Applying Neural Networks to Control Gait of Simulated Robots

Milton Roberto Heinen
UFRGS – Informatics Institute
CEP 91501-970, Porto Alegre, RS, Brazil
mrheinen@inf.ufrgs.br

Fernando Santos Osório
USP São Carlos – ICMC
CEP 13560-970, São Carlos, SP, Brazil
fosorio@icmc.usp.br

Abstract

This paper describes LegGen simulator, 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 two different methods: a finite state machine based on robot's leg joint angles sequences; and a recurrent neural network. The parameters for both methods are optimized using genetic algorithms. The model validation was performed by several experiments realized with a robot simulated using Open Dynamics Engine (ODE) physical simulation engine. The results showed that it is possible to generate stable gaits using genetic algorithms in an efficient manner, using these two different methods.

1 Introduction

The autonomous mobile robots have 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, gyro, etc) and motor's action planning and execution [5, 14]. 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 important disadvantages since they have problems to move across irregular surfaces and to cross borders and edges, like stairs. So, in order to make mobile robots better adapted to human environments and to irregular surfaces, they must be able to have a similar locomotion mechanism used by the humans and animals, that is, they should have legs [5, 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. 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 [4]. Thus, it is interesting to generate the robot gait configuration in an automatic manner, using machine learning techniques to perform this task.

In our previous works, we made a comparative study between robots with four (tetrapod) and six (hexapod) legs [13], and also about the use and the influence of different fitness functions [9, 10] used in robot control evolution. This paper shows a comparative study between the following legged robot control strategies: (i) Control based on FSM (finite state machine); (ii) Control based on ANN (artificial neural networks). In both strategies the parameters optimization was done using genetic algorithms (GA).

This paper is structured as follows: Section 2 describes several concepts relative to mobile robots simulation; Section 3 describes related works in control of legged robots; Section 4 describes the proposed model, called LegGen; Section 5 describes the accomplished experiments and the obtained results; and Section 6 provides some final conclusions and future perspectives.

2 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[23, 17]:

- 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 spent 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

1522-4899/08 $25.00 © 2008 IEEE
DOI 10.1109/SBRN.2008.22

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.

2.1 Physics simulation engine

In order to obtain a 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 and also to interact with the environment they are inserted in. Especially, it is necessary that the robot suffers from instability and fall down if badly positioned and controlled, and also it should interact and collide against the environment objects in a realistic manner [20]. 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 analyzing different possibilities, we chosen a widely adopted physics simulation library, called Open Dynamics Engine (ODE)[1].

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 different 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 with a high precision level [20].

3 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 [16]. 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 [22]. 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 [19], where each joint was controlled by a simple finite state machine, to the very successful algorithmic control of bipeds and quadrupeds by Raibert [21].

Lewis [18] 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 [3] used genetic programming to evolve the control parameters of several robot types. Jacob [15], on the other hand, used reinforcement learning to control a simulated tetrapod robot. Reeve [22] 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 [7]. In our approach, we use in the fitness function, beyond distance traveled, sensorial information (gyroscope and bumpers) to guarantee stable and fast gaits [9, 12].

4 Proposed model

LegGen simulator[2] [9, 13, 12] was developed to accomplish the gait control of simulated legged robots in an automatic way. This simulator is composed of several modules, showed in Figure 1. The module *Robotnik* is responsible for the robot and virtual environment creation using the ODE library. The module *Evolution* is responsible for the evolution of the control parameters using genetic algorithms. The module *Sensorial* is responsible for sensorial information reading during simulation and fitness calculation for each individual. The module *Viewer* is responsible for the visualization of results in a three-dimensional graphic environment. The module *Controller*, implemented using an ANN, is responsible for the robot's joint control.

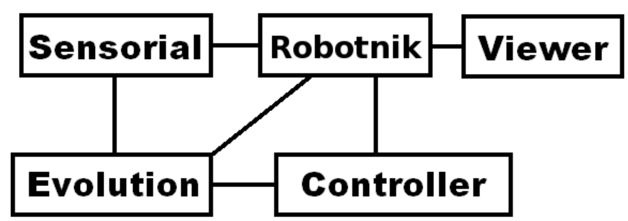


Figure 1. LegGen modules

LegGen simulator 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

[1] Open Dynamics Engine (ODE) – http://www.ode.org

[2] LegGen – http://www.inf.ufrgs.br/~mrheinen126/leggen

this, the simulator 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 robot is placed in the starting position and orientation in the simulation environment;
- The genome is read and the control parameters are set;
- The physical simulation is executed during a predefined amount of time (30 seconds in our experiments);
- Gait information and sensor data are captured during each individual physical simulation;
- Fitness is calculated and returned to 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.

In LegGen simulator, the gait control is accomplished through two strategies: (i) a finite state machine (FSM); (ii) an artificial neural network (ANN). The following sections describe these control strategies.

4.1 Finite state machine control

In LegGen simulator, 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 [2]. 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.

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 GA only generates values between the maximum and minimum accepted values for each specific parameter.

4.2 Neural control

Besides the use of FSMs to control legged robots, we also can use artificial neural networks (*artificial neural networks* – ANN) [8]. This approach has some important and specific limitations: it is quite difficult to have an a priori information about the generation of the control parameters [11]. Since we do not have available the exact and correct sequence of values that should be sent to control the actuators, then it is usually not possible to apply traditional supervised learning algorithms, like *back-propagation* and other similar ones. This is the main reason we decided to adopt genetic algorithms to evolve synaptic weights.

GAs can adjust synaptic weights with the advantage they do not need any local information or local error measure in order to adapt the weights, and so we do not need a training dataset (supervised learning). The weights can be coded into the chromosomes and evolved, using a fitness function to evaluate the robot performance controlled by this evolved ANN. On the other hand, the use of ANNs has some main advantages when used to control robot gait: ANNs are more robust to noise, continue to perform well even when faced to unseen situations, and they usually can obtain a good generalized behavior.

The ANN inputs are the present robot joint angles values (angles at time t, normalized in the range from -1 (α_{mim}) to +1 (α_{max}). In the ANN outputs are obtained the joint angles in the next time step $t+1$, also normalized in the range [-1:+1]. After some preliminary tests, we choose the Elman model of recurrent ANNs, which was very satisfactory when applied in this problem where we need to predict a temporal behavior (sequencing joint angles). The Elman networks are MLP nets with feedback connections from and back to the hidden layer. These connections allow the Elman nets to learn temporal sequences of patterns and then, from the joint angles patterns in time t, they can generate the next joint angles pattern in their outputs. We adopted the hyperbolic tangent function as neuron's activation function, and also the synaptic weights were limited ranging from -1 to +1, which simplify the GA weights optimization. This ANN model and parameters setup was empirically tested and showed to be well suited to the problem in question.

4.3 Evolution

In our model, the control parameters are evolved using genetic algorithms. The GA 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++

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

library. In LegGen System, a genetic algorithm as described by Goldberg in his book [6] 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. Table 1 shows the parameter' values used by GA.

Table 1. Parameters of LegGen simulator

Par-ID	Parameter	Value
1	One point crossover	0.80
2	Mutation rate	0.08
3	Population size	350
4	Number of generations	700

The fitness evaluation uses the following sensorial information that must be calculated: (a) the distance D covered by the robot; (b) instability measure G; 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 x start position and Px_1 is the end x position.

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 [5]. The instability measure G (Gyro) is then calculated by the following equation [7]:

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

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:

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

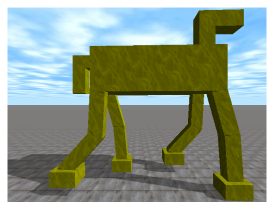
After finished the sensorial information processing, the fitness function F is then calculated through the equation:

$$F = D/(1 + G) \tag{5}$$

where L is the number of robot legs. Analyzing the fitness function, we see that 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.

4.4 Modeled robot

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 usually 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 simulator to accept more complex robot models and joints. Several robot types were developed and tested, before we defined the final main model, presented in Figure 2.



	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

Figure 2. Modeled robot

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.

5 Results

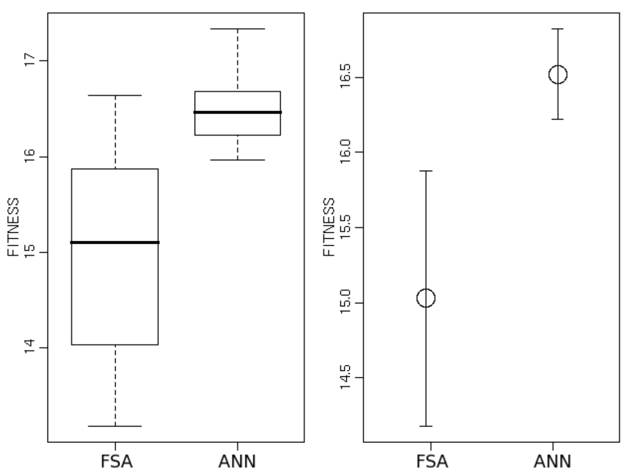
This section describes our experiments and the achieved results. These experiments were done in order to evaluate the GA parameters optimization and robot behavior in both control strategies (FSM and ANN), as described in the previous sections. For each control strategy, we executed 10 different tests, which are presented here. Table 2 shows the obtained results, where we can see each control strategy (FSM and ANN) and the values of the fitness, distance and gyro instability measure respectively (F, D, G) indicated for each experiment (**E**) in both strategies. The two lines below in the table are the average (μ) and standard deviation (σ) indicated over the 10 experiments.

In the experiments using the FSM, we fixed the number of states in the automata to four. In the experiments using the neural network we adopted a network with three neurons

Table 2. Evaluation of the control strategies

	FSM			ANN		
E	*F*	*D*	*G*	*F*	*D*	*G*
1	14.04	32.17	0.128	16.27	29.19	0.079
2	14.28	32.38	0.126	16.63	28.31	0.070
3	13.18	30.33	0.129	16.99	27.85	0.063
4	15.87	26.81	0.069	16.68	27.91	0.067
5	16.64	36.60	0.120	16.16	28.20	0.074
6	16.48	27.69	0.068	15.97	31.13	0.093
7	14.88	31.69	0.112	17.33	29.63	0.070
8	13.77	29.02	0.110	16.65	29.04	0.074
9	15.33	34.41	0.124	16.29	30.15	0.085
10	15.80	37.01	0.134	16.23	29.81	0.083
μ	**15.03**	**31.81**	**0.112**	**16.52**	**29.12**	**0.076**
σ	**1.19**	**3.48**	**0.024**	**0.42**	**1.08**	**0.009**

in the hidden layer. These parameters were defined after a careful preliminary study based on experiments. We spent a total of 149.22 hours processing the final experiments of Table 2. Figure 3 shows the boxplot graph and the confidence interval (CI) of 95%, related to the fitness values obtained in the experiments presented in Table 2.



(a) Boxplot (b) Confidence interval

Figure 3. Boxplot and confidence interval

According to Figure 3 we can affirm that the results obtained by the ANN are clearly superior to those obtained using the FSM, since the confidence intervals are not superposed. Besides that, the results obtained using the SM are more unstable with a large variability. Figure 4 compares the fitness improvement of the population during the evolution (best and average fitness) obtained for each control strategy. The experiments showed n this figure are those that achieved the best results in our simulations.

It is clear that the evolution of the neural control parameters needs more generations (epochs) in order to achieve good results. This is due to the fact that we have a bigger parameters state space when optimizing the ANN (we optimize 3 parameters in the FSM and 44 weights in the ANN). Figure 5 shows one example of the robot gait controlled by an optimized FSM and Figure 6 shows one example of a robot gait obtained using a trained ANN[4].

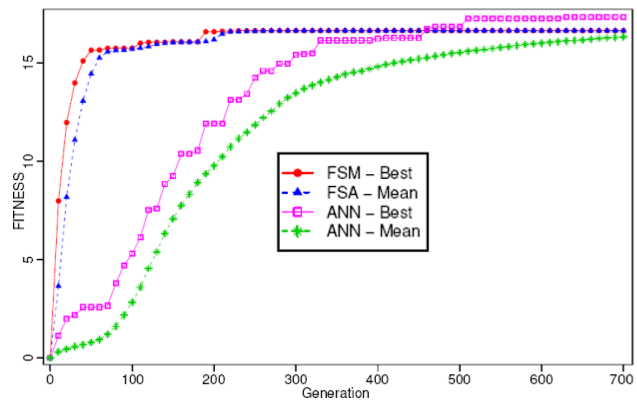


Figure 4. GA optimization

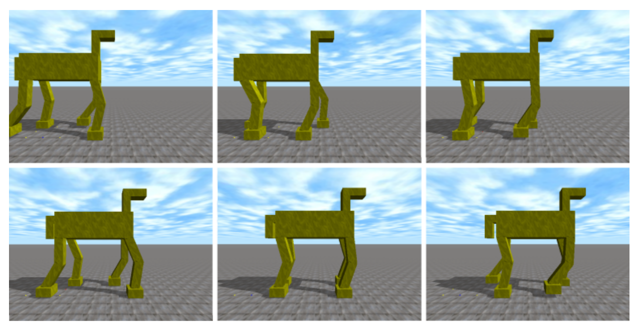


Figure 5. FSM robot control

6 Conclusions and perspectives

The main goal of this paper was to describe LegGen simulator, which was developed in order to study the automatic configuration of parameters used to control the gait of legged robots. In our simulator, the gait control was achieved using genetic algorithms. The GA evolves parameters used to control the robot actuators and this evolution was tested into a virtual environment using the ODE rigid body dynamics simulation tool. The robot joints are controlled using two different strategies: (i) GA evolved a finite state machine and (ii) GA evolved an artificial neural network. Several experiments were achieved, comparing both approaches and demonstrating (with a valid statistical

[4]Some demonstration videos are available in LegGen website.

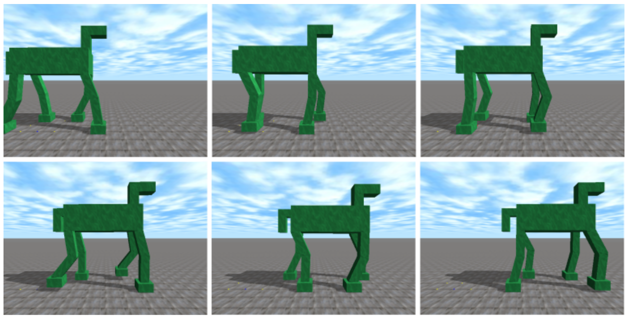


Figure 6. ANN robot control

analysis) that the neural controller is superior to the FSM controller (superior fitness), obtaining a better performance (more stable, better displacement).

Future works include improving the robot gait in order to walk on irregular surfaces and go upstairs or downstairs, and to implement in hardware the simulated robot, once we had now acquired sufficient experience in order to design, implement and fine tune the control of the legged robots.

Acknowledgments

This work was supported by Fapergs, CAPES and CNPq.

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] 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.

[4] 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.

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

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

[7] 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.

[8] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice-Hall, Upper Saddle River, NJ, 2nd edition, 1999.

[9] 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.

[10] 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.

[11] M. R. Heinen and F. S. Osório. Neural networks applied to gait control of physically based simulated robots. In *Proc. Int. Joint Conf. 2006, 9th Brazilian Neural Networks Symposium (SBRN)*, Ribeirão Preto, SP, Brazil, Oct. 2006.

[12] M. R. Heinen and F. S. Osório. Applying genetic algorithms to control gait of simulated robots. In *Proc. IEEE Electronics, Robotics and Automotive Mechanics Conf. (CERMA) 2007*, pages 500–505, Cuernavaca, Mexico, Sept. 2007.

[13] M. R. Heinen and F. S. Osório. Evolving gait control of physically based simulated robots. *Revista de Informática Teórica e Aplicada (RITA)*, XVI(1):119–134, 2007.

[14] M. R. Heinen, F. S. Osório, F. J. Heinen, and C. Kelber. SEVA3D: Using artificial neural networks to autonomous vehicle parking control. In *Proc. IEEE Int. Joint Conf. Neural Networks (IJCNN)*, Vancouver, Canada, July 2006.

[15] 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.

[16] 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.

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

[18] 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.

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

[20] 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 of *Research in Interactive Design – Proc. Virtual Concept Conf. 2006*, pages 1–45. Springer-Verlag, 2006.

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

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

[23] K. Wolff and P. Nordin. Evolutionary learning from first principles of biped walking on a simulated humanoid robot. In *Proc. Business and Industry Symposium of the Advanced Simulation Technologies Conf. (ASTC'03)*, pages 31–36, Orlando, FL, Apr. 2003. SCS.