Applying Genetic Algorithms to Control Gait of Physically Based Simulated Robots

Milton Roberto Heinen and Fernando Santos Osório, *Member, IEEE*

Abstract— **This paper describes our studies in the legged robots research area and the development of the LegGen System, that is used to automatically create and control stable gaits for legged robots into a physically based simulation environment. The parameters used to control the robot are optimized using genetic algorithms (GA). Comparisons between different robot models and fitness functions were accomplished, indicating how to compose a better multi-criterion fitness function to be used in the gait control of legged robots. The best gait control solution and the best robot model were selected in order to help us to build a real robot. The results also showed that it is possible to generate stable gaits using GA in an efficient manner.**

I. Introduction

The autonomous mobile robots have been attracting the attention of a great number of researchers, mainly due to the challenge that this new research domain proposes: make these systems capable of intelligent reasoning and able to interact with the environment they are inserted in, through sensor's perception (infrared, sonar, bumpers, gyroscopes, etc) and motor's action planning and execution[1]. The mobile robots are applied in different important tasks like: bomb disarming, exploration of hostile environments and automatic vehicle conduction[2], [3]. All those systems have the capacity to read sensors data and use this information, in a semi or completely autonomous way, to generate commands moving the mobile robot or vehicle in a safe way. They should not to collide against environment obstacles or to risk its own integrity or the integrity of the elements present in the environment.

At the present time, the most part of mobile robots use wheels for locomotion, what does this task easy to control and efficient in terms of energy consumption, but they have some disadvantages since they have problems to move across irregular surfaces and to cross borders and edges. Thus, these robots are not the most indicated to move into environments specifically designed for humans, because they have several irregularities like staircases and uneven ground[4]. So, in order to make mobile robots better adapted to human environments, they must be able to walk and/or to have a similar locomotion mechanism used by humans and animals, that is, they should have a legged locomotion mechanism[5].

The development of legged robots is a quite difficult task, that needs the configuration of many gait parameters[6]. The manual configuration of these parameters demands a lot of effort and spent time of a human specialist, and the obtained results are usually suboptimal and specific for one robot architecture[7]. Thus, it is interesting to generate the robot gait configuration in an automatic manner, using machine learning techniques[8] to perform this task.

Milton Roberto Heinen and Fernando Santos Osório are with the Applied Computing (PIPCA), Universidade do Vale do Rio dos Sinos (UNISINOS), São Leopoldo, CEP 93022-000, Brazil (email: mheinen@turing.unisinos.br, fosorio@unisinos.br).

One of these machine learning techniques that are most adapted for this specific task are the genetic algorithms (GA)[9], [10]. This is a reasonable choice because according to the Evolution's Theory[11], the locomotion mechanisms of several life forms resulted of the natural evolution. This makes the use of genetic algorithms a natural solution, since they are biologically inspired. From the computational point of view, the genetic algorithms are also very well adapted for the automatic gait configuration of legged robots, because: (a) they use a multi-criterion optimization method to search solutions in the configuration space, that means in our specific case, they are capable to optimize not only the gait velocity, but also the stability and even other gait parameters; (b) they do not need local information for the error minimization, nor the gradient calculation, that is very important for the gait parameters generation and optimization, since it is very difficult to have available some a priori training data for supervised learning; (c) if correctly used, the genetic algorithms are capable to avoid local minima[8].

The main goal of this paper is to describe the LegGen System. This system is capable to automatically evolve the gait control of physically based simulated legged robots using genetic algorithms. This paper is structured as follows: The Section II describes related works in control of locomotion in legged robots; The Section III describes the genetic algorithms and the GAlib software library adopted in our system; The Section IV describes several concepts relative to mobile robots simulation; The Section V describes the proposed system, called LegGen; The Section VI describes the accomplished experiments and the obtained results; and the Section VII provides some final conclusions and future perspectives of this work.

II. Related Works

Control of locomotion in legged robots is a challenging multidimensional control problem[5], [1]. It requires the specification and coordination of motions in all robots' legs while considering factors such as stability and surface friction[12]. This is a research area which has obvious ties with the control of animal locomotion, and it is a suitable task to use to explore this issue[13]. It has been a research area for a considerable period of time, from the first truly independent legged robots like the Phony Pony built by

Frank and McGhee[14], where each joint was controlled by a simple finite state machine, to the very successful algorithmic control of bipeds and quadrupeds by Raibert[15].

Lewis[16] evolved controllers for a hexapod robot, where the controller was evaluated on a robot which learn to walk inspired on insect-like gaits. After a staged evolution, its behavior was shaped toward the final goal of walking. Bongard[17] evolved the parameters of a dynamic neural network to control various types of simulated robots. Busch[18] used genetic programming to evolve the control parameters of several robot types. Jacob[19], on the other hand, used reinforcement learning to control a simulated tetrapod robot. Reeve[13] evolved the parameters of various neural network models using genetic algorithms. The neural networks were used for the gait control of tetrapod robots.

In the most part of these approaches described above, the fitness function used was the distance traveled by the robot in a predefined amount of time. Although this fitness function is largely used, it may hinder the evolution of more stable gaits[20]. In our approach, we use in the fitness function, beyond distance traveled, sensorial information (gyroscope and bumpers) to guarantee stable and fast gaits.

III. Genetic Algorithms

Genetic algorithms are optimization methods of stochastic space state search based on the Darwin's Natural Evolution Theory[11], that were proposed in the 60s by John Holland[21]. They work with a population of initial solutions, called chromosomes, which are evolved through several operations during a certain number of generations, usually reaching a well optimized solution, and preserving the best individuals according to a specific evaluation criterion. In order to accomplish this, in each generation the chromosomes are individually evaluated using a function that measures its performance, called fitness function[10]. The chromosomes with the best fitness values are selected to generate the next generation applying the crossover and mutation operations. Thus, each new generation tends to adapt and improve the quality of solutions, until we obtain a solution that satisfies a specific objective.

The genetic algorithms implementation used in our system is based on the GAlib[1] software library, developed by Matthew Wall of the Massachusetts Institute of Technology. GAlib was selected as it is one of the most complete, efficient and well known libraries for genetic algorithms simulation, and also it is a free open source library based on C++.

In the LegGen System, a simple genetic algorithm described by Goldberg in his book[9] was used, and a floating point type genome was adopted. In order to reduce the search space, alleles were used to limit generated values only to possible values for each parameter. The Algorithm 1 shows a summary of the genetic algorithm used to evolve the robot's gait parameters. This is a basic genetic algorithm with non-overlapping populations, elitism, roulette wheel selection schema and uniform crossover[9]. The fitness values were scaled using the sigma truncation scaling schema[10].

[1]GAlib – http://www.lancet.mit.edu/ga/

Algorithm 1 *Pseudocode of the GA*

```
$F \leftarrow$ FitnessFunction
$G \leftarrow$ NumberOfGenerations
$M \leftarrow$ SizeOfPopulation
$p_c \leftarrow$ FrequencyOfCrossover
$p_m \leftarrow$ FrequencyOfMutation
Population $\leftarrow$ RandomPopulation($M$)
CalculateFitness(Population)
while GenerationNum $< G$ do
    Parents $\leftarrow$ Population
    Children $\leftarrow$ Crossover(Parents, $p_c$)
    NewPopulation $\leftarrow$ Mutation(Children, $p_m$)
    Population $\leftarrow$ NewPopulation
    CalculateFitness(Population)
end while
return (BestIndividual)
```

IV. Mobile Robot Simulation

When someone wants to make experiments in the mobile robots research area, two alternatives are possible: (a) to accomplish the experiments directly in a real robot; or (b) to make experiments using a simulated robot. The use of a real robot has the advantage of to be realistic, but the simulation have the following advantages[22], [23]:

- When using simulated robots, does not exist the risk of robot damages;
- Tasks as the recharge of batteries are not necessary;
- The robot positioning in order to restart a simulation can be accomplished without human intervention;
- The simulation clock can be accelerated, reducing the total amount of spended time for learning;
- Several different architectures and robot models can be tested before the construction of the robot.

For these reasons, we chose to implement our initial experiments using a simulated robot, because this makes possible to discover the most efficient robot architecture to be built in the future. Once the physical robot construction is finished, the control model learned using the simulated robot may be quickly adapted to the real robot, and only a few adjustments may be necessary in order to adapt the simulated model to the reality. Based on these main ideas, we chose to implement a very realistic robot simulator, using a physical simulation engine, so we can build simulated robots very similar to the real models.

A. Physics Simulation Engine

In order to do more realistic mobile robots simulation, several elements of the real world should be present in the simulated model, doing the simulated bodies to behave in a similar way related to the reality. Especially, it is necessary that the robot suffers from instability and falls down if badly positioned and controlled, and also it should interact and collide against the environment objects in a realistic manner. To accomplish that, it is necessary to model the physics laws in the simulation environment (e.g. gravity, inertia, friction, collision). Nowadays, several physics simulation tools exist used for the implementation of physics laws in simulations. After study different possibilities, we chose a widely adopted free open source physics simulation library, called Open Dynamics Engine - ODE[2].

ODE is a software library for the simulation of articulated rigid bodies dynamics. With this software library, it is possible to make autonomous mobile and legged robots simulations with great physical realism. In ODE, several rigid bodies can be created and connected through different types of joints. To move bodies using ODE, it is possible to apply forces or torques directly to the body, or it is possible to activate and control angular motors. An angular motor is a simulation element that can be connected to two articulated bodies, which have several control parameters like axis, angular velocity and maximum force. With these elements, it is possible to reproduce articulations present in real robots, humans or animals, with a high precision level.

V. Proposed System

The LegGen[3] system was developed to accomplish the gait control of simulated legged robots in an automatic way. It was implemented using the C++ programming language and the free software libraries ODE and GAlib. The LegGen System reads two configuration files, one describing the robot format and dimensions and the other file describing the simulation parameters.

In the LegGen System, the gait control is generated using a finite state machine (FSM), in which is defined for each state and for each robot joint their final expected angles configuration[17]. In this way, the controller needs to continually read the joints angle state, in order to check if the joint motor accomplished the task. Real robots do this using sensors (encoders) to control the actual angle attained by the joints[5], [1]. So, in this approach the gait control is accomplished in the following way: initially the controller verify if the joints have already reached the expected angles. The joints that do not have reached them are moved (activate motors), and when all the joints have reached their respective angles, the FSM passes to the following state. If some of the joints have not reached the specified angles after a certain limited time, the state is advanced independently of this. In a future version of the system, we are planning to treat this situation more carefully, because the leg can be blocked by an obstacle and the robot can be damaged in this case.

To synchronize the movements, it is important that all joints could reach their respective angles at almost the same time. This is possible with the application of a specific joint angular velocity for each joint, calculated by the equation:

$$V_{ij} = Vr_i(\alpha_{ij} - \alpha_{ij-1}) \qquad (1)$$

where V_{ij} is the velocity applied to the motor joint i in the j state, α_{ij} is the joint angle i in the j state, α_{ij-1} is the joint angle i in $j-1$ state, and Vr_i is the reference velocity of the i state, used to control the set velocity. The reference velocity Vr is one parameter of the gait control that is also optimized by the genetic algorithm. The other parameters are the joints angles for each state. To reduce the search space, the genetic algorithm only generates values between the maximum and minimum accepted values for each specific parameter.

The Table I shows the parameters used by the LegGen System, with the values used in the simulations (described below in Section VI). The *Crossover*, *Mutation*, *Population size* and *Number of generations* parameters are used directly by the GAlib software. The *Number of states* parameter is the number of FSM states, the *Time of walk* parameter is the time of each individual walk during the fitness evaluation, and the *Velocity min* and *Velocity max* are the relative velocity (Vr) interval generated by GA.

TABLE I
Parameters of the LegGen System

Par-ID	Parameter	Value
1	Crossover	0.60
2	Mutation	0.05
3	Population size	50
4	Number of generations	100
5	Number of states	4
6	Time of walk	60
7	Velocity min	0.0
8	Velocity max	1.0

The LegGen System works as follows: initially the file describing the robot is loaded, and the robot is created in the ODE environment according to file specifications. After this, the system parameters are loaded (Table I), and the genetic algorithm is initialized and executed until the number of generations is reached. The evaluation of each chromosome is realized in the following way:

- The robot is placed in the starting position and orientation in the simulation environment;
- The genome is read and the robot control FSM table values are set;
- The physical simulation is executed during a predefined amount of time (sixty seconds in our experiments);
- Gait information and sensor data are captured during each individual physical simulation;
- Fitness is calculated and returned to the GAlib;

During the simulation, if all paws of the robot leave the ground at same time for more than one second, the simulation of this individual is immediately stopped, because this robot probably fell down, and therefore it is not necessary to continue the physical simulation of this individual.

A. Modeled Robots

According to the documentation, computational complexity when using the ODE library is $O(n^2)$, where n is the amount of bodies present in the simulated physical world. Thus, in order to maintain the simulation speed in

[2]ODE – http://www.ode.org

[3]LegGen – http://www.inf.unisinos.br/˜osorio/leggen

an acceptable rate, we should use few and simple objects. For this reason, all the simulated robots were modeled with simple objects, as rectangles and cylinders, and they have only the necessary articulations to perform the gait. Thus, body parts as the head and the tail are not present in the modeled robots. In order to keep our robot project simple, the joints used in the robots legs just move around the z axis of the robot (the same axis of our knees), and the simulations just used robots walking in a straight line. In the near future, we plan to extend our system to accept more complex robot models and joints.

Several robot types were developed and tested, before we defined the final two main models presented here, that are shown in Figure 1. The robot model, called **HexaL3J**, have six legs and three parts per leg. The paws are wider than the remaining legs, in way to give a large support to the robot.

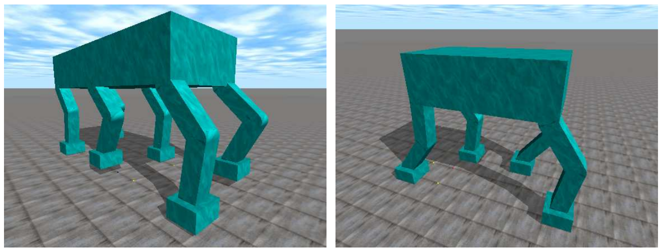

(a) HexaL3J (b) TetraL3J

Fig. 1. Robot models used in the simulations

This robot model differs from the majority of hexapod robots developed, because their legs are not like the insect legs, but their leg joints are like human joint legs. The Figure 1(b) model, called **TetraL3J**, is similar to the previous model, but it has just four legs. Both models in Figure 1(a) and 1(b) have the paws final joint angles automatically calculated using direct kinematics, in such a manner as these paws are always parallel to the ground. The Table II shows the dimensions of the robots in centimeters.

TABLE II
DIMENSIONS OF THE SIMULATED ROBOTS

	TetraL3J			**HexaL3J**		
Part	**x**	**y**	**z**	**x**	**y**	**z**
Body	45.0	15.0	25.0	80.0	15.0	30.0
Thigh	5.0	15.0	5.0	5.0	15.0	5.0
Shin	5.0	15.0	5.0	5.0	15.0	5.0
Paw	8.0	5.0	9.0	8.0	5.0	9.0

The simulated robots dimensions are approximately the dimensions of a medium sized dog. The joint restrictions used in the simulated robot are similar as they biological equivalents, with the following values: Hip=[-60°;15°]; Knee=[0°;120°]; Ankle=[-90°;30°]. All the robot legs have these same joint restrictions.

The use of paws as showed in Figures 1(a) and 1(b) models was designed to allow a more stable walk, mainly when dynamic stability was used. The robot joints have maximum and minimum joint angle limits similar to horses and dogs, but these animals have more articulated members than the implemented in our models.

B. Genome

The genome used in the LegGen system was a simple array of floating points numbers, with alleles limiting the generated values between the minimum and maximum joint limits. The vector width W is:

$$W = \frac{S \times L \times (J - 1)}{2} + S \tag{2}$$

where S is the number of states, J is the number of joints per leg and L is the number of robot legs. For the fitness evaluation, the genome is converted in a FSM control table. The Table III shows a example of a FSM control table.

TABLE III
EXAMPLE OF A GENOTYPE

	State 1	**State 2**	**State 3**	**State 4**
Reference velocity	1.1703	2.0705	1.2499	1.5901
Front legs - hip	0.0698	-0.6283	-0.1570	-0.0698
Front legs - knee	1.0473	0.7504	0.2268	0.4363
Front legs - ankle	-1.1172	-0.1221	-0.0698	-0.3665
Rear legs - hip	-0.1570	-0.0698	0.0698	-0.6283
Rear legs - knee	0.2268	0.4363	1.0473	0.7504
Rear legs - ankle	-0.0698	-0.3665	-1.1172	-0.1221

The first line shows the reference velocity (Vr) of each state, and the following lines shows the desired angles for the hip, knee and ankle joints of each leg in radians. In this example, were used four states in the FSM table. Due to the fact that the robots are symmetrical, just half of the joints are needed to evolve (for just one side of the robot). The joints of the other side are obtained inverting the angle values between the states 1 and 3 and between states 2 and 4, in this case.

C. Fitness Function

In this work, different fitness functions were studied and tested to evaluate their contribution in order to generate a better gait control. The fitness function F of the genetic algorithm used in the first set of experiments was based only in the distance covered by the robot (D) in the x axis:

$$F = D \tag{3}$$

$$D = x_1 - x_0 \tag{4}$$

where x_0 is the robot start position and x_1 is the final robot position in the x axis. Using this fitness function, the individuals that move forward will be rewarded, and the individuals that move backward will be punished, receiving a negative fitness.

We start believing that, with this fitness function, the individuals selected to produce offsprings should be the ones that have a more stable gait: a stable individual should to move longer than the one that fell down. But due to the fact that in the first generations almost all the individuals are unstable and fall down, the selected individuals were the ones that simply fell down in the forward direction. So, the selected individuals are not the individuals that can remain in the upright position and walk longer during the simulation. In this way, the Equation 3 did not lead us to a good solution, and thus the GA makes an almost random search.

To avoid this problem, we developed an other fitness function that use sensorial information in order to make the gait learning more efficient. One of the less expensive and simpler robotic sensors are the bumpers. These contact sensors can be installed under the robot paws, and they indicate when the paw is touching the ground. Thus, we decided to simulate bumpers under the robot paws, and then it was possible to determine how many paws are maintained in contact with the ground for each instant of time. The new fitness calculation was accomplished through the equation:

$$F = D * \mu_P \tag{5}$$

where μ_P is the average number of paws touching the ground, calculated through the equation:

$$\mu_P = \frac{\sum_{i=1}^{N} b_i}{N} \tag{6}$$

where b_i is the number of paws in contact with the ground in the instant i and N is the total number of sensorial (bumper) readings. Using this fitness function, we noticed that an odd behavior began to happen. The individuals that maintained all the paws in the ground and just inclined forward the front of their bodies were rewarded more than those that lifted the paws from the ground during the walk. Thus, other modifications were accomplished in the previous fitness function. This new fitness function is calculated in the following way:

$$F = \frac{D}{1 + B} \tag{7}$$

where B (Bumpers) is calculated through the equation:

$$B = \left(\mu_P - \frac{L}{2}\right)^2 \tag{8}$$

where L is the number of the robot legs. In this way, the individuals that maintain approximately half of the paws in contact with the ground, during walk simulation, will consider in the fitness computation the total distance covered by the robot. The individuals that fell down or did not move the paws from the ground will be punished, and the total distance covered by the robot will be reduced in function of this. This fitness function is useful in a *trot* gait.

Using the Equation 7, the gait learning became much more efficient, but we still have a lot of totally unstable solutions. The visualization of the obtained solutions showed unstable robots where the body height and inclination varies a lot during the walking simulation. Thus, besides the bumpers, we decided to add simulated inertial sensors (gyroscope). These sensors are used in some modern mobile robots and they are becoming a largely adopted device in walking machines. During the walk, readings from the simulated gyroscope are collected, and the robot instability measure G (Gyro) is calculated through the equation[20]:

$$G = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \overline{x}_x)^2 + \sum_{i=1}^{N}(y_i - \overline{x}_y)^2 + \sum_{i=1}^{N}(z_i - \overline{x}_z)^2}{N}} \tag{9}$$

where N is the number of sample readings, x_i, y_i and z_i are the data collected by the simulated gyroscope in the time i, and $\overline{x}_x$, $\overline{x}_y$ and $\overline{x}_z$ are the mean values of gyroscope readings, calculated by the equation:

$$\overline{x}_x = \frac{\sum_{i=1}^{N} x_i}{N}, \quad \overline{x}_y = \frac{\sum_{i=1}^{N} y_i}{N}, \quad \overline{x}_z = \frac{\sum_{i=1}^{N} z_i}{N} \tag{10}$$

The fitness F is then calculated through the equation:

$$F = \frac{D}{1 + G + B} \tag{11}$$

Analyzing the fitness function, we can observe that B reaches its smaller value when the robot maintains half of its endpoints (paws) touching the ground. This condition is desirable when the type of gait adopted is the *trot*. In this way, the best solutions have B close to zero. So, this parameter will have a strong influence in the population evaluation and evolution. Related to the other fitness parameters, the individual better qualified will be the one that has the best relationship between velocity and stability. The best solutions are those that moves fast, but without losing the stability[20]. After we included the instability measure G in the fitness function, we analyzed if it was possible to remove the average number of endpoints touching the ground from the fitness function. This new fitness function is represented in the following equation:

$$F = \frac{D}{1 + G} \tag{12}$$

In the next section we describe the accomplished experiments using these four different fitness functions (Equations 3, 7, 11 and 12), and using the robots with four and six paws. These experiments were accomplished in way to verify which of these fitness functions is more efficient to generate stable gaits for legged robots.

VI. Results

In order to determine the best robot model to build, several experiments were conducted using robots with four and six legs (TetraL3J and HexaL3J). The experiments also were conducted to discover the most suitable fitness function to be applied in the genetic algorithm, which obtains the best robot gait control. Each robot was tested using the four fitness functions described above (Equations 3, 7, 12, and 11).

Our intention is to create a real robot using the fitness function that better improves the gait control learning by the genetic algorithm, but with an accessible final cost. The Equations 12 and 11 need a gyroscope sensor to compute the fitness function, so this solution is more expensive. The

Equation 7 needs just bumper sensors, which have a quite accessible cost and can be easily used in a real robot. Thus, the use of gyroscope sensors would be justifiable only if they present a significant increase in the robot performance (statistically significant results) in terms of speed and stability. In relation to the total number of robot legs, the best choice is to use the smallest possible number of legs that allows a stable gait. The Table IV describes the eight different type of experiments.

TABLE IV

ACCOMPLISHED EXPERIMENTS

EXP	Fitness Function	Robot
01	$F = D$ (Equation 3)	HexaL3J
02	$F = D$ (Equation 3)	TetraL3J
03	$F = D\,/\,(1+B)$ (Equation 7)	HexaL3J
04	$F = D\,/\,(1+B)$ (Equation 7)	TetraL3J
05	$F = D\,/\,(1+G)$ (Equation 12)	HexaL3J
06	$F = D\,/\,(1+G)$ (Equation 12)	TetraL3J
07	$F = D\,/\,(1+G+B)$ (Equation 11)	HexaL3J
08	$F = D\,/\,(1+G+B)$ (Equation 11)	TetraL3J

Due to the stochastic nature of the genetic algorithms, each experiment described in Table IV was repeated 10 times using different random seeds, and the mean and standard deviation of the obtained results were calculated. The Table V shows the results obtained in the experiments.

TABLE V

RESULTS OBTAINED IN THE SIMULATIONS

	F		D		G		B	
EXP	μ	σ	μ	σ	μ	σ	μ	σ
01	191	103	1083	84	1.77	0.67	4.16	2.04
02	279	105	622	117	1.54	2.11	0.49	0.74
03	343	48	680	32	0.97	0.25	0.04	0.03
04	272	89	424	140	0.56	0.18	0.01	0.01
05	301	76	1017	69	0.82	0.18	1.77	0.84
06	278	48	490	72	0.68	0.19	0.12	0.18
07	436	84	689	102	0.53	0.19	0.07	0.05
08	268	70	405	68	0.56	0.30	0.01	0.01

The first column indicates the experiment identification (EXP identifies the same specific experiment in both tables), the second and third columns show, respectively, the mean and the standard deviation of the fitness (F), the fourth and the fifth columns show the mean and the standard deviation of the distance covered by the robot (D) in centimeters, the sixth and seventh columns show the mean and the standard deviation of the robot instability measure (G), and the last two columns show the mean and the standard deviation of the average number of endpoints touching the ground (B).

The Figure 2 shows the *boxplot* graphic of all experiments. From this fitness distributions, we can notice that there is no significant differences between the fitness values obtained in the experiments accomplished with the two robot types. Although the HexaL3J robot is a little bit faster, the difference is not enough to justify the use of a six legged robot.

A. HexaL3J Results

The Figure 3 shows the *boxplot* graphics of the experiments using the HexaL3J robot (Experiments 01, 03, 05 and 07). The Figure 3(a) shows the *boxplot* graphic of the distance covered by the robot (D) and the Figure 3(b) shows the *boxplot* graphic of the robot instability measure (G).

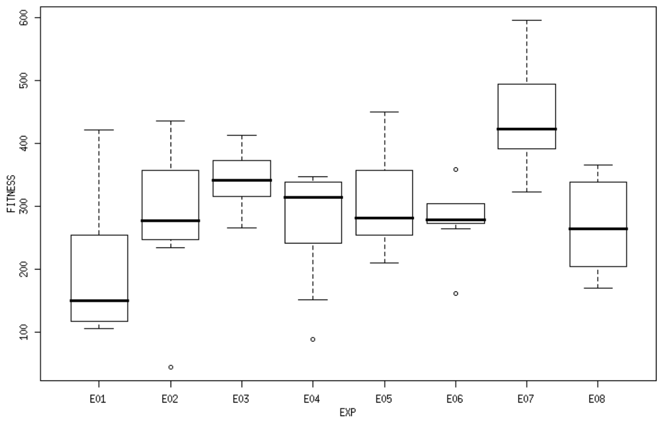
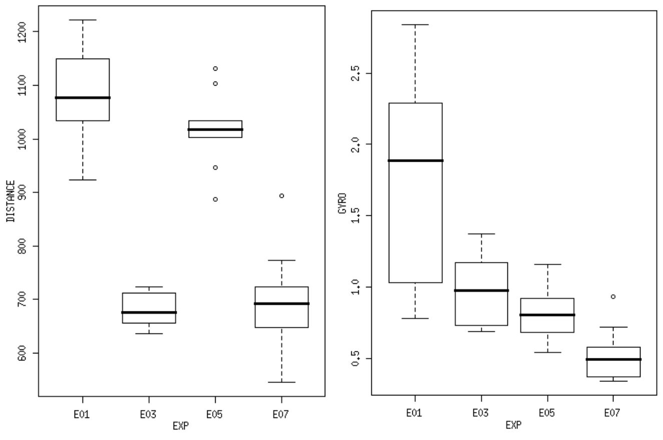


Fig. 2. Boxplot of the all experiments



(a) Distance (b) Instability

Fig. 3. Boxplot of the HexaL3J experiments

From the observed results presented in Table V and the fitness distributions of the Figure 3, we can reach several conclusions. Firstly, the use of bumper sensors (Experiments 03 and 07) resulted in a slower walking, although more stable. The use of a gyroscope sensor (Experiment 05) did not affect the robot velocity in a significant way, but it reduced the instability. The best results in terms of velocity/stability are the ones that used both types of sensors (Experiment 07). So, these results showed that the choice of the proposed multi-criterion fitness function improved the results obtained by the genetic algorithm. The Figure 4 shows snapshots of the generated gait obtained in the Experiment 01, and the Figure 5 shows the generated gait of the Experiment 07.

Analyzing the visual results, it can be noticed that the robot gait of the Figure 5 is much more stable than the gait of the robot showed in Figure 4.

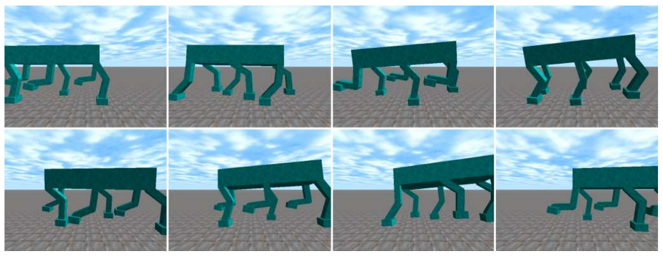


Fig. 4. Experiment 01 - HexaL3J gait

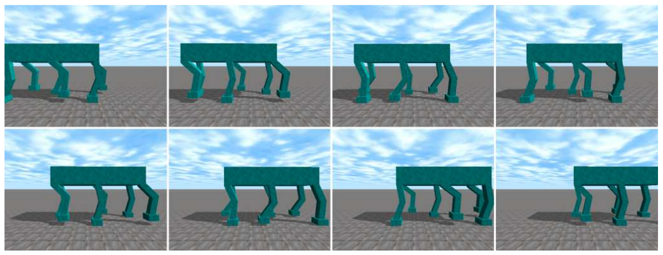
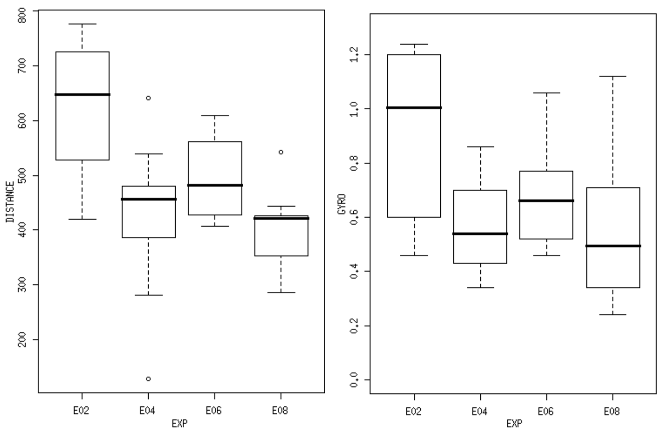


Fig. 5. Experiment 07 - HexaL3J gait

B. *TetraL3J Results*

The Figure 6 shows the *boxplot* graphics of the experiments using the TetraL3J robot (Experiments 02, 04, 06 and 08). The Figure 6(a) shows the *boxplot* graphic of the distance covered by the robot (D) and the Figure 6(b) shows the *boxplot* graphic of the robot instability measure (G).



(a) Distance

(b) Instability

Fig. 6. Boxplot of the TetraL3J experiments

From the observed results presented in Table V and the fitness distributions of the Figure 6, we can reach the conclusion that although the distance covered by the robots in the experiments 04 and 08 are a little bit smaller, the differences are not statistically significant. Considering the instability, the use of bumper sensors (Experiment 04) increased the stability more than the use of the gyroscope sensor (Experiment 06). The Figure 7 shows an example of the generated gait obtained in the Experiment 02, and the Figure 8 shows an example of generated gait obtained in the Experiment 08.

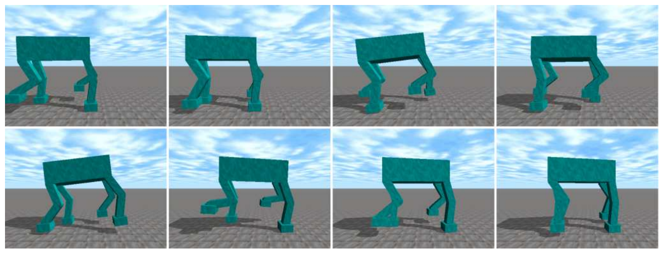


Fig. 7. Experiment 02 - TetraL3J gait

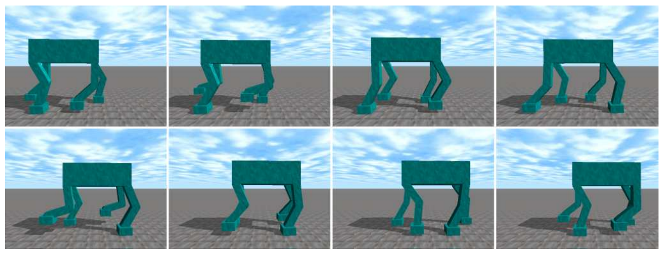


Fig. 8. Experiment 08 - TetraL3J gait

The Figure 9 shows an example of generated gait obtained in the Experiment 04. Although the use of the gyroscope has practically the same benefits of the bumpers in stability terms, the absence of it changes the overall robot behavior. We can see in the Experiment 04 that the robot doesn't maintain the same walking direction, because the only sensorial information that it receives are the covered distance in the x axis and the bumper readings. In this case there is no robot orientation control.

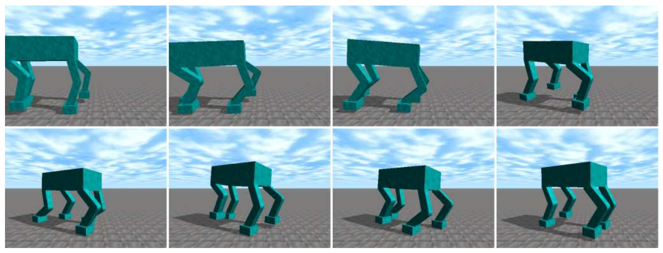


Fig. 9. Experiment 04 - TetraL3J gait

Analyzing the visual results, it can be noticed that the experiments using the Equation 11 (Experiment 08) are much more stable than others, as showed in Figure 8.

The Figure 10 show the evolution and the relationship between the fitness and the number of generations in a experiment using the TetraL3J. The bright points (not filled) show the best fitness values for each generation, and the dark points (filled) show the mean fitness values of the population for each generation. The Figure 11 shows the relationship between instability and velocity in 200 experiments accomplished using the TetraL3J robot and the Equation 11.

We observed in the Figure 11 that an interesting relationship between velocity and instability exists. This suggest

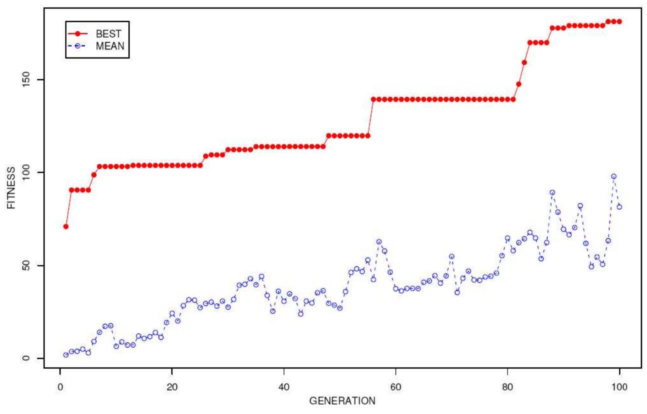


Fig. 10. Generations x fitness

that the high speed solutions are less stable, but is possible to obtain stable solutions with moderate speeds. The main goal of our optimization search GA algorithm was to obtain control solutions with the lowest possible instability (solution points close to the x axis) and with the greater possible distance coverage (solution points far from the y axis), in order to maximize the velocity and to minimize the instability. The Figure 11 shows that we achieved that goal.

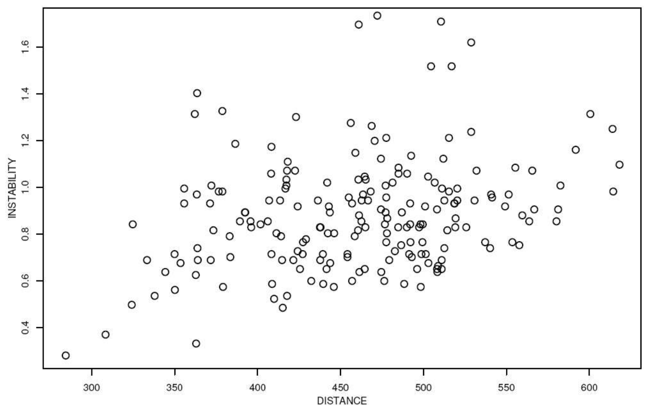


Fig. 11. Velocity x instability

VII. Conclusions and Perspectives

Based on the performed experiments, we observed that six legged robots are able to move faster than four legged robots, like it occurs in the nature with some arthropods and insects, that are very fast animals if we consider the covered distances related to their small size. The number of legs seems also to play and important role related to their movement skills. In four legged robots, we observed through our simulations that fitness functions with sensorial information are very useful to generate stable gaits, and we also concluded that although these robot models are slower than six legged robots, they are also able to develop stable gaits.

We concluded that both robot models are possible configurations to be adopted in a physical construction of a real robot, but the four legged robot is less expensive than the six leg robot. We also achieved a good performance with our control gait system implementation, which can provide a stable gait control to legged robots.

The perspectives of this work includes to adapt gait control in order to make possible control robots moving over irregular surfaces and to climb or to descend stairs, as well as, this work will help us in the physical robot construction based on the specifications of our best learned models. The real robot implementation created from the virtual model will help us to validate the control system in real conditions.

References

[1] G. A. Bekey, *Autonomous Robots: From Biological Inspiration to Implementation and Control*. Cambridge, MA: MIT Press, 2005.

[2] F. J. Heinen and F. S. Osório, "HyCAR - a robust hybrid control architecture for autonomous robots," in *Proc. Hybrid Intelligent Systems (HIS)*, vol. 87. Santiago, Chile: IOS Press, 2002, pp. 830–840.

[3] C. Kelber, C. R. Jung, F. S. Osório, and F. J. Heinen, "Electrical drives in intelligent vehicles: Basis for active driver assistance systems," in *Proc. IEEE Int. Symposium on Industrial Electronics (ISIE)*, vol. 4, Dubrovnik, Croatia, 2005, pp. 1623–1628.

[4] R. Knight and U. Nehmzow, "Walking robots - a survey and a research proposal," Univ. Essex, Essex, UK, Technical Report CSM-375, 2002.

[5] G. Dudek and M. Jenkin, *Computational Principles of Mobile Robotics*. Cambridge, UK: Cambridge Univ. Press, 2000.

[6] G. Wyeth, D. Kee, and T. F. Yik, "Evolving a locus based gait for a humanoid robot," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, vol. 2, Las Vegas, NV, Oct. 2003, pp. 1638–1643.

[7] S. Chernova and M. Veloso, "An evolutionary approach to gait learning for four-legged robots," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, Sendai, Japan, Sept. 2004.

[8] T. Mitchell, *Machine Learning*. New York: McGrall-Hill, 1997.

[9] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[10] M. Mitchell, *An Introduction to Genetic Algorithms*. Cambridge, MA: MIT Press, 1996.

[11] C. Darwin, *Origin of Species*. London, UK: John Murray, 1859.

[12] N. Kohl and P. Stone, "Policy gradient reinforcement learning for fast quadrupedal locomotion," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, New Orleans, LA, Apr. 2004, pp. 2619–2624.

[13] R. Reeve and J. Hallam, "An analysis of neural models for walking control," *IEEE Trans. Neural Networks*, vol. 16, no. 3, pp. 733–742, May 2005.

[14] R. B. McGhee, "Robot locomotion," *Neural Control of Locomotion*, pp. 237–264, 1976.

[15] M. H. Raibert, *Legged Robots That Balance*. Cambridge, MA: MIT Press, 1986.

[16] M. A. Lewis, A. H. Fagg, and A. Solidum, "Genetic programming approach to the construction of a neural network for control of a walking robot," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, Nice, France, 1992, pp. 2618–2623.

[17] J. C. Bongard and R. Pfeifer, "A method for isolating morphological effects on evolved behaviour," in *Proc. 7th Int. Conf. Simulation of Adaptive Behaviour (SAB)*. Edinburgh, UK: MIT Press, Aug. 2002, pp. 305–311.

[18] J. Busch, J. Ziegler, C. Aue, A. Ross, D. Sawitzki, and W. Banzhaf, "Automatic generation of control programs for walking robots using genetic programming," in *Proc. European Conf. Genetic Programming (EuroGP)*, Berlin, Germany, 2002, pp. 258–267.

[19] D. Jacob, D. Polani, and C. L. Nehaniv, "Legs than can walk: Embodiment-based modular reinforcement learning applied," in *Proc. IEEE Int. Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, Espoo, Finland, June 2005, pp. 365–372.

[20] D. Golubovic and H. Hu, "Ga-based gait generation of sony quadruped robots," in *Proc. 3th IASTED Int. Conf. Artificial Intelligence and Applications (AIA)*, Benalmadena, Spain, Sept. 2003.

[21] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: Univ. Michigan Press, 1975.

[22] K. Wolff and P. Nordin, "Evolutionary learning from first principles of biped walking on a simulated humanoid robot," in *Proc. Advanced Simulation Technologies Conf. (ASTC)*, Orlando, FL, Apr. 2003.

[23] A. M. Law and D. W. Kelton, *Simulation Modeling and Analysis*. New York: McGraw-Hill, 2000.