotics: biology and technology, 11(7-8), 1461-1478, 1998.

[6] D. Parisi, F. Cecconi, S. Nolfi, *Econets: Neural networks that learn in an environment*, Network, 1, 149-168, 1990.

[7] H. T. Huynh, *Hematocrit estimation from compact single hidden layer feedforward neural networks trained by evolutionary algorithm*, in Proceeding of IEEE Congress on Evolutionary Computation, 2962 – 2966, 2008.

[8] Y. Delican, *Evolutionary algorithms based RBF neural networks for Parkinson's disease diagnosis*, Proceeding of 7th International Conference on Electrical and Electronics Engineering, II, 311-315, 2011.

[9] V. Fernandez, Eduardo, *Assessing the positional values of chess pieces by tuning neural networks' weights with an evolutionary algorithm*, in Proceeding of the World Automation Congress, 1-6, 2012.

Aloysius Aldo Gani obtained his bachelor degree in Computer Technology from Universitas Pelita Harapan, Tangerang, Indonesia in 2011, and Master of Science degree in Embedded System from Nanyang Technological University, Singapore in 2012. He is currently working as an embedded software engineer in Gemalto, Singapore. His research interests are embedded systems, artifical neural network and evolutionary computing.

Sofyan Tan (M'11) received his B.S. degree in computer engineering from Universitas Bina Nusantara, Jakarta, Indonesia, in 2002, and the M.Eng. degree in electronic engineering from the University of Tokyo, Tokyo, Japan, in 2008. He was a lecturer at the department of Computer Technology, Universitas Pelita Harapan, Tangerang, Indonesia, from 2009 to 2011. He is presently a lecturer at Universitas Bina Nusantara, Jakarta, Indonesia. His research interests include evolutionary computing, visual servoing, embedded systems design, adaptive control systems, wireless channel prediction, and digital wireless modulation.

Meiliayana received her Bachelor of Electrical Engineering in 1994 and her Master of Electrical Engineering in 2001, both of them are from Universitas Trisakti, Jakarta – Indonesia. She was an entrepreneur (1994 -1997), a lecturer at Electrical Engineering Department, Universitas Trisakti (1997 – 2003), and a lecturer at Computer System Department, Universitas Pelita Harapan, Tangerang – Indonesia (2003 - 2011). Since 2012, she is a HMI programmer for CNC Machine at P.T. Suma Mekatronik Indonesia – Tangerang. Research focus is in automation, robot movement, and pattern recognition.