Evolving Gait Control of Physically Based Simulated Robots

Milton Roberto Heinen [1]
Fernando Santos Osório [2]

Abstract: This paper describes the LegGen System, used to automatically create and control stable gaits for legged robots into a physically based simulation environment. In our approach, the gait is defined using three different methods: a finite state machine based on robot's leg joint angles sequences; a locus based gait defining the endpoint (paw) trajectory and using inverse kinematics to determine joint angles; and through a half ellipse cyclic function used to define each endpoint trajectory. The parameters used to control the robot through these methods are optimized using Genetic Algorithms. The model validation was performed by several experiments realized with different configurations of four and six legged robots, simulated using the ODE physical simulation engine. A comparison between these different robot configurations and control methods was realized, and the best solution was selected in order to help us to build a physical legged robot. The results also showed that it is possible to generate stable gaits using Genetic Algorithms in a efficient manner, using these three different methods.

1 Introduction

The autonomous mobile robots has been attracting the attention of a great number of researchers, 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 [10, 17]. 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 [18]. So, in order to make mobile robots better adapted to human environments and to irregular surfaces, they must be able to walk and/or to have a similar locomotion mechanism used by the humans and animals, that is, they should have a legged locomotion mechanism [1].

However, the development of legged robots capable to move in irregular surfaces is a quite difficult task, that needs the configuration of many gait parameters [28]. The manual configuration of these parameters demands a lot of effort and spent time of a human specialist,

[1]Informatics Institute, UFRGS, CEP 91501-970
{mrheinen@inf.ufrgs.br}
[2]Applied Computing, UNISINOS, CEP 93022-000
{fosorio@unisinos.br}

and the obtained results are usually suboptimal and specific for one robot architecture [5]. Thus, it is interesting to generate the robot gait configuration in an automatic manner, using Machine Learning techniques [23] to perform this task.

One of these Machine Learning techniques that are most adapted for this specific task are the Genetic Algorithms (GA) [8, 22]. This is a reasonable choice because according to the Evolution's Theory [6], the locomotion mechanisms of life forms resulted from the natural evolution, what makes the use of Genetic Algorithms a natural solution since they are biologically inspired and can generate biologically plausible solutions. 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 don't need local information for the error minimization, nor the gradient calculation, what 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 [22].

The main goal of this paper is to describe the LegGen System [12, 13, 11]. 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 3 describes the Genetic Algorithms and the GAlib software library adopted in our system; The Section 4 the use of a physical simulation engine, the Section 5 describes some possible alternatives to legged robot configuration; The Section 6 describes the LegGen system, and the robots used in the simulations; The Section 7 describes the accomplished experiments and the obtained results; and the Section 8 provides some final conclusions and future perspectives.

2 Related Works

Control of locomotion in legged robots is a challenging multidimensional control problem [7, 1]. It requires the specification and coordination of motions in all robots' legs while considering factors such as stability and surface friction [19]. 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 [27]. 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 [21], where each joint was controlled by a simple finite state machine, to the very successful algorithmic control of bipeds and quadrupeds by Raibert [26].

Lewis [20] 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 [2] evolved the parameters of a dynamic neural network to control various types of simulated robots. Busch [4] used genetic programming to evolve the control parameters of several robot types. Jacob [16], on the other hand, used reinforcement learning to control a simulated tetrapod robot. Reeve [27] 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 [9]. In our approach, we use in the fitness function, beyond distance traveled, sensorial information (gyroscope and bumpers) to guarantee stable and fast gaits.

3 Genetic Algorithms

Genetic Algorithms are optimization methods of stochastic space state search based on the Darwin's Natural Evolution Theory [6], that are proposed in the 60s by John Holland [15]. 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 [22]. 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 was based on the GAlib software library[3], developed by Matthew Wall of Massachusetts Institute of Technology (MIT). 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 and open source C++ library.

In the LegGen System, a Genetic Algorithm as described by Goldberg in his book [8] 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.

4 Mobile Robot Simulation

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

[3]the GAlib is available for download in the site `http://www.lancet.mit.edu/ga/`

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 [24]. 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[4]. ODE is a software library for the simulation of articulated rigid bodies dynamics. With this software library, it's 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's 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's possible to reproduce articulations present in real robots, humans or animals, with a high precision level [24].

5 Gait Generation

In legged robots, the gait control can be generated in several different ways. One of the most simple ways to control the robot joints is using a Finite State Machine (FSM), in which is defined for each state the duration of the state and the activation level (force/velocity) for each joint. The Table 1 illustrates an example of a FSM table used to control an articulated mobile robot. Since the most of the robots are symmetrical (and great part of living forms too), only one half of the robot's states need to be learned - the other half is obtained by switching the left legs values with right legs values.

Table 1. Example of a FSM table

	State 1	**State 2**	**State 3**	**State 4**
Duration	0.05	0.10	0.55	0.80
Joint A	1.25	2.45	3.00	0.95
Joint B	2.70	0.15	0.70	1.95
Joint C	1.15	1.65	0.30	2.70

The main disadvantage of this approach described above is that the robot control is accomplished without any feedback from the external world, that is, no sensor data is used by the robot controller. But in a real robot, small differences in the actuators behavior, in the

[4]A library ODE is free and available for download in the site `http://openode.sourcefourge.net`

battery level charge, the friction in the joints or external obstacles collision may change the effective velocity and response of the joints, and thus the robot may not work as expected.

An alternative approach was developed in this work, using a FSM table similar to the above described method, but instead of determining the duration and the velocity of each state, it is determined for each state and for each robot joint their final expected angles configuration. 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 [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 can reach their respective angles at almost the same time. This is possible with the application of a specific joint angular velocity adjust value for each joint, calculated by the equation:

$$V_{ij} = k_i(\alpha_{ij} - \alpha_{ij-1}) \tag{1}$$

where V_{ij} is the velocity adjust 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 k_i is a constant of the i state, used to control the set velocity. The k is a 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 second possible approach we implemented uses a “locus based gait” to control the robot joints [28]. In this approach, instead of using a FSM to determine velocity or angles of each joint, the positions of each endpoint (“foot” or “paw”) are determined and specified by spatial coordinates in the x, y and z axis. In order to generate the gait, the controller needs to calculate the inverse kinematics and then use the calculated joints angles to move the robot. The advantage of this technique is that the search space can be reduced, even if the legs are composed by several segments. The main disadvantage is that we need a fast and efficient way to calculate the inverse kinematics. This also requires some specific knowledge about the robot structure and about the direct kinematics model implemented. In our implementation, the direct kinematics was calculated using the R Statistical Software[5] and the inverse kinematics was calculated in real time using the Powell’s method [3, 25].

[5]The R Software is a free statistical software available for download in the site `http://www.r-project.org/`

Another way we used to control the robot gait was to model the trajectory of endpoints using a cyclical function, as for example a half ellipse. The Figure 1, reproduced from [9], illustrates this situation.

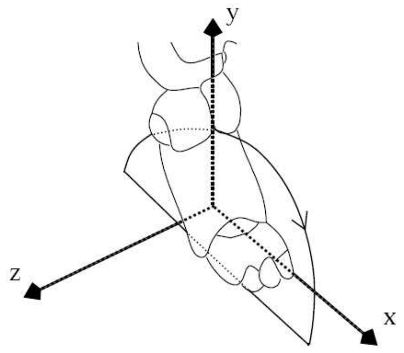


Figure 1. Example of a half ellipse trajectory [9]

With the trajectory defined by a half ellipse, the only parameters that need to be optimized by the Genetic Algorithm are the position of each leg's ellipse center (x, y and z axis coordinates), the height and width of each ellipse, the total time cycle duration and the specific time duration of the cycle during which the leg is touching the ground. The advantage of this approach is that theoretically it reduces the search space, simplifying the learning task. The main disadvantages are: (a) the half ellipse restricts a lot the possible robot movements, hindering gait in irregular surfaces; (b) the inverse kinematics also needs to be calculated in a fast and efficient way.

6 Proposed System

The LegGen System was developed to accomplish the gait control of simulated legged robots in an automatic way [12, 13, 11, 14, 14]. 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. Table 2 shows the parameters used by the LegGen System, with the values used in the simulations (described below in Section 7). 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 k interval generated by the Genetic Algorithm.

Table 2. Parameters of the LegGen System

Par-ID	Parameter	Value
1	Uniform 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 2), 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;
- The genome is read and the robot control FSM table values are set;
- The physical simulation is executed during a predefined time;
- Gait information and sensor data are captured during each physical simulation;
- Fitness is calculated and returned to the GAlib.

The way as the FSM table values are set by the genome depends of the gait generation method used. The LegGen System implements three methods for the gait generation: (a) FMS angles table; (b) locus based gait; (c) half ellipse controller. The fitness evaluation uses the following sensorial information that must be calculated: (a) the distance D covered by the robot; (b) instability measure G; and (c) average number of endpoints touching the ground B. The covered distance D is given by the equation:

$$D = Px_1 - Px_0 \tag{2}$$

where D is the distance traveled by the robot in the x axis (forward walk following a straight line), Px_0 is the robot start position and Px_1 is the final robot position in the x axis.

The instability measure is calculated using the robot position variations in the x, y and z axis. These variations are collected during the physical simulation, simulating a gyroscope/accelerometer sensor, which is a sensor present in some modern robots [7]. The instability measure G (Gyro) is then calculated by the following equation [9]:

$$G = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \overline{x}_x)^2 + \sum_{i=1}^{N}(y_i - \overline{x}_y)^2 + \sum_{i=1}^{N}(z_i - \overline{x}_z)^2}{N}} \tag{3}$$

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 gyroscope reading means, calculated by the equation:

$$\overline{x}_x = \frac{\sum_{i=1}^{N} x_i}{N}, \quad \overline{x}_y = \frac{\sum_{i=1}^{N} y_i}{N}, \quad \overline{x}_z = \frac{\sum_{i=1}^{N} z_i}{N} \tag{4}$$

To obtain the average number of endpoints touching the ground, we simulate touch sensors that imitate the operation of bumpers [7] placed under each paw of the robot. During the simulation, the number of endpoints touching the ground is collected, and at the end of each physical simulation the average number B is calculated using the equation:

$$B = \frac{\sum_{i=1}^{N} b_i}{N} \tag{5}$$

where N is the number of samples collected and b_i is the number of endpoints touching the ground at the time i. After finished the sensorial information processing, the fitness function F is then calculated through the equation:

$$F = \frac{D}{1 + G + (B - L/2)^2} \tag{6}$$

where L is the number of robot legs. Analyzing the fitness function, we see that B reaches its best value when the robot maintains half of its endpoints touching the ground, what is desirable when the gait used is the *trot*. In this way, the best solutions have $(B - L/2)^2$ close to zero, so this parameter will have a strong influence in the population evaluation and evolution. Related to the other parameters, the individual better qualified will be the one that has the best relationship between velocity and stability, so the best solutions are those that moves fast, but without losing the stability [9].

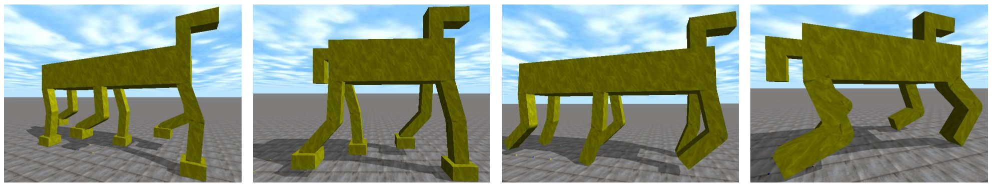
During the simulation, if all paws of the robot leave the ground at same time for more than one second, the simulation of this individual is immediately stopped, because this robot probably fell down, and therefore it is not necessary to continue the physical simulation until the predefined end time.

6.1 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 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. 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 four main models presented here, that are shown in Figure 2. The Figure 2(a) model, called **HexaL3J**,



(a) (b) (c) (d)

Figure 2. Robot models used in the simulations

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. The Figure 2(b) model, called **TetraL3J**, is similar to the previous model, but it has just four legs. Both models in Figure 2(a) and 2(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 model of Figure 2(c), called **HexaL2J**, is similar to the Figure 2(a) model, but it has just two articulations per leg, in other words, it doesn't have paws. Thus, in this model all the joint angles are calculated by the Genetic Algorithm, without using direct kinematics. At last, the model of Figure 2(d) , called **TetraL2J**, is similar to the previous one, with two articulations per leg and no paws, but it has just four legs. The Table 3 shows the dimensions of the robots in meters. The simulated robots dimensions are approximately the dimensions of a dog.

The use of paws as showed in Figures 2(a) and 2(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.

Table 3. Dimensions of the simulated robots

	Body			**Thigh** and **shin**			**Paw**		
Robot	x	y	z	x	y	z	x	y	z
HexaL3J	0.80	0.15	0.30	0.05	0.15	0.05	0.08	0.05	0.09
TetraL3J	0.45	0.15	0.25	0.05	0.15	0.05	0.08	0.05	0.09
HexaL2J	0.80	0.15	0.30	0.05	0.15	0.05	-	-	-
TetraL2J	0.45	0.15	0.25	0.05	0.15	0.05	-	-	-

7 Results

In order to determine the best robot model to build, several experiments were conducted. The first experiments aimed to discover the best robot configurations, including the number of robot legs, and the number of segments per leg. Our intention was to construct a robot using the smallest amount of articulations, in order to simplify the hardware and reduce the robot building costs. Other tests were conducted to discover the most suitable method to be applied in the robot gait control. Several tests were made using a FSM angles table *(Angles)*, a locus based gait *(Locus)* and the half ellipse modeling *(Ellipse)*. The gait method adopted in our experiments was mainly the *trot* (two legs are lift at the same time), but some other experiments were made using four legged robots in a *walk* gait (just one leg per time is moved away from the ground). The Table 4 shows the results obtained in the accomplished experiments. Each type of experiment present in this table was repeated ten times using different random seeds, and the mean and standard deviation values relative to the fitness function and sensors information obtained from these experiments were calculated.

The first column indicates the experiment identification, the fifth and sixth columns show, respectively, the mean and the standard deviation of the fitness (F), the seventh and the eighth columns show the mean and the standard deviation of the distance covered by the robot (D) in meters and the last two columns show the mean and the standard deviation of the robot instability measure (G). The Figure 3 shows the *boxplot* graphic of all experiments, the Figure 4(a) shows the *boxplot* of the TetraL3J experiments (Exp 3, 7, 9 and 11) and the Figure 4(b) shows the *boxplot* of the TetraL2J experiments (Exp 8, 8, 10 and 12).

From the observed results presented in Table 4 and the fitness distributions of the Figure 3, we can reach to the following conclusions:

- Six legged robots are faster than four legged robots;
- Six legged robots without paws are a little bit faster than equivalent robots with paws;
- Four legged robots without paws did not achieve a satisfactory displacement;
- Although not significant statistically, the experiments using the FSM angles table were a little bit more efficient than the others;

Table 4. Results obtained in the simulations

				F		*D*		*G*	
Exp	*Gait*	*Method*	*Robot*	μ	σ	μ	σ	μ	σ
01	HexaL3J	Angles	Trot	4.364	0.850	6.894	1.026	0.600	0.182
02	HexaL2J	Angles	Trot	4.453	0.555	7.067	0.952	0.591	0.144
03	TetraL3J	Angles	Trot	2.618	0.804	3.831	0.938	0.502	0.222
04	TetraL2J	Angles	Trot	1.830	0.725	3.301	0.789	0.912	0.372
05	HexaL3J	Locus	Trot	3.450	0.361	5.720	0.559	0.665	0.152
06	HexaL2J	Locus	Trot	2.957	0.332	4.548	0.662	0.542	0.179
07	TetraL3J	Locus	Trot	1.509	0.512	2.327	0.773	0.547	0.166
08	TetraL2J	Locus	Trot	0.992	0.366	2.013	0.696	1.040	0.315
09	HexaL3J	Ellipse	Trot	1.801	0.430	2.435	0.579	0.359	0.107
10	HexaL2J	Ellipse	Trot	0.478	0.230	0.763	0.396	0.557	0.183
11	TetraL3J	Ellipse	Walk	1.493	0.480	2.174	0.763	0.447	0.106
12	TetraL2J	Ellipse	Walk	0.462	0.221	0.801	0.378	0.763	0.402

- The *trot* gait is faster than the *walk* gait, remaining as stable as in the *walk* gait.

The first conclusion is quite logical since the six legged robots have static stability in the *trot*, allowing to obtain faster velocities without falling risks. The second and third conclusions show that six legged robots don't need paws, but the four legged robots need them, and this is also results from the fact that six legged robots are more stable. The fourth conclusion demonstrates the good performance of experiments based on the FSM angles table, which can be due to the difficulties to calculate inverse kinematics in real time, and also because the endpoints sometimes do not follow exactly the planned trajectory (as observed in some simulations). The last conclusion shows that the use of the *trot* in the four legged robots with paws generates a stable gait, but not in the four legged robot without paws, showing the importance of the stability in gait control.

The Figure 5(a) show the evolution and the relationship between the fitness and the number of generations in a experiment. 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 5(b) shows the relationship between instability and velocity in the experiments accomplished using the TetraL3J robot.

We observed in the Figure 5(b) that an interesting relationship between velocity and instability exists. The population evolution emerged two separated groups (main diagonals in Figure 5(b)) that are both capable of moving across long distances, but one group assures a good stability of their movements, and the other group don't. This suggest we can obtain very stable solutions and solutions that remain under control.

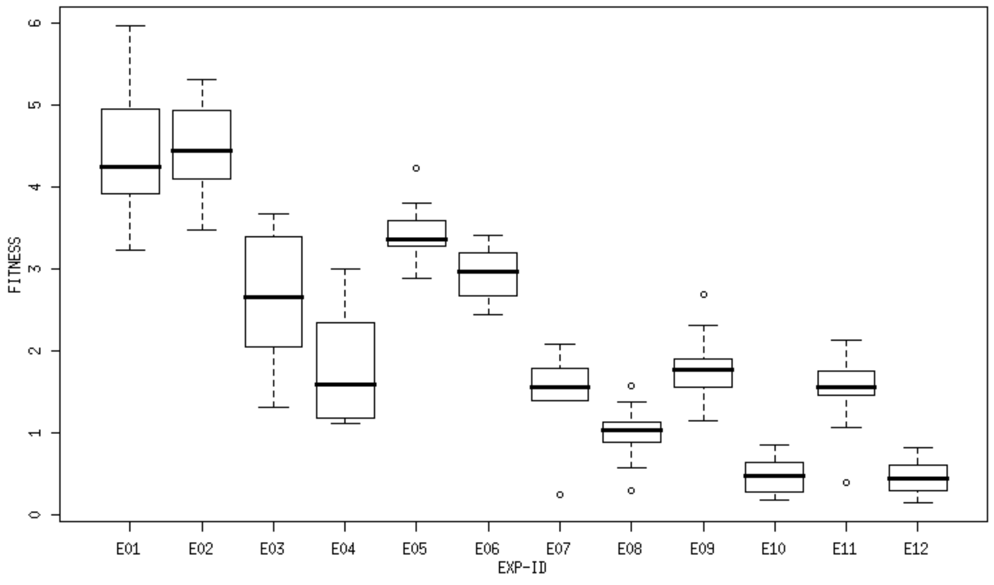


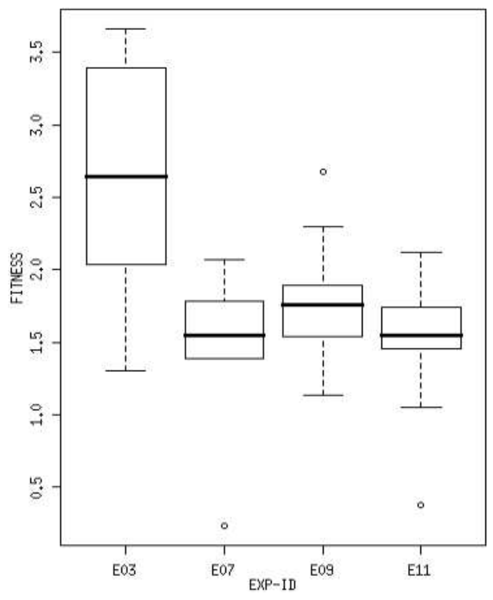
Figure 3. Boxplot of the all experiments

The main goal of our optimization search GA algorithm was to obtain control solutions with the lowest possible instability (solutions 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 5(b) shows that we achieved that goal and also obtained unstable good solutions. The Figure 6(a) shows the gait control simulation of a TetraL3J robot, and the Figure 6(b) shows the gait control simulation of a HexaL2J robot[6].

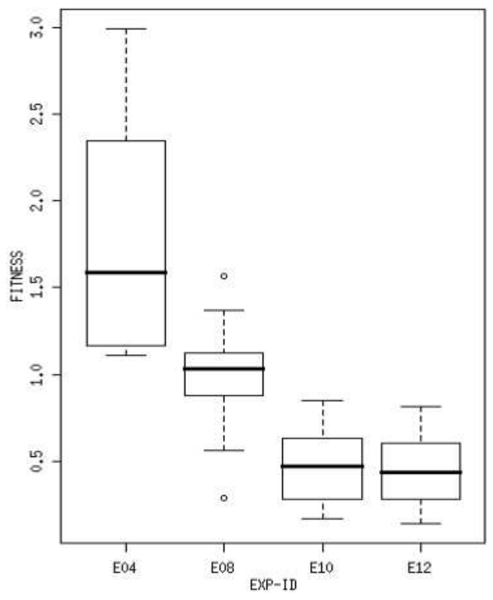
8 Conclusions and Perspectives

Based on the performed experiments, we observed that six legged robots are able to move faster than other robots, using very small or even no paws, like it occurs in the nature with some Arthropods: Insects and Arachnids are very fast animals if we consider the covered distances related to their small size, and the number of legs seems also to play and important role related to their movement skills. In four legged robots with paws, we observed through our simulations that endpoints with a larger support surface are necessary

[6]Some videos of accomplished experiments are available in http://www.inf.unisinos.br/~osorio/leggen/



(a)

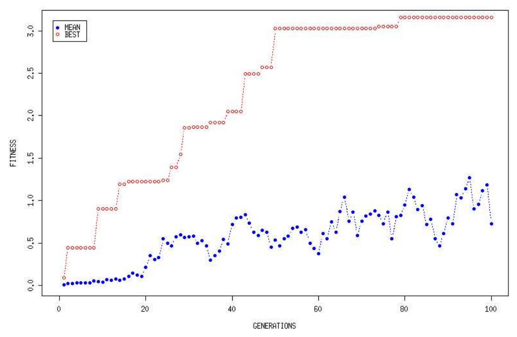


(b)

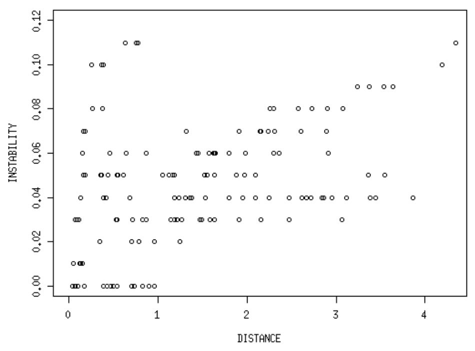
Figure 4. Boxplot of the some experiments

for a stable gait, and again it is interesting to try to make a parallel between our physically based and biologically inspired evolved robots and the real animals present in the nature. We also concluded that both robot models, the 6 legs with 2 joints and no paws robot and the 4 legs with 3 joints including the paws, are possible and viable configurations to be adopted in a physical construction of a real robot, and also our control gait system implementation can achieve a good performance and can provide a stable gait control.

In relation to the gait type, the *trot* is a quite efficient gait, including for the four legged robots with paws. On the other side, if we consider the modeling of the gait style (gait control method), the differences between the three implemented methods are not very significant, but we consider that the FSM angles table was more efficient and robust than the other two approaches, being chose to be adopted in our future research. The use of half ellipse trajectory didn't made the learning easier, nor the incorporation of this knowledge in the model. Besides that, the inverse kinematics method based on a half ellipse didn't help us to obtain a more generalized gait control. For more information about the LegGen simulator,



(a) Generations x Fitness



(b) Velocity x Instability

Figure 5. Fitness evolution

results and recent research please refer to this work [11].

The perspectives of this work includes to adapt gait control in order to make possible control robots moving over irregular surfaces and to climb and to descend the 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 a virtual model will help us to validate the control system in real conditions.

References

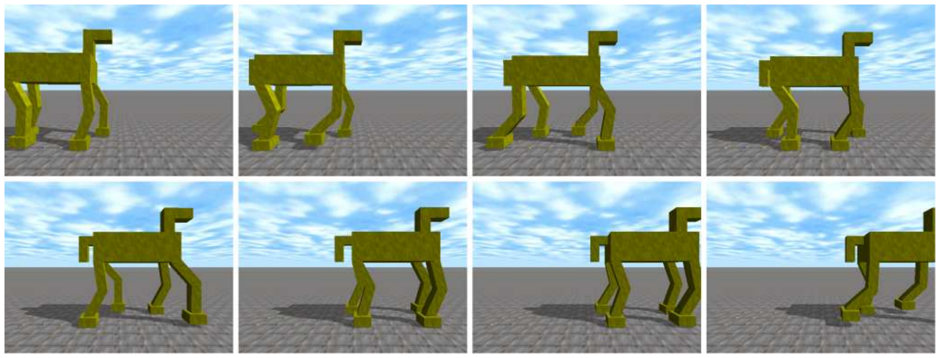
[1] G. A. Bekey. *Autonomous Robots: From Biological Inspiration to Implementation and Control*. MIT Press, Cambridge, MA, 2005.

[2] 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)*, pages 305–311, Edinburgh, UK, Aug. 2002. MIT Press.

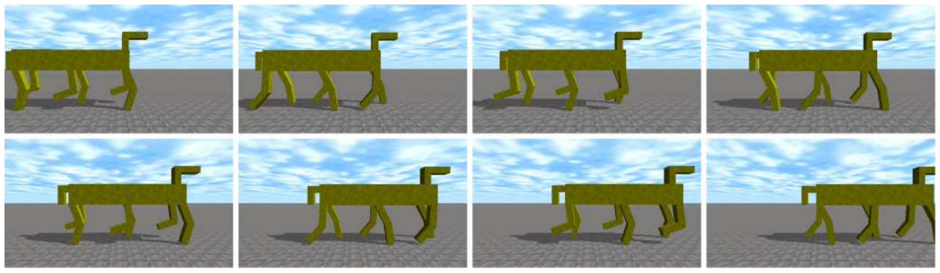
[3] R. P. Brent. *Algorithms for Minimization without Derivatives*. Prentice-Hall, Englewood Cliffs, NJ, 1973.

[4] 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. 5th European Conf. Genetic Programming (EuroGP)*, volume 2278 of *LNCS*, pages 258–267, Kinsale, Ireland, Apr. 2002. Springer-Verlag.

[5] 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.



(a) TetraL3J gait



(b) HexaL2J gait

Figure 6. Examples of evolved gaits

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

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

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

[9] 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.

[10] F. J. Heinen and F. S. Osório. HyCAR - a robust hybrid control architecture for autonomous robots. In *Proc. Hybrid Intelligent Systems (HIS)*, volume 87, pages 830–840, Santiago, Chile, 2002. IOS Press.

[11] M. R. Heinen. Controle inteligente do caminhar de robôs móveis simulados. Master's thesis - applied computing, Universidade do Vale do Rio dos Sinos (UNISINOS), São Leopoldo, RS, Brazil, 2007.

[12] M. R. Heinen and F. S. Osório. Applying genetic algorithms to control gait of physically based simulated robots. In *Proc. IEEE Congr. Evolutionary Computation (CEC)*, Vancouver, Canada, July 2006.

[13] M. R. Heinen and F. S. Osório. Gait control generation for physically based simulated robots using genetic algorithms. In *Proc. Int. Joint Conf. 2006, 10th Ibero-American Conference on AI (IBERAMIA), 18th Brazilian Symposium on AI (SBIA)*, LNCS, Ribeirão Preto - SP, Brazil, Oct. 2006. Springer-Verlag.

[14] M. R. Heinen and F. S. Osório. Uso de algoritmos genéticos para a configuração automática do caminhar em robôs móveis. In *Anais do Encontro de Robótica Inteligente (EnRI)*, Campo grande, MS, Brazil, July 2006.

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

[16] 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)*, pages 365–372, Espoo, Finland, June 2005.

[17] 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)*, volume 4, pages 1623–1628, Dubrovnik, Croatia, 2005.

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

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

[20] 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)*, pages 2618–2623, Nice, France, 1992.

[21] R. B. McGhee. Robot locomotion. *Neural Control of Locomotion*, pages 237–264, 1976.

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

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

[24] F. S. Osório, S. R. Musse, R. Vieira, M. R. Heinen, and D. C. Paiva. *Increasing Reality in Virtual Reality Applications through Physical and Behavioural Simulation*, volume 2, pages 1–45. Springer-Verlag, Berlin, Germany, 2006.

[25] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge Univ. Press, Cambridge, MA, 1992.

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

[27] R. Reeve and J. Hallam. An analysis of neural models for walking control. *IEEE Trans. Neural Networks*, 16(3):733–742, May 2005.

[28] 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)*, volume 2, pages 1638–1643, Las Vegas, NV, Oct. 2003.