Smooth Gait Generation for Quadrupedal Robots Based on Genetic Algorithm Optimization

Zainullah Khan, Farhat Naseer, Khawaja Fahad Iqbal, Sara Ali, Muhammad Sajid and Yasar Ayaz
Robotics and Intelligent Systems Engineering (RISE) Lab,
Department of Robotics and Artificial Intelligence,
School of Mechanical and Manufacturing Engineering (SMME),
National University of Sciences and Technology (NUST),
H-12 Campus, Islamabad, Pakistan
Email: {zain.9496, farhaty16}@gmail.com, {fahad.iqbal, sarababer, m.sajid, yasar}@smme.nust.edu.pk

***Abstract*—Gait generation is the process of finding a sequence of robot leg movements, which propel the robot in the desired direction when executed in a certain order. It is an optimization problem where multiple parameters need to be tuned in order to generate an optimal gait. In this paper, we propose a novel technique to improve the gait quality of a quadrupedal robot. In our proposed technique, we create an optimal fitness function for a Genetic Algorithm (GA) optimizer and use a trapezoidal velocity profile for joint movements. Our quadrupedal robot consists of 8 joints, 2 per leg. All joints are actuated by servo motors. The robot joints are controlled using a single layer Artificial Neural Network (ANN) whose inputs are the current robot joint angles and outputs are the target joint angles. The ANN is called every time the joints reach their target positions. A GA is used to optimize the ANN weights. The GA runs for a total of 100 generations over a population size of 10. The fitness function is a combination of the total distance traveled by the robot, and a scaling factor for the fitness value based on the overall joint movements. This discourages the GA from optimizing gaits that tend to an idle state. The controllers are selected based on how well they maximize the fitness function. The simulation of the robot is carried out in Open Dynamics Engine (ODE). The results show that the proposed technique considerably improves the overall fitness of the gait and the total distance traveled by the robot. Moreover, the proposed technique converges to an optimal gait in under 20 generations whereas the existing method takes over 40 generations. Furthermore, the robot joint movement is much smoother in the proposed method hence reducing the jerking in the robot motion.**

***Index Terms*—Gait Generation, Quadrupedal Robot, Genetic Algorithm, Neuro-evolution, Gait Evolution.**

I. Introduction

Mobile robots must have the ability to traverse the area in which they operate. This is relatively easy for wheeled robots. However, mobility is not that easy for legged robots as it involves controlling multiple Degrees of Freedom (DoF) of a robot. Successfully actuating the robot joints in a certain sequence can propel the robot in the desired direction. However, finding the right sequence is a very difficult task in robotics as it involves a number of parameters associated with each DoF of the robot. These parameters need to be optimized in order to make the robot move from one position to another. The optimization of these parameters can be done with a number of methods.

Gait generation in the past has been done using a number of methods. In [1], [2] and [3] authors have applied linear transforms to the joint sensor values to produce actuator movements which helps in propelling the robot forward. Since this is a direct transformation, the actuation is solely dependent on this linear transformation, therefore, any minute noise in the system can cause instability which can result in the robot falling over.

In [4] and [5] authors use decomposition-based optimization to split the single problem of optimizing multiple parameters into multiple smaller problems of optimizing single parameters. However, decomposition-based optimizations can take a very long time to converge to an optimal solution.

The authors in [6] and [7] generate gaits for their quadrupedal robot by defining a foot trajectory that each leg of the robot must follow sequentially. This method requires Inverse Kinematics calculations for every single point on the foot trajectory that must be visited. This method often yields slow gaits [8] [9]. Moreover, the fixed shape of the foot trajectory is not optimal for different terrain types [10] [11].

The authors in [12] use Central Pattern Generators (CPG) to generate gaits for a quadrupedal robot. CPGs use oscillators to generate gaits that are cyclical in nature. CPGs have been widely used to generate robot gaits. Another method that is similar to CPGs are Compositional Pattern Producing Networks (CPPNs). CPPNs also use oscillatory signal to generate a gait, however, CPPNs also depend on robot sensor input. CPPNs can produce gaits that are more complex and stable gaits [13]. However, as highlighted by [14], CPPNs can take a very long time to converge to an optimal solution.

Genetic Algorithm (GA) is also used to optimize gait for quadrupedal robots. In [15] and [16] the authors generate gaits for the AIBO quadrupedal robot. In [15] the gait was generated by optimizing an optimal circular leg motion, while in [16] the gait was generated by producing a wheel-like motion. In [17] Authors use GA on a hexapod. The foot positions and animations of the robot are used as a genome and GA is used to find best performing animations. In [18] the authors use GA to optimize an Artificial Neural Network

978-1-6654-6896-1/22/$31.00 ©2022 IEEE

2022 2nd International Conference on Artificial Intelligence (ICAI) | 978-1-6654-6896-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICAI55435.2022.9773617

(ANN) controller to generate gaits for a quadrupedal robot. The authors successfully generate gaits for the robot. The results produced by [18] showed that GA can converge to an optimal solution in as little as 100 generations. However, the simulation in [18] lacked joint velocity profile and the fitness function did not penalize the controller when out of range target values were produced for the joints.

The aforementioned methods of gait generation methods may yield gaits that can cause the robot to fall over. These methods can also take a very long time to converge to an optimal robot gait.

In this paper we address the problem of unstable gait generation by designing a new fitness function that penalizes the robot controller when the robot falls over or produces an unstable gait. In order to further stabilize the gait we have introduced trapezoidal joint velocity profile to make the joint movement smoother. Moreover, our robot controller is called only when all the robot joints have reached their target positions, this further helps in generating a stable robot gait. We use GA to optimize our controller to address the convergence problem faced in the other methods because GAs, when used with a large population size, can converge to the optimal solution relatively quickly.

The remainder of the paper is ordered as follows; In section 2 we will talk about our robot design, in section 3 we will discuss about the controller that we are using and we will briefly discuss our optimization method. In section 4 we will talk about our experiments and finally in section 5 we will talk discuss our results and compare them to the results produced by [18].

II. Robot Design

Our quadrupedal robot has 2 segments per leg and there are 2 joints per leg of the robot. There are a total of 8 joints. The robot joints are modeled after the MG996R servo motors. The robot joints are constrained to move between -60 and +60 degrees.

A touch sensor has been attached to the robot body that detects when the robot body is touching the ground.

In order to minimize the strain on the robot joints a trapezoidal velocity profile has been used which helps the joints to initially accelerate slowly until they reach the top motor speed, that's when the motors move with constant speed and then they start decelerating as they approach their target angle and eventually come to rest.

The robot body is shown Fig. 1.

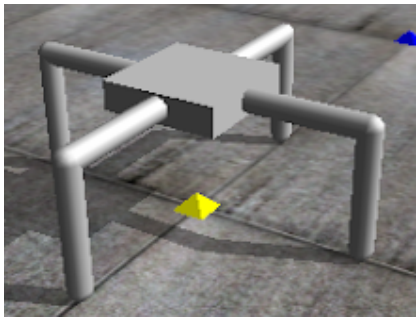

Fig. 1. The robot body.

III. Controller

The robot joints are controlled using a single ANN. The ANN has 9 inputs and 8 outputs. The first 8 inputs of the ANN take the current joint angle of the robot and the 9th input is a bias neuron. The ANN outputs the target angles for each of the robot joints. The ANN is shown in Fig. 2.

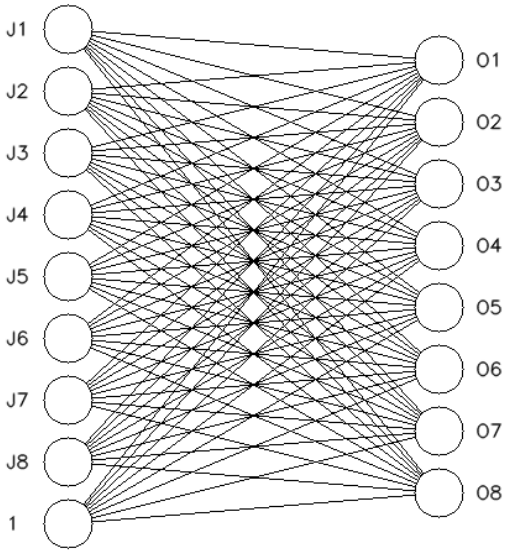

Fig. 2. The ANN that is used to control the robot joint motors.

The activation function used at the output of the ANN is the hyperbolic tangent, which gives a value between -1 and 1. This value is scaled by multiplying it with the permissible robot joint range. The activation function function formula is given in Eq. (1).

$$O = tanh(w.J) \tag{1}$$

Where w is the weight matrix and has dimensions of 8 x 9, J represents the ANN input vector and has 9 x 1 dimensions. The output is represented by vector O which has 8 x 1 dimensions.

The ANN controller is optimized using GA. The GA creates a population on the ANN weights and then uses them to control the robot. Depending on the performance of the ANN controller, they are selected to be part of the next generation. The ANN controller weights are also mutated slightly to introduce variety. The new population is tested again. This process goes on until a set number of generations have passed. The flowchart of the GA that is used in this paper is given in Fig. 3.

IV. EXPERIMENTS

A. Simulation

The robot is simulated in the open source physics engine Open Dynamics Engine (ODE) [19], where we have set up an obstacle free flat terrain. The GA spawns a population of 10 robots on the terrain and sets the default position of all the robot joints to 0 radians initially. Each robot has its own ANN controller, the weights of the ANN are initialized at random in the first generation. The robots are all simulated for 20 seconds, at the end of the simulation the robots are evaluated using our fitness function, which is detailed in section 5B.

After evaluation, the best performing controller is selected and its weights are copied over to the rest of the population. All of the controllers inside the population except for one are mutated. This ensures that even if all the controllers in the next generation produce low fitness results, the best performing controller is preserved and can be copied over again to make further changes. Every single controller weight has a 5% probability of being mutated.

The complete flowchart of the GA optimization is given in Fig. 3.

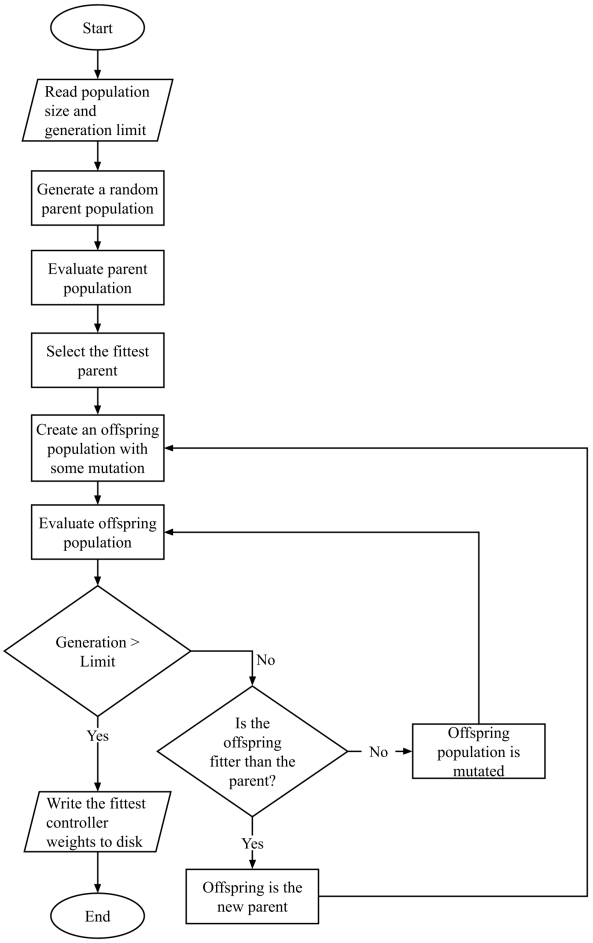

Fig. 3. The flowchart of the GA that is used to optimize the robot controller.

The ODE simulation environment parameters that we used for our experiments are given in the TABLE I.

TABLE I
SIMULATION PARAMETERS

Parameter Name	Parameter Value
Simulation Time	20s
Step Time	0.005s
Gravity	-9.8 ms^{-2}
Number of Joints	8
Motor Torque	11 $kgcm$
Motor Speed	$0.17s/60^o$
Motor Mass	55 $grams$
Robot Body Length	20 cm
Robot Leg Length	20 cm
Robot Height	20 cm
Robot Body Mass	130 $grams$
Robot Leg Mass	100 $grams$
Population	10
Generations	100
Mutation Rate	5%

The GA optimizes the controller for 100 generations with a population size of 10.

B. Selection Criteria

Our GA selects the best performing robot based on our fitness function which is composed of four different parts. These individual parts are given below.

$$f1 = \sqrt{x^2 + y^2} \tag{2}$$

Where x and y are the final position of the robot inside the simulation, therefore, Eq. (2) represents the distance traveled by the robot in the simulation.

$$f2 = \sum_{t=1}^{T} \sum_{j=0}^{J} ||\theta_{t-1,j}| - |\theta_{t,j}|| \tag{3}$$

T is the total number of timesteps in the simulation, J represents the total number of robot joints. $\theta_{t,j}$ represents the value of joint j at timestep t. Eq. (3) calculates the total movement of the robot joints during the simulation. This part of the function ensures that idle robots receive a lower fitness value, hence promoting more active robots in the GA selection.

$$f3 = 1 - TS \tag{4}$$

TS holds the value of the touch sensor mounted on the back of the robot. The value of the touch sensor is 1 when it is touching the ground. Hence, Eq. (4) assigns a fitness value of 0 to all the robots that turn over during the simulation.

$$f4 = \frac{1}{OR + 1} \tag{5}$$

OR represents the number of times the controller produces angle values that are over the permissible range. The value of Eq. (5) penalizes the controllers that produce target angles that are outside the robot constraints.

$$fitness = f1.f2.f3.f4 \tag{6}$$

Finally, the product of all the values is used to measur the total fitness of the robot given in Eq. (6). We are selectin robots that have the highest fitness value. However, it should b noted that our fitness function mainly depends on the euclidea distance travelled by the robot, given by Eq. (2). The rest of th parts of the fitness functions scale the actual distance travelle by the robot. Therefore, the fitness function selects robots tha have travelled the farthest from their spawn location.

V. Discussion

A. Fitness

The fitness plot of our method compared to results of [18 is given is Fig. 4.

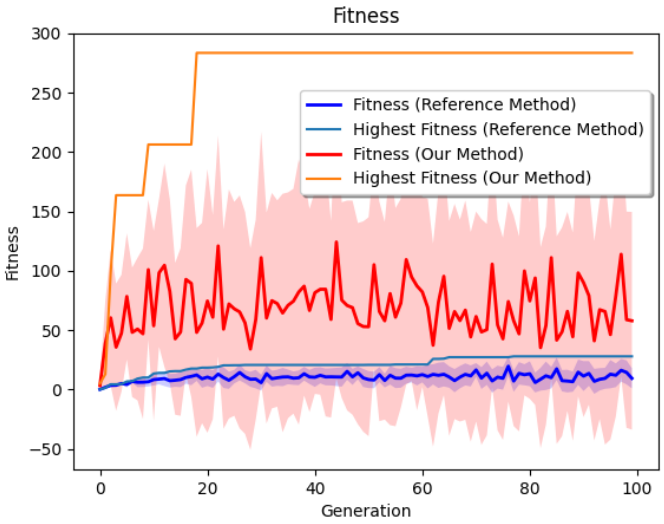

Fig. 4. The robot fitness of our method compared to the fitness of [18]. Our method produces higher fitness values.

It can be seen that our method performs significantly better than the method used by [18]. Our controllers converge to an optimal controller within 20 generations, while the existing method takes over 40 generations to converge to a solution. The average fitness and the maximum fitness of our controller is significantly higher than that of [18]. Our method is capable of producing gaits that can remain stable for longer periods of time which is evident from the higher fitness values.

B. Distance

The distance plot of our method is given in Fig. 5.

The distance covered by our robot is higher than that of [18]. Similar to the fitness plot, our method converges to an optimal controller with fewer generations than the existing method. Similarly, the maximum and average distance traveled by the robot in our proposed method is much higher which shows the robot is able to produce stable gaits using our method.

It can be seen from Fig. 4 and Fig. 5 that [18] produces results with lower standard deviation while our method produces results that have a much higher standard deviation. This shows that there is a significant amount of variation between

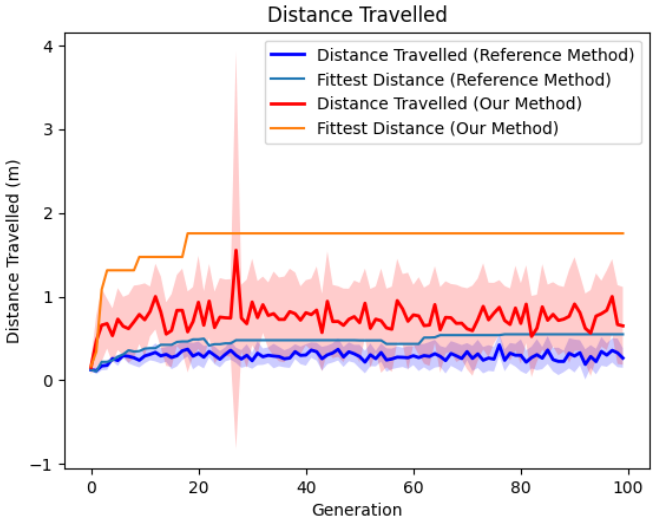

Fig. 5. The distance covered by our robot in the simulation environment is much higher than the distance covered by [18].

the individuals within a single generation in our method, while in the method used by [18] the variation is significantly less. The reason for such high standard deviation in our method is that the GA mutation produces controllers that are very different from one another, this can be fixed by lowering the mutation rate. The variation in our method makes it possible for the GA to converge to an optimal result quickly while at the same time the variation means that the system is prone to small changes in the controller weights.

C. Joint Movement

The joint movement of the fittest robot after 100 generations is given in Fig. 6.

Fig. 6 shows that the motion produced by our proposed method is much smoother this is because our method uses trapezoidal velocity profile. Furthermore, since our method calls the controller only after all the joints have reached their target position, this means that our method calls the controller more times which is why our robot shows more joint position changes than [18]. However, each joint position change in our method is much smoother.

The sharp peaks in the method of [18] means that the robot will abruptly change its joint position without slowing down first. This puts strain on the robot joints. While in our method the joints accelerate and then cruise at a constant velocity and then decelerate until they come to rest. The smooth joint motion helps the robot in moving from one position to another with a smooth motion which helps in reducing jerky movement, hence making the robot gait more stable compared to [18].

VI. Conclusion and Future Work

Gait generation for legged robots is a difficult task because it involves the optimization of a large number of parameters.

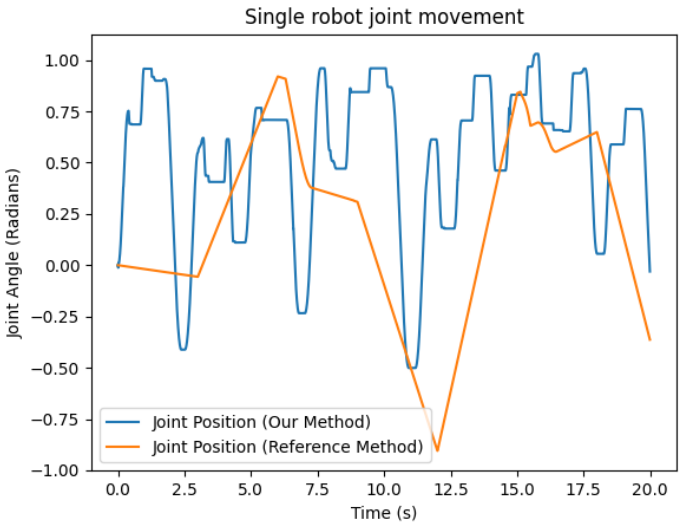

Fig. 6. Joint movement of one of the robot joints form our method compared to [18].

The optimization can be done with a number of techniques, however, most of them either take too long to converge or they produce unstable gaits. Therefore, in this paper we use GA to optimize an ANN controller for a quadrupedal robot. The robot is simulated in ODE and the controllers are tested on the robot, the best performing controllers are selected based on our selection criteria. Moreover, we have applied trapezoidal velocity profile to the joint movement of our robot in simulation. In our results the robot achieved a higher fitness score, traveled greater distance and also produced motion that was much smoother than the previous results. The robot in our proposed method is able to traverse it's environment with a stable gait and does not fall over quickly. Moreover, our method also converged to an optimal solution in lesser number of generations than in a previous method.

Our research focused on simulating the gaits, in the future our evolved gaits could be implemented on a hardware robot to test how well the gaits can be transferred to the real world.

REFERENCES

[1] J. Degrave, K. Caluwaerts, J. Dambre, and F. Wyffels, "Developing an embodied gait on a compliant quadrupedal robot," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2015, pp. 4486–4491. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7354014

[2] K. Caluwaerts, M. D'Haene, D. Verstraeten, and B. Schrauwen, "Locomotion Without a Brain: Physical Reservoir Computing in Tensegrity Structures," *Artificial Life*, vol. 19, no. 1, pp. 35–66, Jan. 2013. [Online]. Available: https://doi.org/10.1162/ARTL_a_00080

[3] K. Caluwaerts, J. Despraz, A. Işçen, A. P. Sabelhaus, J. Bruce, B. Schrauwen, and V. SunSpiral, "Design and control of compliant tensegrity robots through simulation and hardware validation," *Journal of the Royal Society Interface*, vol. 11, no. 98, Oct. 2014. [Online]. Available: https://royalsocietypublishing.org/doi/full/10.1098/rsif.2014.0520

[4] W.-L. Ma and A. D. Ames, "From bipedal walking to quadrupedal locomotion: Full-body dynamics decomposition for rapid gait generation," in *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2020, pp. 4491–4497. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9196841

[5] M. Fevre, B. Goodwine, and J. P. Schmiedeler, "Design and experimental validation of a velocity decomposition-based controller for underactuated planar bipeds," *IEEE Trans. Robot. Autom.*, vol. 3, no. 3, pp. 1896–1903, Jul. 2018. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8291493

[6] P. Eckert, A. E. Schmerbauch, T. Horvat, K. Söhnel, M. S. Fischer, H. Witte, and A. J. Ijspeert, "Towards rich motion skills with the lightweight quadruped robot serval," *Adaptive Behavior*, vol. 28, no. 3, pp. 129–150, 2020. [Online]. Available: https://doi.org/10.1177/1059712319853227

[7] T. Li, L. Zhou, Y. Li, H. Chai, and K. Yang, "An energy efficient motion controller based on SLCP for the electrically actuated quadruped robot," *Journal of Bionic Engineering*, vol. 17, no. 2, pp. 290–302, 2020. [Online]. Available: https://doi.org/10.1007/s42235-020-0023-6

[8] J. Kim, D. X. Ba, H. Yeom, and J. Bae, "Gait optimization of a quadruped robot using evolutionary computation," *Journal of Bionic Engineering*, vol. 18, no. 2, pp. 306–318, 2021. [Online]. Available: https://doi.org/10.1007/s42235-021-0026-y

[9] R. Sarwar, A. Zia, R. Nawaz, A. Fayoumi, N. R. Aljohani, and S.-U. Hassan, "Webometrics: evolution of social media presence of universities," *Scientometrics*, vol. 126, no. 2, pp. 951–967, 2021. [Online]. Available: https://doi.org/10.1007/s11192-020-03804-y

[10] X. Zhu, M. Wang, X. Ruan, L. Chen, T. Ji, and X. Liu, "Adaptive motion skill learning of quadruped robot on slopes based on augmented random search algorithm," *Electronics*, vol. 11, no. 6, 2022. [Online]. Available: https://www.mdpi.com/2079-9292/11/6/842

[11] S. Mohammad, M. U. Khan, M. Ali, L. Liu, M. Shardlow, and R. Nawaz, "Bot detection using a single post on social media," in *2019 Third World Conference on Smart Trends in Systems Security and Sustainablity (WorldS4)*. IEEE, 2019, pp. 215–220. [Online]. Available: https://doi.org/10.1109/WorldS4.2019.8903989

[12] J. Zhang, F. Gao, X. Han, X. Chen, and X. Han, "Trot gait design and CPG method for a quadruped robot," *Journal of Bionic Engineering*, vol. 11, no. 1, pp. 18–25, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1672652914600160

[13] J. Yosinski, J. Clune, D. Hidalgo, S. Nguyen, J. C. Zagal, and H. Lipson, "Evolving robot gaits in hardware: the HyperNEAT generative encoding vs. parameter optimization," in *Proc. of the 20th European Conf. on Artificial Life*, 2011, pp. 890–897.

[14] K. Glette, G. Klaus, J. C. Zagal, J. Torresen *et al.*, "Evolution of locomotion in a simulated quadruped robot and transferral to reality," in *Proc. of the 7th Int. Symp. on Artificial Life and Robotics*, 2012, pp. 1–4.

[15] H. Suzuki, H. Nishi, A. Aburadani, and S. Inoue, "Animal gait generation for quadrupedal robot," in *2nd Int. Conf. on Innovative Computing, Information and Control (ICICIC)*, 2007, p. 20. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/4427667

[16] D. Golubovic and H. Hu, "GA-based gait generation of sony quadruped robots," in *Proc. 3rd IASTED Int. Conf. on Artificial Intelligence and Applications (AIA)*, 2003.

[17] S. N. Miandoab, F. Kiani, and E. Uslu, "Generation of automatic six-legged walking behavior using genetic algorithms," in *Int. Conf. on INnovations in Intelligent SysTems and Applications (INISTA)*, 2020, pp. 1–5. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9194620

[18] U. Mir, Z. Khan, U. I. Mir, F. Naseer, and W. Shah, "Evolution of locomotion gaits for quadrupedal robots and reality gap characterization," in *Int. Conf. on Theory and Practice of Natural Computing*. Springer, 2019, pp. 197–207.

[19] R. Smith, "Open dynamics engine," 2005. [Online]. Available: https://www.ode.org/