Cellular Neural Network Trainer and Template Optimization for Advanced Robot Locomotion, Based on Genetic Algorithm

Alireza Fasih [1], Jean Chamberlian Chedjou [2], Kyandoghere Kyamakya [3]
Transportation Informatics Group, University of Klagenfurt-Austria
[1] afasih@edu.uni-klu.ac.at
[2] jean.chedjou@uni-klu.ac.at
[3] kyandoghere.kyamakya@uni-klu.ac.at

***Abstract*- A new learning algorithm for advanced robot locomotion is described in this paper. This method involves both Cellular Neural Networks (CNN) technology and evolutionary algorithms. Learning is formulated as an optimization problem. CNN Templates are derived by genetic algorithms after an optimization process [1]. A template generates a specific wave on CNN that leads to the best motion of a walker robot. Details of the algorithm and several applications and simulation results are shown and commented. It is shown that an irregular and even a disjointed walker robot can move with the highest performance due to this method.**

***Keywords*- Cellular Neural Network, Walker Robot, Locomotion, Genetic Algorithms.**

I. INTRODUCTION

Today, some of the main goals of robotics science, mechatronics and artificial intelligence lie in designing mechanisms close to or mimicking as good as possible some natural structures or animal behavioral models. According to this theory, nature selects stable genes for breed and weak genes will fall in the nature [2]. The good genes that can adapt animal structure to the environment have higher chances for breed and evolution. Animal locomotion is trained and adapted according to the animal's body structure. Flexible and rigid materials are developed/adapted for energy saving. Since, many scientific groups worldwide are intensively working on the modeling and simulation of animal walking motion.

Currently, we know that the walking motion of animals is a stereotype. In a large variety of animals a central neural controller does organize this motion. Central pattern generator (CPG) is a main unit for controlling limbs for walking [3]. This part does contain all the mechanisms needed to generate the rhythmic pattern of movement. With this model, designing a walker, a swimmer or a flyer robot that is close to natural locomotion mechanisms/behaviors is possible. Due to the recent advances in electronics and the ability of cellular neural networks to solve partial differential equations in real time, we are able to simulate a Reaction-Diffusion model by a specific CNN architecture (RD-CNN that is reaction-diffusion cellular neural network).

Some previous works have already been done based on this technique. In this paper we will introduce a general and effective method combining CNN with genetic algorithms (GA) and that can support and drive many types of structures for walker robots. The key point for success in this approach is the natural modeling and the use of computational units close to the biological models; this is the case for both CNN and GA. The central parts of this scheme are a CNN processor and an evolutionary training unit. Cellular neural networks (CNN) are a parallel computing paradigm similar to neural networks, with the difference that communication is allowed between neighboring units only.

In this paper, we show that the driving of a walker robot by an evolutionary training of a CNN is possible and that we are able to drive directly any type of walker robot with this architecture. Widespread natural movement mechanisms of animals like worms, insects, quadrupeds and biped can be modeled with this structure and they can move easily to any direction and in any situation.

Due to neighborhood connections' structure of CNN processors this type of neural network is very close to biological neural structures. Results of simulation show that this structure has a high potential for driving walker robotic structures [4].

One key challenge is that of finding the best signal for driving walker robot joints that leads to minimum energy consumption and best locomotion performance. The main goal is therefore of finding a suitable CNN template that generates an efficient wave for driving walker robot joints. CNN has a high potential to generate appropriate waves.

By changing the connection formation between CNN columns and robot actuator, converging time for getting the best result might be changed. After a learning phase the CNN output waves will be reshaped according to the robot structure for an efficient locomotion.

II. CELLULAR NEURAL NETWORK MODEL

Cellular Neural Networks have been introduced by Leon O. Chua and Yang from university of California at Berkeley in 1988. This type of neural network is a reduced version of Hopfield Neural Network. In this technology each cell is connected only to neighboring cells. Due to local connections between a cell and the neighbors, the implementation of this type of neural network on chip is feasible. The mathematical model for each cell is a first-order equation that is shown in equation-1.

$$\dot{x} = -x + \sum T_B * u + \sum T_A * y + I \qquad (1)$$

In equation (1) 'T_A' denotes the 3×3 feedback template and 'T_B' denotes the 3×3 control template. 'I' is a bias value whereby 'y' is a linear sigmoid function in output of each cell. 'u' denotes the input value and 'x' is the state of CNN. The size of the 'u' matrix in this paper depends upon the number of joints in the walker robot.

III. USING GENETIC ALGORITHM IN CNN TEMPLATE OPTIMIZATION

A genetic algorithm is a search technique used in computing to find either exact or approximate solutions to an optimization or a search problem. Genetic algorithms (GA) are categorized as a global search heuristics and also it is a type of evolutionary algorithm that uses techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover. In this research we use GA to find the best templates for an optimum robot locomotion behavior.

According to Fig.1, the CNN outputs should be connected to the robot hinges. These connections lead to driving robot actuators in hinges. Wave rhythms that are generated from the CNN processor outputs can drive walker robot on the specific path and direction. After a learning step, output waves are adapted to an optimum robot locomotion. In a learning based on GA, each chromosome is a solution for robot locomotion. But, in the first generation of chromosomes, locomotion is neither optimum nor suitable and the robot behavioral locomotion is random without any purpose. In fact, each chromosome is a CNN template that is reshaped in a one dimensional array.

According to Fig.2, each chromosome does contain a feedback template, a control template and a bias value. One particular important part of this algorithm is the design of the fitness function. Fitness function or cost function defines/fixes indirectly, the robot behavior [5].

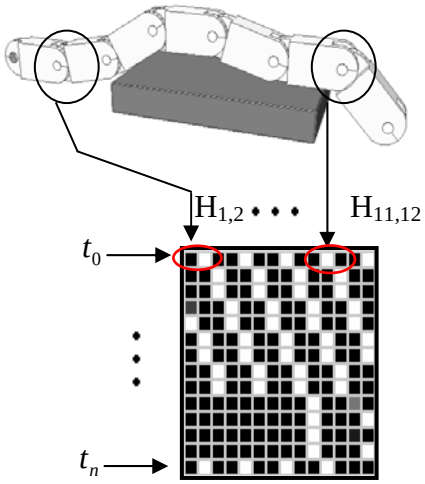

Figure 1. Robot hinges connection to CNN

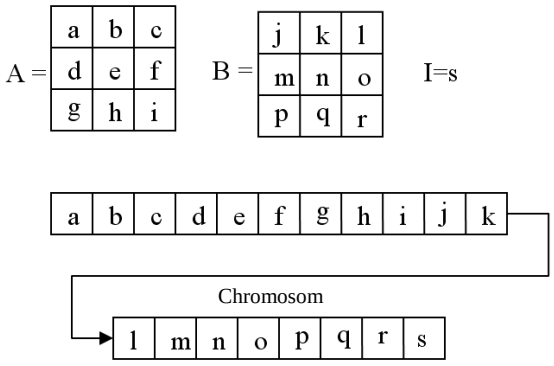

Figure 2. Template Encoding in an Array List

A fitness function is a particular type of objective function that quantifies the optimality of a solution. Input data for the fitness function is based on measurements of robot parts orientation, location and displacement. In fitness function we don't define any behavioral locomotion exactly. On the other hand, we define a function that satisfies the target or destination. For example, in some time, we need the robot to escape from a position without any specific direction. For this purpose, we must define a simple function that return 1/distance of the robot center to the initial position. In this example, we didn't define any detail for locomotion behavioral. After generating new chromosomes, we obtain the corresponding fitness value by applying the fitness function. The main point on applying fitness function is that this function is not a real time procedure and that the fitness calculation result will be only ready after a certain period of time beyond the time the wave effect will act on robot hinges actuators. In fact,

one cycle of time is not sufficient for robot movement. Each generated wave will be applied to robot actuators. However, by measuring some robot parameters like robot center position, robot angle and so on, solving the fitness function is easier and faster.

In the initial state of the training phase the algorithm needs to select some randomly generated chromosomes. There is no rule on how many chromosomes should be generated in the initial population. But, depending on the problem this number may be different [6]. Some authors define 20 generations for the initial state. After each generation, a fitness function does evaluate the cost of chromosomes in the simulator. Each evaluation takes 3 seconds and we do need 60 seconds for evaluating the cost of 20 chromosomes approximately. After this step, chromosomes and fitness values will be sorted by order of minimum fitness values in a link list.

Now, it is time for selection and breed. Experiments show that 50% of the best chromosomes are fit for the crossover step and this range has good chance of generating the better chromosomes. In each step, one randomly selects 2 chromosomes in this range for the crossover process. Many evaluations show that using two-point technique for the crossover is better for this purpose.
We used a two point crossover method for this algorithm; in this process program we select two points randomly on the array of chromosomes. Then the contents of the chromosomes between these two points will be exchanged. The chance for a mutation event is 10%. This rate is essential for passing from local minimum. In the long term, this rate of the mutation event results in an increase of the quality of chromosomes in the list [7].

IV. Robot Simulation and Training Algorithm

One of the most important parts of this research is simulating both robot and environment. Some authors have implemented the robot and a virtual world by simulation of dynamic rigid bodies. The robot depends on the physical parameters and is implemented in the environment. Each part of the robot has a mass, a center of mass, an elasticity parameter, and both dynamic and static friction coefficients.
Fig.3 shows an implemented “snake robot” in the simulator which has joints with 2 degrees of freedom. The hinges do not have any limitation in rotation. Nevertheless, applying limitation in rotation range is possible in this simulator for each joint separately.

According to Fig.1, each column of the CNN processor is connected to robot actuators. Hence, each actuator of the robot should connect to one of the columns separately. Since robot actuators’ response time is not equal for all of them we do assume/take the maximum delay for playing the wave on the robot actuators. This delay interval is essential for robot movement. The goal of the learning unit is that of finding optimum templates to move the robot according to our desires. Finding these templates for a specific movement mechanism/pattern is essential and suitable for use in a multi layer task manager or controlling unit. We are able to use these templates for a low level robotic activity. When a high level controller send commands to the robot for doing a specific task another controller needs to manage some low level skills like running, turning, jumping and so on, which are necessary to ensure the realization of the high level task [8-11]. Therefore, by understanding some template robot properties the high level task management is very simple in the high level controller. In the above referenced evaluation, authors tried to find lateral undulation locomotion for a snake robot. Each hinge has 2-DOF and can turn in 2 directions.

With this method based on genetic algorithms, an optimum template ensures that the robot can move or act according to our desires. The most important point in this learning method is that we don’t predefine any robot kinematics for movement in the fitness function. The fitness function is a simple and important function that defines the robot behavior in the environment. Complicated rules and equations in the fitness function can not improve the robot behavioral performance and at times a simple definition can result in the best robot behavior exactly.

Equation-2 shows the used fitness function for snake robot lateral undulation locomotion according to the Fig.3.

$$Fitness = RMS \times \frac{1}{AVG} \times DIST \tag{2}$$

$$RMS = \sqrt{\sum_{i=1}^{7} \left(AVG - L_i\right)^2} \tag{3}$$

$$AVG = \frac{\sum_{i=1}^{7} L_i}{7} \tag{4}$$

$$DIST = X_{L_1} \tag{5}$$

The term ‘AVG’ denotes the mean of distances between parts and ‘x’ axis. The term ‘L_i’ denotes the distance of the i’th part of snake robot to the ‘x’ axis. RMS denotes the roots mean square error of the robot part’s position to the ‘x’ axis. X_{L1} denotes the forward distance towards the ‘x’ axis.

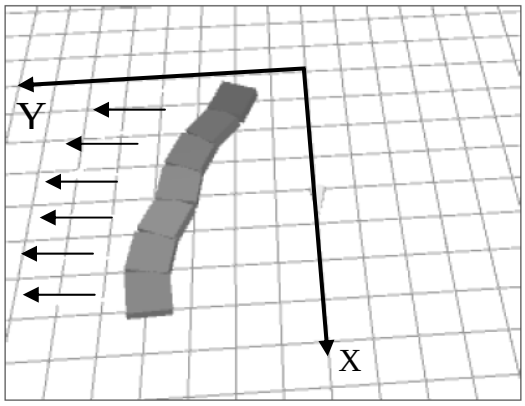

Figure 3. Snake robot lateral undulation locomotion

In lateral undulation locomotion, this term of the fitness-function must be close to zero.

This fitness-function defines the "snake robot" behavior for lateral undulation locomotion tasks. The first term in the fitness function shows that robot must keep itself in-line and moves parallel towards the 'x' axis. The second term shows that the robot must escape from the 'x' axis and the 3rd term in the fitness function shows that the robot mostly don't move in the frontal direction.

$$A = \begin{bmatrix} 0.8 & 0.01 & 1.79 \\ 3.44 & -2.85 & -4.89 \\ 4.33 & 2.18 & -4.17 \end{bmatrix}, B = \begin{bmatrix} -3.46 & -2.39 & -1.07 \\ 0.51 & 0.45 & -1.09 \\ 0.6 & -4.27 & -3.06 \end{bmatrix}, I = 3.29 \quad (6)$$

After a first generation of 100 chromosomes, the robot learns to move in lateral undulation with a new CNN template, which is generated by this evolution method. This template is shown in equation-6. A task manager in the high level can select a best template for performing a specific task by the robot. On the other hand, each template is a robot program for movement and for doing given tasks.

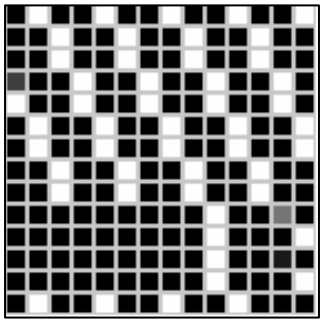

Figure 4. Wave generated for lateral undulation locomotion

In another evaluation we define a fitness function according to Equation-7. This function is defined for robot rectilinear locomotion with minimum sidle. According to this equation, each term must be close to zero. The first term shows that the robot must have a minimum deviance to the 'x' axis. The second term shows that the robot should not be away from this axis. And the last term shows that the robot must crawl on the 'x' axis. After each breed, a new chromosome will be added to chromosome population. After checking of new chromosomes by the fitness function, they will be sorted in a population list ordered by the best fitness. According to the evolution theory chromosomes in the future can/will inherit from the best and fit chromosomes.

After nearly 790 chromosome generations the robot will have learned to move with the highest speed. Since the evaluation time for calculation of fitness value is fixed and definitely the fastest chromosome is the chromosome that can move more than other chromosomes.

$$Fitness = RMS \times AVG \times \frac{1}{DIST} \quad (7)$$

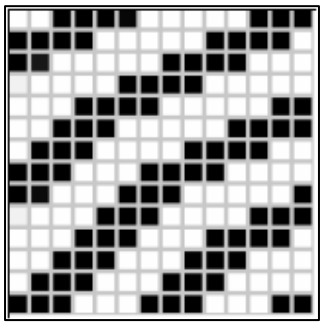

Figure 5. Wave generated for rectilinear locomotion

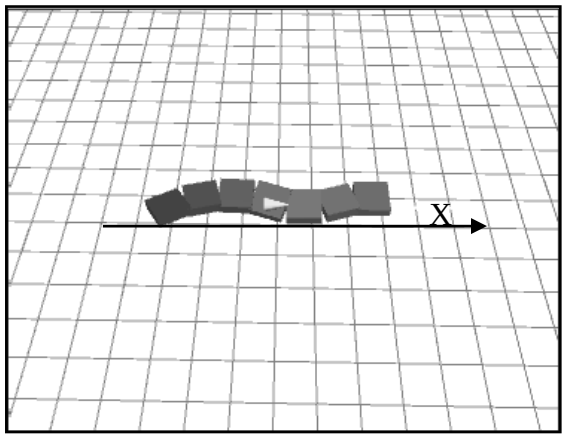

Figure 6. Snake Robot rectilinear locomotion

(8)

$$A=\begin{bmatrix}-2.67 & -2.74 & -2.01\\ -4.62 & -4.15 & 1.69\\ -0.92 & 2.1 & 0.58\end{bmatrix}, B=\begin{bmatrix}4.83 & 3.43 & -5\\ -0.97 & -2.39 & 4.27\\ -3.73 & 1.74 & -1.44\end{bmatrix}, I=3.05$$

With equation-8, the CNN processor can generate a hinge wave according to Fig.5. This wave is optimum for robot rectilinear locomotion by an evolution algorithm. Fig.6 shows the robot during the simulation in rectilinear locomotion.

Fig.7 shows the results from fitness function obtained after 790 chromosome generations; the robot has learned the best movement and locomotion. Extending this architecture or learning method is possible. By connecting the CNN outputs to unknown/arbitrary robot actuators, the robot can learn any locomotion. Due to the high capacity of CNN we can connect CNN's output to robot hinges actuators by any arrangement and structure. The results should be the same; but both the learning and optimization times might change. The implementation of CNN in this simulation is based on the Euler method.

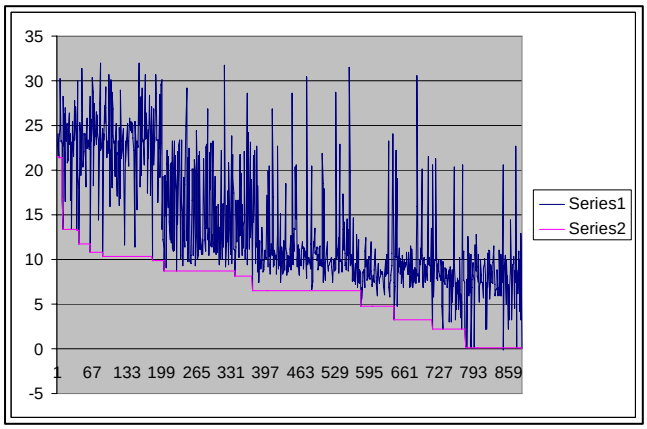

Figure 7. Series 1 is fitness-function value; Series2 is fitness-function minimum value, during cycle of time in learning process.

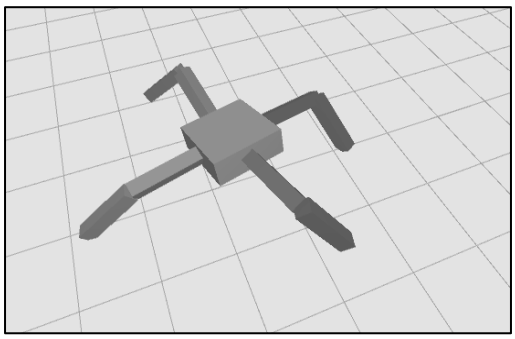

Figure 8. Learning 4legs semi-spider robot

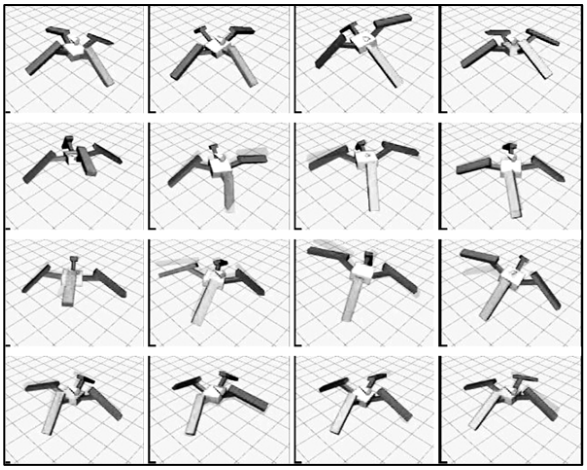

Figure 9. 4-leg robot spider, during the turning.

Fig.8 shows a spider robot with 4 legs and 16 degree of freedom. Each hinge has 2 degree of freedom in rotation. By a 16×16 CNN array this robot can move easily.
Fig.9 Shows the sequence of robot locomotion after learning. In this test, robot must turn around the 'z' axis.

V. Conclusion

In this paper, a concept has been presented that shows how a robot can learn locomotion based on an evolutionary technique. This research showed that combining CNN with an evolution scheme like GA is very effective and suitable for learning for different types of robots: high DOF robots, symmetrical, unsymmetrical and defective robots. Due to the intrinsic characteristics of CNN, this type of neural network is very close to natural processors and robot controllers.

Future Works

Finding evolutional structure for robot, and implementation of a physical robot based on this method is our aim for the future. One of the main important topics in this field is simulation of ecosystem and robot environment which are under consideration.

REFERENCES

[1]. Kozek, T., T. Roska, and L.O. Chua, *Genetic algorithm for CNN template learning.* Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on [see also Circuits and Systems I: Regular Papers, IEEE Transactions on], 1993. 40(6): p. 392-402.

[2]. Arena, P., L. Fortuna, and M. Branciforte, *Reaction-diffusion CNN algorithms to generate and controlartificial locomotion.* Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on [see also Circuits and Systems I: Regular Papers, IEEE Transactions on], 1999. 46(2): p. 253-260.

[3]. Arena, P., et al., *An autonomous mini-hexapod robot controlled through a CNN-based CPG VLSI chip.* Cellular Neural Networks and Their Applications, 2006. CNNA'06. 10th International Workshop on, 2006: p. 1-6.

[4]. Chua, L.O. and T. Roska, *Cellular Neural Networks and Visual Computing: Foundation and Applications*. 2002: Cambridge University Press.

[5]. Cho, S.B., *Evolving multiple sensory-motor controllers based on cellularneural network.* IFSA World Congress and 20th NAFIPS International Conference, 2001. Joint 9th, 2001. 4.

[6]. Cigale, B. and D. Zazula, Segmentation Of Ovarian Ultrasound Images Using Cellular Neural Networks. International Journal of Pattern Recognition and Artificial Intelligence, 2004. 18(4): p. 563-581.

[7]. Bilotta, E., G. Cutri, and P. Panano, *Evolving Robot's Behavior by Using CNNs.* Proc. Ninth Int. Conf Simulation of Adaptive Behavior (sab'06) From Animals to Animats, 2006. 9: p. 631-639.

[8]. Arena, P., et al., *An adaptive, self-organizing dynamical system for hierarchical control of bio-inspired locomotion.* Systems, Man and Cybernetics, Part B, IEEE Transactions on, 2004. 34(4): p. 1823-1837.

[9]. Muthuswamy, B., *Implementing Central Pattern Generators for Bipedal Walkers using Cellular Neural Networks.* 2005.

[10]. Arena, P., et al., *Realization of a CNN-driven cockroach-inspired robot.* Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on, 2006: p. 4.

[11]. Arena, P., et al., *Bio-inspired robotics: application of a CNN-based CPG VLSI chip to control an autonomous mini-hexapod robot.* Cellular Neural Networks and Their Applications, 2006. CNNA'06. 10th International Workshop on, 2006: p. 1-1.