Study of Emergence of Locomotion in Hexapod using Genetic Algorithm

Abstract— **In nature, animals use networks of neurons in their brain to learn a task. And also, by various crossovers and mutations, every new generation of organism is better adapted to survive in the current environment. Here we use these two concepts – Genetic Algorithm & Neural Networks – to make a robot learn how to walk in a straight line. We do a computer simulation of a hexapod robot in 'pyrosim' simulator by programming the same in python language.**

Keywords— Robot Simulation, pyrosim, Genetic Algorithm, Neural Network.

I. Introduction

In nature, animals and insects move autonomously in their environment to perform their daily function. But to keep up with the everchanging environment, animals need to evolve with every new generation. An organism's physical traits (phenotype) are decided by its genetic makeup, which it gets through a crossover of its parent's genetics. So, every generation has a new genetic data. And, sometimes, there would be random mutations while copying genetic data from the parent – which lead to new physical traits. If these new traits assist the animal in the new environment, the next generation will again retain these genes. But if these newly formed traits hamper the survival of the animal dies. Thus, this process of natural selection verifies the theory of survival of fittest. Thus, its only through Genetic Evolution that animals evolve. So here we see that, although if we start with crude genetics; with crossovers and mutations, we will get different genetic makeup for everyone. And by taking fitness into the account, we will move towards the optimal genetics for the current environment.

This method of Genetic Algorithm could be used to optimize the performance of robots as well. And with the use of Neural networks along with Evolutionary Algorithm, we can make a robot learn and evolve to do any task – we award the 'genetic makeup' of a robot when it performs that particular task and penalize it on any other. This field of robotics is called Evolutionary Robotics.

Evolutionary robotics is a method of developing controllers for autonomous robotic functions using evolutionary computations [1]. Evolutionary computations involve the development of optimization algorithms inspired by biological evolutionary processes. So, the selection of controllers for the

robot is obtained through optimizations using algorithms. e.g.: genetic algorithms. In genetic algorithms, population of controllers is applied crossover and mutation and assessed according to fitness function so the highest fitness population is obtained. The controller are sometimes obtained from artificial neural networks.

An artificial network is an interconnected group of nodes, inspired by brain neurons. Neural networks are generally the controller used in evolutionary robotics [2]. The neural controller is represented by the connection weight used in it . So evolutionary computations are used to evolve the connection weights of the controllers to obtain an optimal solution to perform the described robotic function [4].

Genetic algorithm is an adaptive and heuristic evolutionary computational algorithms used for optimizing a given set of solutions [3]. In genetic algorithm, a population of candidate solution is evolved through crossover and mutation operations through a number of generations to obtain the heuristic optimal solution. The population of candidate solution is often obtained randomly. From this population, the individuals are selected for cross-over often by measuring the fitness values associated with each individual or randomly. The individuals, selected undergo crossover. Analogous to sexual reproduction in biology, the two parent chromosomes are recombined to obtain off springs during the crossover operation. After crossover, mutation is performed on the individuals obtained according to mutation probability. In mutation, the genetic value of the chromosome will be altered to perform genetic diversity. For each generation, the weightage is applied to the neural network to obtain the fitness values of the hexapod motion , thereby evolving the robotic motion units like feet or meters. Basically, it determines heading, pitch, roll

II. ELEMENTS

a. Simulator:

We use pyrosim (Python Robot Simulator) for simulation. It basically is a python interface primarily focused on neurally controlled robots. It uses C++ language for its physics engine, which then we can utilize by programming in python language. With it, we can create bodies (cylinders, boxes, etc.), joints and neural networks for the robot. We used this simulator on a Linux machine setup (Ubuntu 19.04).

b. Robot Model:

We model the hexapod robot using functions in above simulator python library. Structurally the hexapod has these components:

- Main central Body – box (1 * 2.2 * 0.1& mass of 10)
- 6 legs – cylinder (3 on each side, & each consisting of Femur and Tibia)
- 12 hinge joints (6 for hips, & 6 for knees)

c. Neural network:

For each of the 6 hip joints on the robot, we have one hidden neuron and one output motor neuron respectively. Hence in the neural network, the single hidden layer has 6 neurons which are then connected to the output layer of 6 neurons. So basically, there are 3 layers of neurons as shown below:

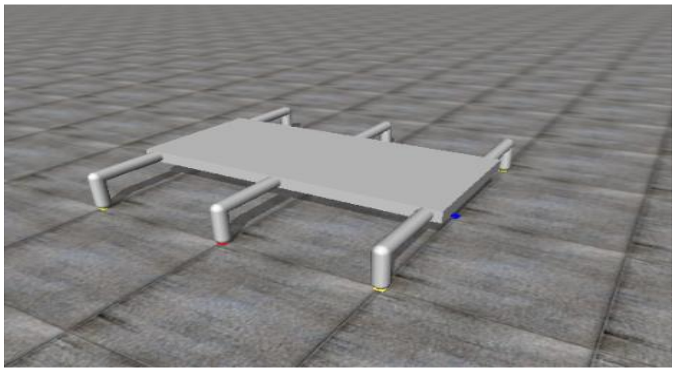
FIGURE 1

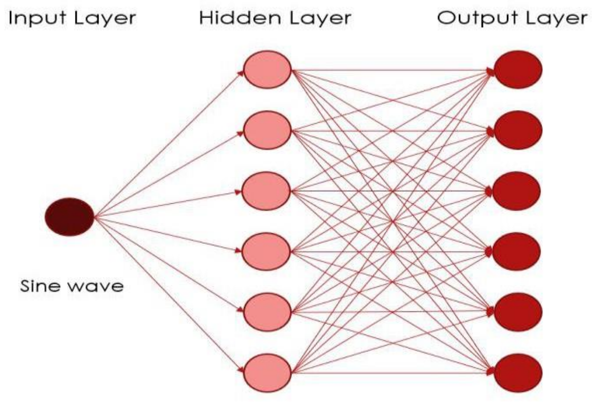

FIGURE 2

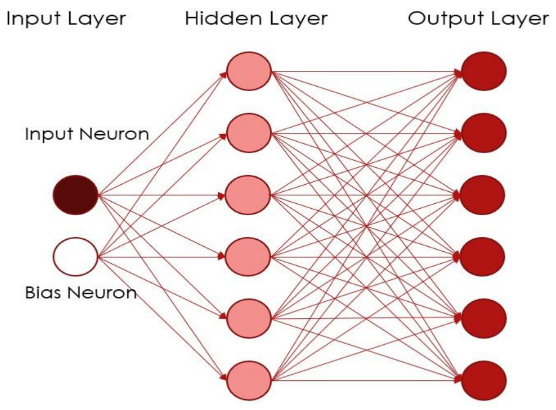

FIGURE 3

Inspired from central pattern generator, input to the function neuron is decided as the python in built sin function Hence, we consider the same for our model as well. And hence we give an input of sin wave to the input neuron

d. Evolutionary Algorithm:

For the above fixed neural network, we need to find out the weight in each synapse. In totality for case 1 we need 42 weights & for case 2 we need 49 weights. These synapse weights (1 * 42 vector for case 1) are then optimized for straight line locomotion by using Evolutionary Algorithm. Initially, we assign random weights to the neural network and check for the fitness of that model (the calculation of the fitness is explained below). Then the fittest amongst the initial population of weights are then selected to produce the next generation by a simple crossover of the weight vectors. Thus, the whole cycle repeats again until the desired number of iterations of generation is satisfied. Here each generation performs the given task better than the previous parent generation, hence having a higher fitness value. Also, there are added some amount of random mutations, which further adds a layer of external variables into the weights, which ultimately account for the fitness as well.

III. METHOD

We use a genetic algorithm library in python, 'pyeasyga' for the calculations with the below specifications for the function:

- Population Size= 15
- Number of generations = 20
- Crossover Probability = 0.8
- Mutation Probability = 0.2
- Elitism = True
- Maximum Fitness = True

We calculate fitness of the model with reference to our desired goal of straight-line locomotion. We simply penalize to any movement which does not contribute to our goal. We define fitness function as

$$-2*y - 10*\Sigma|X| - 10*\Sigma Z$$

Since the robot would be travelling in the –y axis direction, the first term would become positive. We penalize for any movement along the x direction (body moving in transverse direction) and z direction (Body touching ground) with a penalty of 10..

The robot has a position tracker on it which gives its direction at all time in all three coordinates. It also has a contact sensor which can detect contact of the main body with ground.

IV. RESULTS AND CONCLUSION

The gaits observed in both cases are almost similar. The motion of the legs is coupled in both cases. The motion is synchronized in which 3 legs move simultaneously as shown in figures which are taken while the robot is being simulated. The simulation was run for 20 generations by varying penalties and varying mutation probabilities and cross over probabilities. In most of the cases, the gait observed is similar to the synchronized motion observed. Even after adding a bias, the genetic algorithm converged to the synchronized motion. Each simulation has been run for 500 time steps. However, the genetic algorithm was able to find a feasible solution but it was not able to evolve to natural gait observed in hexapods that is tripod gait. This can be explained as neural networks have the flexibility to alter only the magnitude of the input sine function. Neural network cannot alter the phase of the input sine function for each leg. Hence all the legs can be related to a single phase of the sine function.

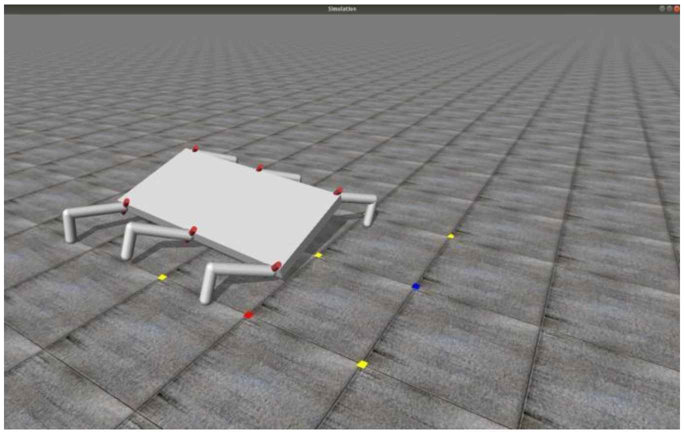
FIGURE 4

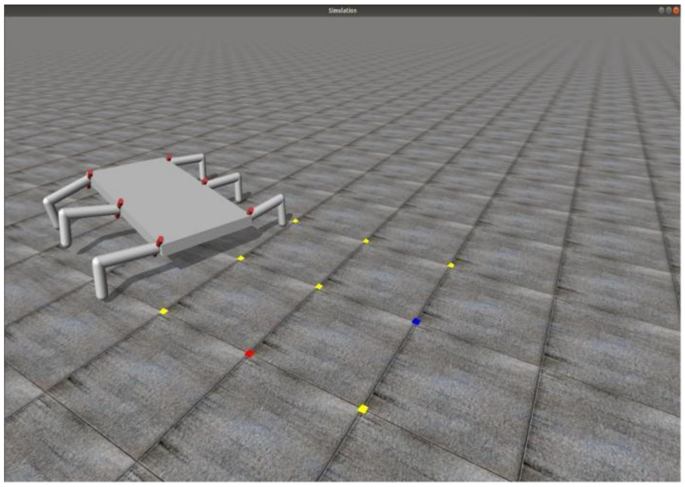
FIGURE 5

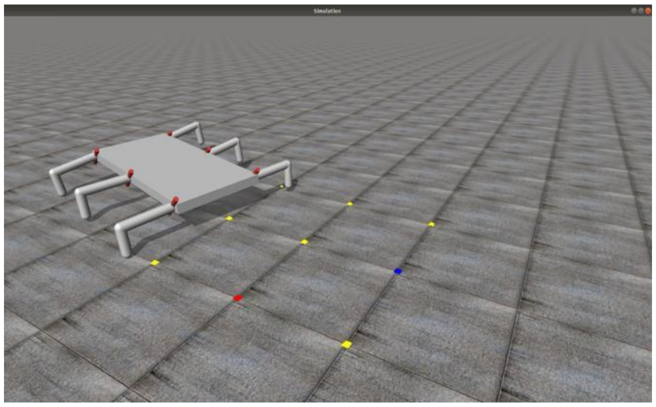
FIGURE 6

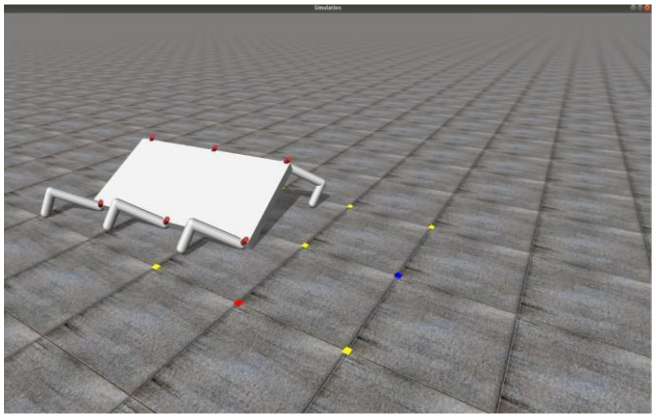
FIGURE 7

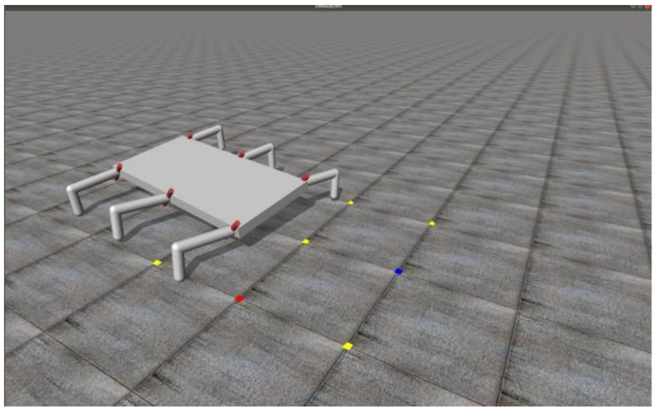
FIGURE 8

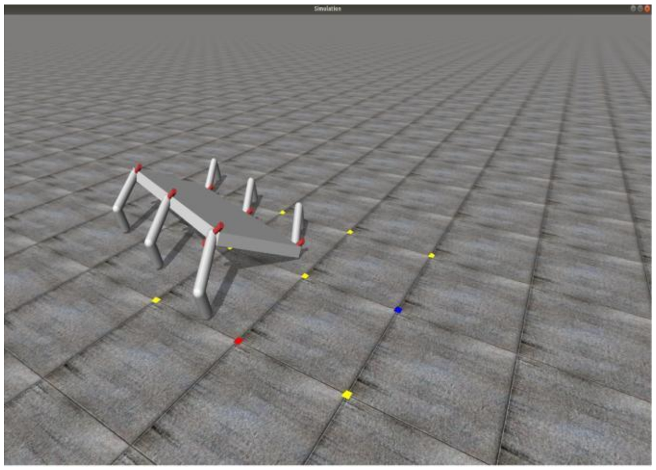
FIGURE 9

V.FUTURE WORK

Future work of this project includes evolving of tripod gait in a hexapod when both knee joint is fixed and free. Also the evolutionary algorithm doesn't change the neural network used and utilizes just the basic crossover and mutation to obtain the desired result. A powerful genetic algorithm can provide better results. After evolving the gait in a hexapod, the network should be implemented on the hexapod bot in real time to measure the functionality of the obtained network.

VI. REFERENCES

[1] Evolutionary neurocontrollers for autonomous mobile robots-
D. Floreanoa,*, F. Mondadaa,
Ab Laboratory of Microcomputing (LAMI), Swiss Federal Institute of Technology (EPFL), CH-1015 Lausanne, Switzerland
bK-Team SA, CH-1028 Pre ´verenges, Switzerland

[2] SimulatingtheEvolutionofSoftandRigid-BodyRobots-
http://dx.doi.org/10.1145/3067695.3082051

[3] A neuromechanical investigation of salamander locomotion- Auke Jan Ijspeert Brain Simulation Laboratory, Hedco Neuroscience Building, University of Southern California, Los Angeles 90089, U.S.A,

[4] A.H. Cohen. Evolution of the vertebrate central pattern generator for locomotion. In A. H. Cohen, S. Rossignol, and S. Grillner, editors, Neural control of rhythmic movements in vertebrates. Jon Wiley & Sons, 1988

[5] How To Evolve Autonomous Robots: Different Approaches In Evolutionary Robotics- https://www.researchgate.net/publication/37427167

[6] Evalution of a Subsumption architecture Neurocontroller- Julian Togelius

[7] O. Ekeberg A combined neuronal and mechanical model of fish swimming . Biological Cybernetics.

[8] W.C. Eurich, G. Roth, H. Schwegler, and W. Wiggers Simulander: A neural network model for the orientation movement of salamanders . Journal of comparative physiology.