Gait Control Generation for Physically Based Simulated Robots Using Genetic Algorithms

Milton Roberto Heinen and Fernando Santos Osório

Unisinos - PIPCA - Applied Computing
São Leopoldo, CEP 93022-000, Brazil
mheinen@turing.unisinos.br, fosorio@unisinos.br

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 fitness functions were accomplished, indicating how to compose a better multi-criterion fitness function to be used in the gait control of the legged robots. The best gait control solution and the best robot model were selected in order to help us to build a real robot in the future. The results also showed that it is possible to generate stable gaits using GA in an efficient manner.

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[1]. 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. 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[2]. 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[3]. Thus, it is interesting to generate the robot gait configuration in an automatic manner, using Machine Learning techniques to perform this task.

One of these Machine Learning techniques that are most adapted for this specific task are the Genetic Algorithms (GA)[4,5]. This is a reasonable choice because according to the Evolution's Theory[6], the locomotion mechanisms of several 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[5].

The main goal of this paper is to describe the LegGen System[7], that 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 2 describes the Genetic Algorithms and the GAlib software library adopted in our system; The Section 3 describes several concepts relative to legged robots, as the static and dynamic stability, the use of a physical simulation engine, the legged robot configuration, and the fitness functions used in the experiments; The Section 4 describes the LegGen system, and the robots used in the simulations; The Section 5 describes the executed experiments and the obtained results; and the Section 6 provides some final conclusions and future perspectives of this work.

2 Genetic Algorithms

Genetic Algorithms are optimization methods of stochastic space state search based on the Darwin's Natural Evolution Theory[6], that were proposed in the 60s by John Holland[8]. They work with a population of initial solutions, called chromosomes, which are evolved through several operations during a certain number of generations, usually finding a sub-optimal 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[5]. Individuals are selected to generate the next generation with probability proportional to their fitness values, and the crossover and mutation operations are applied in these individuals. Thus, each new generation tends to adapt and improve the quality of solutions, until we obtain a solution that satisfies a specific objective[4].

The Genetic Algorithms implementation used in our system was based on the GAlib software library[1], 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 open source library based on C++. In the LegGen System, a simple

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

Genetic Algorithm was used with a floating point type genome. In order to reduce the search space, alleles were used to limit generated values only to possible values for each parameter (maximum and minimum joint angles). In the executed experiments (described in Section 5) uniform crossover with probability of 0.6 was used, Gaussian mutation with probability of 0.05, population size of 50 individuals and 100 maximum generations.

3 Legged Robots

In this section, important concepts related to gait generation and control for legged robots are described, beginning with the stability concept.

3.1 Stability

A very important concept related to legged robot's gait is the stability. Thus, in order to make the robot move into an environment avoiding to fall down, it is necessary to have a stable gait, and this stability can be static or dynamic[1]. A robot is said to exhibit static stability when the robot's center of gravity remains inside the convex hull of the support polygon defined by the legs currently touching the ground. The major advantage of static stability is that the robot do not risk to fall down if it remains static during a certain period of time, where this stability can be maintained while an other leg moves or even if an energy failure occurs.

If the center of gravity of the robot is allowed to move outside of the support polygon convex hull and the robot continues to move in a controlled manner, the robot is said to exhibit dynamic stability. The dynamic stability is more difficult to reach, because it demands a sophisticated model of robot's dynamics and the use of inertia[1].

3.2 Mobile Robot Simulation

When someone wants to make experiments in the mobile robots research area, two alternatives are possible: (a) to execute 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 be realistic, but the simulation have the following advantages:

- When using simulated robots, doesn't exist the risk of robot damages and tasks as the exchange and recharge of batteries are not necessary[9];
- The robot positioning in order to restart a simulation can be accomplished automatically, without human intervention;
- The simulation clock can be accelerated, reducing the total amount of spent time for learning.

For these reasons, we chose to implement our initial experiments using a simulated robot, through the implementation of a very realistic robot simulator, using a physical simulation engine, so we can build simulated robots very similar to the real models.

3.3 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, making 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'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.

3.4 Gait Generation

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[7]. 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 for each joint, calculated by the equation:

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

[2] ODE – http://www.ode.org

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.

3.5 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:

$$F = D = (Px_1 - Px_0)\,, \tag{2}$$

where Px_0 is the robot start position and Px_1 is the final robot position in the x axis. Using this fitness function, the individuals that moved forward will be rewarded, and the individuals that moved backward will be punished, receiving a negative fitness.

We started believing that with this fitness function, the individuals selected to produce offsprings would 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 2 didn't lead us to a good solution, and thus the GA makes an almost random search in the search space.

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:

$$F = D * \mu_P\,, \tag{3}$$

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

$$F = D/(1+B)\,, \tag{4}$$

where B is calculated through the equation:

$$B = (\mu_P - L/2)^2\,, \tag{5}$$

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 didn't move the paws from the ground will be punished, receiving lower fitness values. This type of fitness function is more indicated for a *trot* gait.

Using the Equation 4, the gait learning became much more efficient, but we still had 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[10]:

$$G = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \overline{x}_x)^2 + \sum_{i=1}^{N}(y_i - \overline{x}_y)^2 + \sum_{i=1}^{N}(z_i - \overline{x}_z)^2}{N}}\,, \tag{6}$$

where N is the number of sample readings, x_i, y_i and z_i are the data collected by the simulated gyro at the time i, and $\overline{x}_x$, $\overline{x}_y$ and $\overline{x}_z$ are the mean values of gyro readings. The fitness function F is then calculated through the equation:

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

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[10]. 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:

$$F = D/(1+G)\,. \tag{8}$$

In Section 5 we describe the executed experiments using these four different fitness functions (Equations 2, 4, 7 and 8), and using the robots with four paws. These experiments were executed in order to verify which of these fitness functions is more efficient to generate stable gaits for legged robots.

4 Proposed System

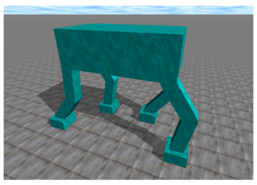
The LegGen System[3][7] was developed to accomplish the gait control of simulated legged robots in an automatic way. It was implemented using the C++ programming language and the free software libraries ODE and GAlib. The LegGen System works as follows: initially the file describing the robot is loaded, and the robot is created in the ODE environment according to file specifications. After this, the system parameters are loaded, 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 simulated 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 (60 seconds in our experiments);
- 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 until the predefined end time.

4.1 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. Thus, body parts as the head and the tail are not present in the modeled robots. In order to keep our



	Dimensions		
Part	**x**	**y**	**z**
Body	45.0cm	15.0cm	25.0cm
Thigh	5.0cm	15.0cm	5.0cm
Shin	5.0cm	15.0cm	5.0cm
Paw	8.0cm	5.0cm	9.0cm

Fig. 1. Robot model used in the simulations

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

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

developed and tested, before we defined the final main model, showed in the Figure 1, with four legs and three parts per leg. The simulated robots dimensions are approximately the dimensions of a dog.

5 Results

In order to determine the best fitness function, several experiments were conducted using the four fitness functions described in Section 3.5 (Equations 2, 4, 8, and 7). Our main objective was to discover which fitness function represents the best relation between stability (better gait performance) and cost (less hardware expenses).

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 8 and 7 need a gyroscope sensor to compute the fitness function, so these solutions are more expensive. The Equation 4 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 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 1 describes the results obtained in the four different type executed experiments. Due to stochastic nature of the Genetic Algorithms, each experiment described in Table 1 was repeated 30 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.

Table 1. Results obtained in the simulations

		F		D		G		B	
Exp	Fitness	μ	σ	μ	σ	μ	σ	μ	σ
E01	Equation 2	205.81	76.07	432.55cm	114.56cm	1.42	1.26	0.12	0.60
E02	Equation 4	225.29	51.60	439.63cm	57.32cm	1.02	0.42	0.01	0.02
E03	Equation 8	261.72	42.04	440.60cm	64.48cm	0.69	0.13	0.01	0.01
E04	Equation 7	268.77	37.84	454.55cm	68.48cm	0.69	0.17	0.01	0.01

The first column indicates the experiment identification, the second indicates the fitness function used, the third and fourth columns show, respectively, the mean (μ) and the standard deviation (σ) of the fitness function (F). The fifth and sixth columns show the μ and the σ of the distance covered by the robot (D) in centimeters, the seventh and eighth columns show the μ and the σ of the robot instability (G), and the last two columns show the μ and the σ of the average number of endpoints in the ground (B). The Figure 2 shows the boxplot graph and the 90% confidence interval of the Table 1 results.

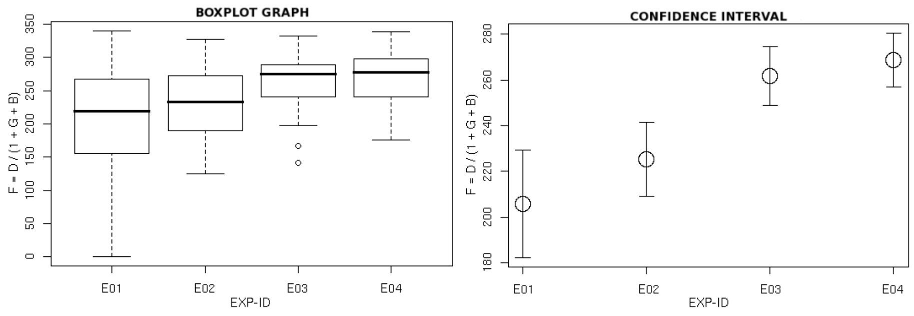


Fig. 2. Statistical analysis of results

From the observed results presented in Table 1, we can reach the conclusion that the distance covered by the robots in the experiment E04 are greater than the distance covered by the robots in the others experiments.

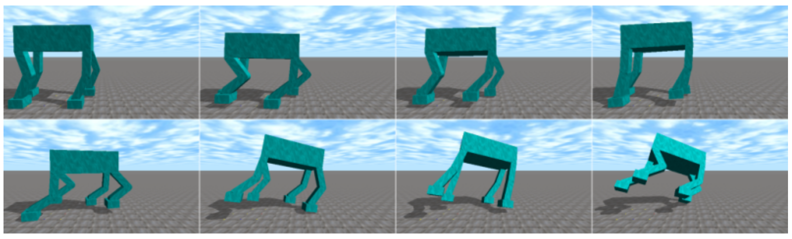


Fig. 3. Example of a generated gait (experiment 01)

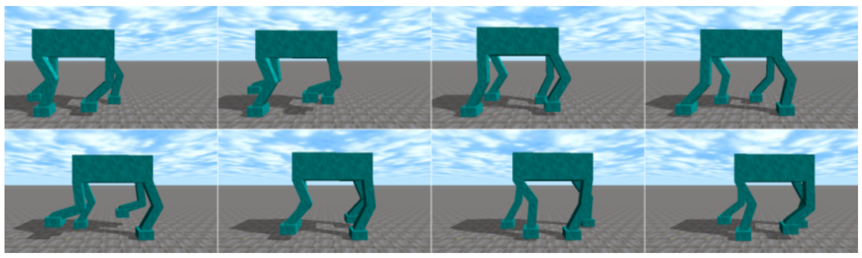


Fig. 4. Example of a generated gait (experiment 04)

Considering the instability, the use of gyroscope sensor (E03) increased the stability more than the use of the bumper sensors (E02). The Figure 3 shows an example of generated gait obtained in the experiment E01, and the Figure 4 shows an example of generated gait obtained in the experiment E04[4].

[4] Some videos are available in `http://www.inf.unisinos.br/~osorio/leggen`

6 Conclusions and Perspectives

Based on the performed experiments, we observed that fitness functions with additional sensorial information are very useful to generate stable gaits. We also concluded that although these gaits can be slower than gaits generated with simple fitness functions (few sensorial information), we are able to obtain more stable gaits.

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 a virtual model will help us to validate the control system in real conditions. We are also using the implemented system to test other robot configurations with different number of legs (4, 6, 8) and joints orientation (forward, backward) with promising results.

References

1. Dudek, G., Jenkin, M.: Computational Principles of Mobile Robotics. Cambridge Univ. Press, Cambridge, UK (2000)
2. Wyeth, G., Kee, D., Yik, T.F.: Evolving a locus based gait for a humanoid robot. In: Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS). Volume 2., Las Vegas, NV (2003) 1638–1643
3. Chernova, S., Veloso, M.: An evolutionary approach to gait learning for four-legged robots. In: Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS), Sendai, Japan (2004)
4. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading, MA (1989)
5. Mitchell, M.: An Introduction to Genetic Algorithms. MIT Press, Cambridge, MA (1996)
6. Darwin, C.: Origin of Species. John Murray, London, UK (1859)
7. Heinen, M.R., Osório, F.S.: Applying genetic algorithms to control gait of physically based simulated robots. In: Proc. IEEE Congr. Evolutionary Computation (CEC), Vancouver, Canada (2006)
8. Holland, J.H.: Adaptation in Natural and Artificial Systems. Univ. Michigan Press, Ann Arbor, MI (1975)
9. Wolff, K., Nordin, P.: Evolutionary learning from first principles of biped walking on a simulated humanoid robot. In: Proc. Advanced Simulation Technologies Conf. (ASTC), Orlando, FL (2003)
10. Golubovic, D., Hu, H.: Ga-based gait generation of sony quadruped robots. In: Proc. 3th IASTED Int. Conf. Artificial Intelligence and Applications (AIA), Benalmadena, Spain (2003)