Cellular Neural Networks-Based Genetic Algorithm for Optimizing the Behavior of an Unstructured Robot

Alireza Fasih
Transportation Informatics Group, Institute of Smart Systems Technologies, University of Klagenfurt
Klagenfurt, Austria
E-mail: alireza.fasih@uni-klu.ac.at

Jean Chamberlain Chedjou
Transportation Informatics Group, Institute of Smart Systems Technologies, University of Klagenfurt
E-mail: jean.chedjou@uni-klu.ac.at

Kyandoghere Kyamakya
Transportation Informatics Group, Institute of Smart Systems Technologies, University of Klagenfurt
E-mail: kyandoghere.kyamakya@uni-klu.ac.at

Abstract

A new learning algorithm for advanced robot locomotion is presented in this paper. This method involves both Cellular Neural Networks (CNN) technology and an evolutionary process based on genetic algorithm (GA) for a learning process. Learning is formulated as an optimization problem. CNN Templates are derived by GA after an optimization process. Through these templates the CNN computation platform generates a specific wave leading to the best motion of a walker robot. It is demonstrated that due to the new method presented in this paper an irregular and even a disjointed walker robot can successfully move with the highest performance.

Keywords: Cellular Neural Networks, Robot locomotion, Simulation, Genetic Algorithms.

1. Introduction

Nowadays, 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, the nature selects the powerful and stable genes for breed, and weak genes fall/disappear in the nature [1]. The good genes that can adapt the animal structure to the environment have higher chances for breed and evolution. The animal locomotion is trained and adapted according to the animal's body structure. One key issue in the training process is based on the energy saving. This justifies the striking interest devoted to the modeling and simulation of animal walking motion with the aim of optimizing the energy consumption [2-4]. It is well-known that the walking motion of animals is of a stereotype. In a large variety of animals a central neural controller does organize/coordinate the motion. A central neural controller (e.g. the central pattern generator (CPG)) is a main unit for controlling limbs for walking [5]. The CPG unit does contain all the mechanisms needed to generate the rhythmic pattern of movement. This unit is suitable for designing walker, swimmer, or flyer robots which are close to natural locomotion mechanisms/behaviors. Due to recent advances in electronics and the ability of cellular neural networks to solve partial differential equations in real time, it is possible to simulate a Reaction-Diffusion model by a

specific CNN architecture, the so-called reaction-diffusion cellular neural network (RD-CNN).

A striking interest has been devoted to the robot control based on the RD-CNN technique [1, 5, 6]. In this technique, the mathematical model describing the robot behavior must be well-defined. This is a serious limitation as modeling the complex behavior of robots is challenging. In this paper we introduce a robots control method which is not based on the mathematical modeling of the robots behavior. It is rather a general and effective method combining CNN with genetic algorithms (GA). This method can support and drive many types of structured and unstructured walker robots. The method/approach is based on both the natural modeling and the use of computational units close to biological models. A combination of both CNN (i.e. for computation) and GA (i.e. for optimizing the nature) is a good tool for modeling and controlling robots dynamics. The central parts of this scheme are made-up of a CNN processor and an evolutionary training unit. A Cellular neural network (CNN) is a parallel computing paradigm similar to the artificial neural networks computation platform, with the difference that in CNN the communication is allowed between neighboring units. This feature of the CNN processor makes it a good computation platform to analyze the dynamics of biological neurons. This paper shows the possibility of directly driving a walker robot by an evolutionary training of a CNN processor. This method is further efficient to model widespread natural locomotion mechanisms of animals (e.g. worms, insects, quadrupeds, biped, etc) [7]. This locomotion is modeled in the 3D space describing the real environment and in very difficult situations (i.e. rough, bumpy, and/or scaly surfaces) as well. The challenging focus is finding the best signal for driving walker robot joints with minimum energy consumption and the best locomotion performance. This can be achieved by finding suitable CNN templates to generate an efficient wave for driving the walker robot joints. This paper is organized as follows. Section 2 discusses the use of genetic algorithms is the optimization of CNN templates. Section 3 presents the training algorithm and some simulation results as well. To finish, section 4 formulates some concluding remarks. Further, the quintessence of the obtained results is summarized, and some open research questions are outlined.

2. Using Genetic Algorithms for CNN template optimization

The concept of Cellular Neural Networks (CNN) was introduced by Leon O. Chua and Yang [8]. CNN is a computation platform which is mathematically modeled by Eq. 1

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

where, 'T_A' denotes the 3×3 feedback template and 'T_B' stands for the 3×3 control template. 'I' is a bias value and 'y' is the linear output sigmoid function of each cell. 'u' denotes the input value and 'x' is the state of each cell. The input value is discretised into pixels and represented in a table of numbers called matrix. The size of this matrix depends upon the number of joints in the walker robot. In Eq. 1, the stars stand for convolution operations.

The genetic algorithm (GA) is a heuristic search technique used in computing to find either exact or approximate solutions for optimizing a given problem. The GA is an evolutionary algorithm that uses techniques inspired from biology such as inheritance, mutation, selection, and crossover. In this paper, this algorithm is used for finding the best templates for optimum robots locomotion.

The complete structure of the system used for the training process is shown in Fig. 1. This structure consists of six main parts: (1) Initial Population; (2) Crossover; (3) Mutation; (4) Fitness Function; (5) Decoding; (6) Cellular Neural Network Simulator. In Fig. 2 the connections between the robot actuators/hinges and the CNN outputs are shown. These connections are exploited in the control of both robot hinges and actuators. Wave rhythms are generated from the CNN processor outputs which can drive the walker robot on a specific path and/or direction depending on the high level task each of which consists of many low level tasks. After the learning phase, output waves can drive the robot with a minimum energy and a good efficiency. This driving depends upon specific choices of templates values. Each templates set is a solution for

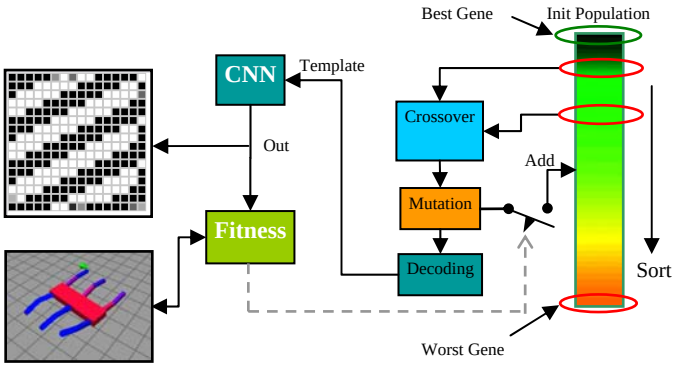

Fig. 1. System Architecture Diagram

driving the robot by means of (or by performing) some specific low level tasks. For optimizing these solutions,

the templates must be coded as chromosomes as shown in Fig. 3. In the initialization phase, Fig. 3 generates many random chromosomes (in this case CNN templates), each being a solution for driving the robot. In fact, each chromosome is a CNN template that is reshaped in a one dimensional array. According to Fig. 3, each chromosome does contain a feedback template, a control template and a bias value.

Various methods exist (in genetic algorithms) for coding data as chromosomes. This paper implements two different methods for coding and generating chromosomes. The first method is based on the IEEE-754 scheme which is a floating point technique. In this technique, each value must be converted to binary format according to the IEEE-754 floating point technique. The IEEE floating point format consists of three main parts: the sign, the exponent, and the mantissa [12]. The number of bits for each field is shown in the table below.

Table 1: Single Precision - IEEE Floating Point Format Structure

Sign	Exponent	Mantissa
1 bit	8 bit	23 bit

With the floating data types mentioned in Table 1, it is possible to store values between the ranges $\left[1.5\times10^{-45},\ 3.4\times10^{38}\right]$. The use of this method as a gene coder requires the definition of a mask for some bits. Otherwise, the random chromosome generator will generate values out of the range $\left[-5V,\ +5V\right]$. This condition is of high importance as a hardware implementation (using TTL devices) of this algorithm is under consideration. In the second method implemented, a "real" data type value is used as a chromosome coding. For this step, a random function generates a value in the acceptable range. The implementation of this method is easier than to first method. The results from the two methods were compared and a very good similarity was obtained between them. Nevertheless, the convergence time in binary coding was 10 percent faster. One particular important part of this algorithm is the design of the fitness function. This function or cost function defines/fixes indirectly the robot behavior [9]. This function is a particular type of objective function that quantifies the optimality of a solution in Genetic Algorithms. The input data for the fitness function are based on measurements of robots' parts orientation, location and displacement. In the fitness function we

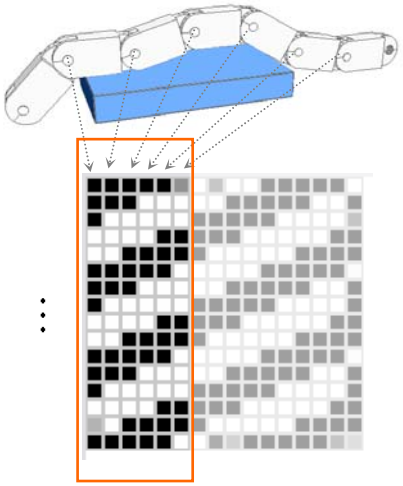

Fig. 2. Robot hinges connection to CNN

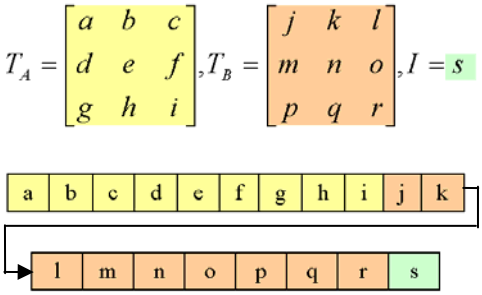

Fig. 3. Template Encoding in an Array List

don't define any behavioral locomotion exactly, like a robot kinematics. On the other hand, we define a function that satisfies the target or destination without any details. For example, in some circumstances, we need the robot to escape from a position without any specific direction. For this purpose, we must define a simple function for measuring the length between the central point (i.e. the gravity center) of the robot and the initial position. In this example, we didn't define any detail for the locomotion behavior. After generating new chromosomes, we obtain the corresponding fitness value by applying the fitness function. The main point on applying the fitness function is that this function is not a real time procedure and that the result from the fitness calculation 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 (i.e. one period of the wave acting on the actuator) is not sufficient for measuring with good accuracy the position in space of the robot. Many cycles of the wave generated are necessary to be applied to robot actuators. By measuring some robot parameters like the position

of the robot central point, the robot angle (related to the global coordinates of the system) and so on, the fitness function is quantified. In the initial state of the training phase the algorithm selects some randomly generated chromosomes. There is no rule on how many chromosomes should be generated in the initial population. But this number varies depending upon the complexity of the problem [10]. Some authors have defined 100 generations of chromosomes/genes for the initial state [10-12]. After each generation, a fitness function is used to evaluate the cost of chromosomes in the simulator leading to maximum efficiency. During our computations, each evaluation took approx. 3 seconds and the program spent approx. 60 seconds to evaluate appropriated chromosomes. After this step, chromosomes and fitness values will be sorted with the aim/goal of minimum fitness values in a link list. The next step concerns the crossover (i.e. both selection and breed) of chromosomes. Our experiments have shown that 50% of the best chromosomes are fitting for the crossover. It was found that this range has a good probability to generating the better chromosomes. In each step, we randomly select 2 chromosomes in this range for the crossover process. Many evaluations have shown that the use of the "two-point" technique for the crossover is the best solution. In this process we define two points randomly on the selected chromosomes; the contents of the chromosomes between these two points are exchanged (Fig. 4). In Fig. 4 P1 and P2 are two randomly selected points. S1 and S2 are two selected parents/chromosomes. The crossover leads to two new "children" (see Ch1 and Ch2 in Fig. 4) with new properties. During the trial and error process, we obtained that the good probability for mutation is around 10%. This rate is essential for avoiding the local minimum trap. In the long term, this rate of the mutation event results in an increase of the quality of chromosomes in the list [13].

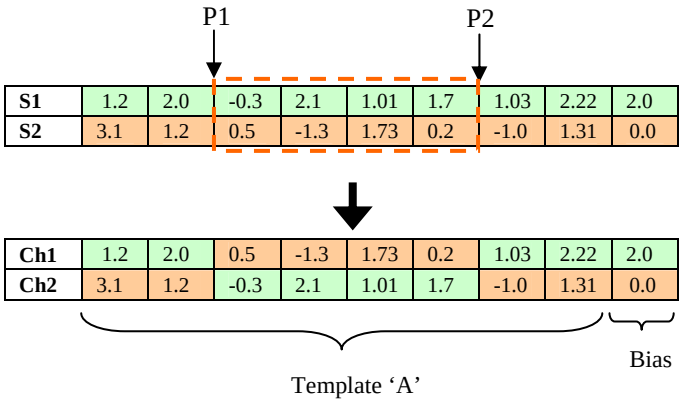

Fig. 4. Two-Point Crossover Method for Template 'A'

3. Training Algorithm and Simulation Results

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 [14, 15]. 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. 5 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 the rotation range is possible in this simulator for each joint separately. According to Fig. 2, each column of the CNN processor is connected to robot actuators. Hence, each actuator of the robot should be connected to one of the columns separately. Since the 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 the robot locomotion/movement. The goal of the learning process 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 the use in a multi layer tasks manager or controlling unit. We are able to use these templates for a low level robotic activity. When a high level controller sends commands to the robot for performing 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 [2, 3, 16, 17]. 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 two degree of freedom (2-DOF) and can turn in 2 directions. With the 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/locomotion 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. Eqs.-2 define the fitness function used for the snake robot lateral undulation locomotion according to Fig. 5.

$$Fitness = RMS \times \frac{1}{AVG} \times DIST \quad (2.a)$$

$$RMS = \sqrt{\sum_{i=1}^{7} (AVG - L_i)^2} \quad (2.b)$$

$$AVG = \frac{\sum_{i=1}^{7} L_i}{7} \quad (2.c)$$

$$DIST = X_{L_1} \quad (2.d)$$

The term 'AVG' denotes the mean 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} stands for the forward distance towards the 'x' axis.

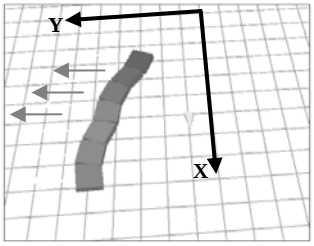

Fig. 5. Snake robot lateral undulation locomotion

In the 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 (*RMS*) term in the fitness function shows that the robot must keep itself in-line and move parallel towards the 'x' axis. The second term (*AVG*) shows that the robot must escape from the 'x' axis and the 3rd term (*DIST*) in the fitness function shows that the robot mostly don't move in the frontal direction.

$$Ta = \begin{bmatrix} 0.8 & 0.01 & 1.79 \\ 3.44 & -2.85 & -4.89 \\ 4.33 & 2.18 & -4.17 \end{bmatrix}, Tb = \begin{bmatrix} -3.46 & -2.39 & -1.07 \\ 0.51 & 0.45 & -1.09 \\ 0.6 & -4.27 & -3.06 \end{bmatrix}, I = 3.29 \quad (3)$$

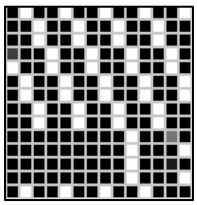

Fig. 6. Wave generated for lateral undulation locomotion

After a first generation of 100 chromosomes, the robot learns to move in the lateral undulation with a new CNN template, which is generated by this evolution method. This template is shown in Eq. (3). 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/locomotion and for performing given tasks. In another evaluation we define a fitness function according to Eq. (4). This function is defined for robot rectilinear locomotion with a minimum sidle. According to this equation, each term must be close to zero. The first (*RMS*) term shows that the robot must have a minimum deviance to the 'x' axis. The second (*AVG*) term shows that the robot should not be away from this axis. And the last term (*DIST)* shows that the robot must crawl on the 'x' axis. After each breed, a new chromosome will be added to the chromosome population. After the 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, after many generations, some chromosomes ("children") can inherit good properties from others ("parents") which are best and fit chromosomes.

After nearly 790 chromosome generations the robot will have learned to move with the highest speed. With Eq. (5), the CNN processor can generate a hinge wave according to Fig. 7. This wave is optimum for the robot rectilinear locomotion by an evolution algorithm. Fig. 8 shows the robot during the simulation in rectilinear locomotion. Fig. 9 is the plot of the time evolution of the fitness function obtained after 790 chromosome generations; the robot has learned the best movement and locomotion. The extension of this architecture or learning method to another kind of robot 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 although both learning and optimization times might change.

$$Fitness = RMS \times AVG \times \frac{1}{DIST} \quad (4)$$

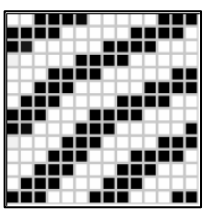

Fig. 7. Wave generated for rectilinear locomotion

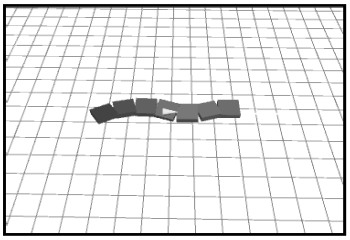

Fig. 8. Snake Robot rectilinear locomotion

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

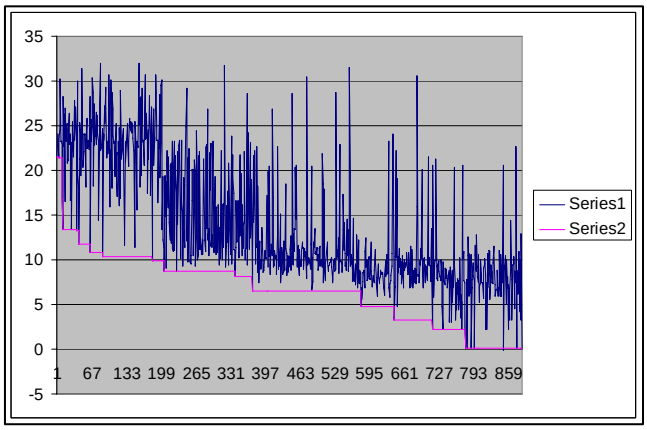

Fig. 9. Series 1 is fitness-function value; Series2 is fitness-function minimum value, during cycle of time in learning process.

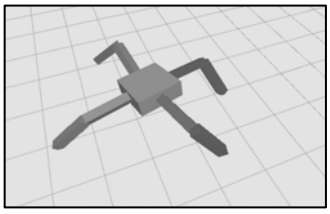

Fig. 10. Learning 4-legs semi-spider robot

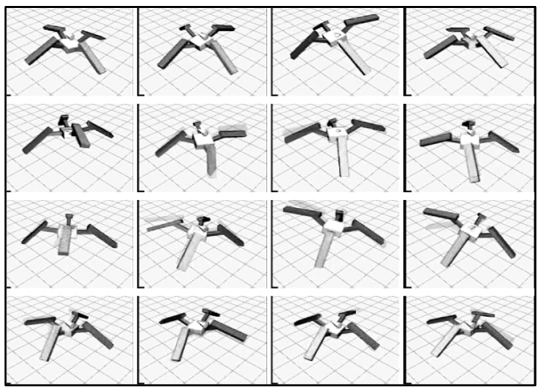

Fig. 11. 4-leg robot spider, during the turning.

Fig. 10 shows a spider robot with 4 legs and 16 degree of freedom. Each hinge has 2 degree of freedom in rotation. A 16×16 CNN array can be used to drive this robot. Fig.11 shows the sequences of the robot locomotion after the learning process. In this test, the robot must turn around the 'z' axis. Another test in Fig. 10 shows the design of a 6 legs insect robot for locomotion learning. The robot has 12 degree of freedom in hinges. In this case, the aim is moving around the circle with a given radius. After nearly 2500 iterations it was found that the result converged to zero. The CNN template shown in Eq. (6) is optimized for this purpose. The fitness function in Eq. (7) is used to generate the CNN output wave shown in Fig. 13.

$$T_A = \begin{bmatrix} 1.73 & -1.85 & 1.64 \\ 0.35 & -1.68 & -3.2 \\ -2.23 & -1.3 & 2.35 \end{bmatrix}, T_B = \begin{bmatrix} 4.03 & -3.05 & 3.52 \\ -3.2 & 2.61 & -2.49 \\ 4.88 & -3.02 & -0.95 \end{bmatrix}, I = -4.01 \quad (6)$$

According to Eq. (7), the fitness value has a direct relation with the distance between the initial position point (*Init_Center_Pos*) and the robot position (*Robot_Pos*) divided by (*R*). Further, this function has an inverse relation with the robot movement (*Robot_ Movement*).

$$Fitness = (1 - Dist(Init\_Center\_Pos, Robot\_Pos) / R) * (1/Robot\_Movement) \quad (7)$$

By optimization the robot movement/locomotion, the fitness value will converge to zero.

For the case of an unstructured robot, we designed a broken-leg spider. In this test, the aim is learning the robot for the turning left and right skill as a complete and perfect spider. In Fig. 14 is shown the representation of this type of robot.

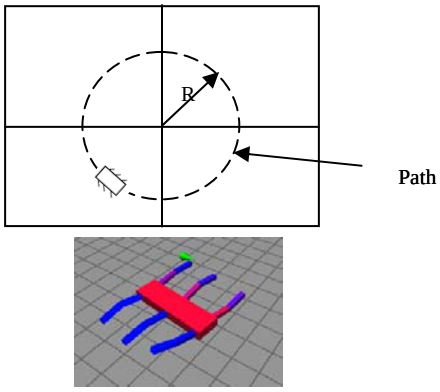

Fig. 12. Moving 6-Leg Robot, around the Circle

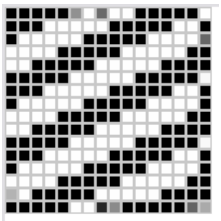

Fig. 13. Wave generated for circular locomotion

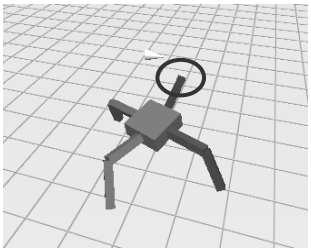

Fig. 14. Broken Leg Spider as an unstructured robot

The definition of the fitness function is a little bit sensitive in this case. According to Eq. (8), the fitness value has a direct relation to the distance between the initial position point (*Init_Center_Pos*) and the robot position (*Robot_Pos*). Further, this function has an inverse relation with the robot angle (*Robot_Angle*). During the optimization phase, the aim/goal is converging the fitness function to zero.

$$Fitness = Dist\,(Init\_Center\_Pos, Robot\_Pos) * (1/Robot\_Angle) \tag{8}$$

$$T_A = \begin{bmatrix} -0.61 & 1 & -3.85 \\ 4.61 & 0.21 & -4.77 \\ 0.97 & 3.25 & -4.17 \end{bmatrix}, T_B = \begin{bmatrix} 4.72 & 1.2 & -2.61 \\ -2.19 & 2.25 & 3.04 \\ -2.49 & 3.33 & 4.97 \end{bmatrix}, I = -3.92 \tag{9}$$

After nearly 3300 chromosome generations and evolution, the robot is capable to turning over on its Yaw axis. Fig. 15 shows the result of the wave pattern for unstructured spider turning. The usability of the templates in this paper can be summarized as follows. Templates are stored in a list/memory. By a high level task management templates are selected. This selection depends on the high level task management decision and the environment situation as well. Further, a factor of high importance is the behavioral architecture and behavioral programming. In fact, low level skills like moving forward, turning left and right, jumping, etc. for insuring the control of the robot are very important. The choice of templates is highly influenced by these factors.

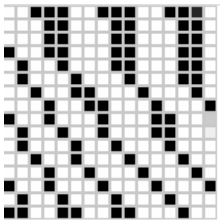

Fig. 15. Wave generated for Broken-Leg Spider; Turning Skill

4. Conclusion

This paper has presented a concept based on an evolutionary technique for the robot locomotion learning. The technique proposed was a combination of both CNN and genetic algorithms. The motivation of this combination can be justified by the high accuracy of the CNN processors and their good computational speed as well. Further, the topology of CNN is flexible for designing neuro-evolutive systems. The genetic algorithm was exploited for the training process in order to determine the best genes according to the pre-defined requirements (i.e. dada requirements) for the design process. Two types of robots were considered (i.e. both structured and unstructured robots). For each of these types, algorithms were developed to derive the appropriate chromosomes from which corresponding templates were derived. The results in this paper have shown that combining the cellular neural networks (CNN) technology with an evolution scheme like genetic algorithm (GA) is very effective and suitable for learning the movement /locomotion of different types of robots (e.g. high DOF robots, symmetrical, unsymmetrical and defective robots). Due to the intrinsic characteristics of the CNN, this type of neural network is very close to natural processors and therefore is efficient for building robot controllers. During the training process, we found that the complexity of the environment (e.g. rough, bumpy, and/or scaly surfaces) was a key factor influencing the results. Basically, the technique developed in this paper provided interesting results with high accuracy in complex environments. Nevertheless, we found that the accuracy of the results decreases with the increasing complexity of the environment (e.g. ecosystem and robot environment). An interesting issue under investigation in subsequent and future works is implementing/developing high accuracy methods for robot control in very difficult environments.

References

[1] P. Arena, L. Fortuna, and M. Branciforte, "Reaction-diffusion CNN algorithms to generate and control artificial locomotion," in *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications* 46(2), pp. 253-260, (1999).

[2] P. Arena, L. Fortuna, M. Frasca, L. Patane and M. Pollino, "Bio-inspired robotics: application of a CNN-based CPG VLSI chip to control an autonomous mini-hexapod robot," in *Proc. of the 10th International Workshop on Cellular Neural Networks and Their Applications (CNNA'06)*, pp. 1-1, (2006).

[3] P. Arena, L. Fortuna, M. Frasca and G. Sicurella, "An adaptive, self-organizing dynamical system for hierarchical control of bio-inspired locomotion," in *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 34(4) pp. 1823-1837, (2004).

[4] L. Fortuna, L. Patane, D. E. E. e Sistemistico and V. A. Doria, "Hexapod locomotion control through a CNN based decentralized system," in *International Symposium on Proceedings of the ISIE*, pp. 4, (2002).

[5] P. Arena, L. Fortuna, M. Frasca, L. Patane and M. Pollino, "An autonomous mini-hexapod robot controlled through a CNN-based CPG VLSI chip," in *Proc. of the 10th International Workshop on Cellular Neural Networks and Their Applications (CNNA'06)*, pp. 1-6, (2006).

[6] M. Branciforte, G. Di Bernardo, F. Doddo and L. Occhipinti, "Reaction-diffusion CNN design for a new class of biologically-inspired processors in artificial locomotion applications,".in *Proc. of the Seventh International Conference on Microelectronics for Neural, Fuzzy and Bio-Inspired Systems (MicroNeuro'99)*, pp. 69-76, (1999).

[7] Gabriele Manganaro, P. Arena, L. Fortuna, *Cellular Neural Networks: Chaos, Complexity and VLSI Processing*, Springer 1999, pp.152-161

[8] Chua, L.O. and L. Yang, "Cellular neural networks: theory," in *IEEE Transactions on Circuits and Systems* 35(10), pp. 1257-1272, (1988).

[9] Cho, S.B., "Evolving multiple sensory-motor controllers based on cellular neural networks," in *Proc. of IFSA World Congress and 20th NAFIPS International Conference 4*, pp. 2218-2222, (2001)

[10] Cigale, B. and D. Zazula, "Segmentation of Ovarian Ultrasound Images Using Cellular Neural Networks," in *International Journal of Pattern Recognition and Artificial Intelligence* 18(4), pp. 563-581, (2004).

[11] Beasley, J.E. and P.C. Chu, "A genetic algorithm for the set covering problem," in *European Journal of Operational Research* 94(2), pp. 392-404, (1996).

[12] Horn, J., N. Nafpliotis, and D.E. Goldberg, "A niched Pareto genetic algorithm for multiobjective optimization," in *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence 1*, pp. 82-87, (1994).

[13] Bilotta, E., G. Cutri, and P. Panano, "Evolving Robot's Behavior by Using CNNs," in *Proceedings of 9th Int. Conf Simulation of Adaptive Behavior (sab'06) From Animals to Animats 9*, pp. 631-639, (2006).

[14] Wolff, K. and P. Nordin, "Learning Biped Locomotion from First Principles on a Simulated Humanoid Robot Using Linear Genetic Programming," in *Lecture Notes in Computer Science*, pp. 495-506, (2003).

[15] D. Marbach and A. Ijspeert, "Co-evolution of configuration and control for homogenous modular robots," *in Proceedings of the Eighth Conference on Intelligent Autonomous Systems (IAS8)*, IOS Press, pp. 712–719, (2004).

[16] B. Muthuswamy, *Implementing central pattern generators for bipedal walkers using cellular neural networks*, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA, Masters thesis, (2005).

[17] P. Arena, L. Fortuna, M. Frasca, L. Patane and M. Pavone, "*Realization of a CNN-driven cockroach-inspired robot,"* in *Proceedings of Circuits and Systems. IEEE International Symposium on ISCAS*, pp. 4, (2006).