Evolving autonomous locomotion of virtual characters in a simulated physical environment via neural networks and evolutionary strategies

Stefan Marks
email: dev.stefan.marks@gmx.net

Wolfram Conen
email: wolfram.conen@informatik.fh-gelsenkirchen.de

Thomas Kollakowsky
email: thomas.kollakowsky@informatik.fh-gelsenkirchen.de

Gregor Lux
email: gregor.lux@informatik.fh-gelsenkirchen.de

University of Applied Sciences Gelsenkirchen
Department of Computer Science
Neidenburger Str. 43, 45877 Gelsenkirchen, Germany
http://www.informatik.fh-gelsenkirchen.de

Abstract

In this paper we examine principles to automatise or to support the process of animation of an articulated character by allowing it to "learn" its movements autonomously. For this, it is necessary to model the physical properties of the character, connect virtual sensors and actuators to a neural network, and adjust the weights and thresholds of the network with evolutionary strategies. We conclude with a discussion and showcase of the results and present ways for further improvement.

Keywords

Animation, Virtual Reality, Physics, Simulation, Neural Network, Evolution, Gait

1. Introduction

Animating an articulated character (e.g. for a movie) is a tedious process that has not changed much since the beginning days of animation. Even with todays modern computer technology and computer based characters, tools or methods only support the animator (e.g. motion capture, bones, frame interpolation, constraints), but do not really relieve him of the task of moving every body member separately to achieve a natural looking locomotion.

We examined the approach to enable the animator to simply give the virtual character combinations of commands, like "walk forward", "turn left" which are then executed in a natural looking way that needs only minor tuning for a realistic look.

To achieve this, the virtual character needs an artificial intelligence in form of a neural network "attached" to his virtual body. This network receives input from various kinds of sensors in the body like the position of each joint, the force at the bottom of the feet, the velocity and orientation of the head, etc. It continuously processes this information and generates output signals for actuators like virtual muscles that enable the autonomous movement.

The most important question is, how the neural networks "learns" the correct behaviour. One possible way could be to enable the network to learn continuously while trying to improve its movements. This task is rather complex, firstly because of the difficulty to immediately decide whether a movement is good or bad, and secondly because of the choice and implementation of an appropriate learning algorithm.

In the following we examine the approach of tuning the neural network parameters with evolutionary algorithms.

2. Related Work

The basic idea of using evolutionary algorithms to evolve movement dates back to [11]. Since then, several approaches related to this idea have been examined using either principles similar to our work [7; 9; 10] or different (e.g. genetic programming on virtual register machines [14; 15], neural network architectures similar to the human cerebellum [12], additional evolution of the morphology [8; 13]).

The commercial software *endorphin* from NaturalMotion[1] combines readily-trained neural networks with accurate biomedical models of bi- and quadruped characters.

Our work differs in at least three aspects from one or more of these publications:

- **Given morphology**
  With respect to the planned application, our work starts with a given character morphology rather than the need to evolve it at first.
- **Multiple forms**
  The above works concentrate mostly on just one form of a character whereas we evolve the gaits of a mono-, bi- and a quadruped.
- **Universality**
  Instead of creating a single monolithic program, we focus on a modular design that enables the reuse and exchange of components for related or even different projects.

[1] `http://www.naturalmotion.com`

3. Concept

Figure 1 shows the main functional blocks necessary for the task:

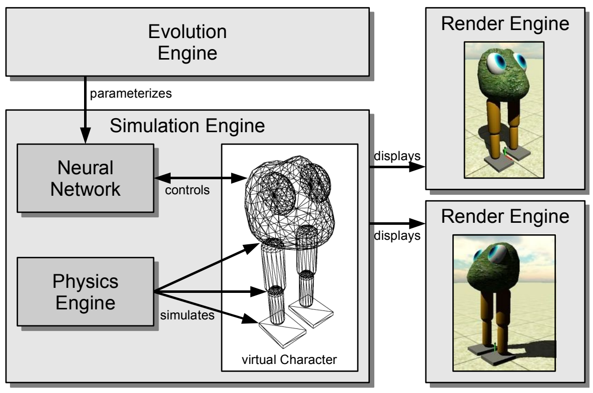

Figure 1: Functional blocks

The **physics engine** is responsible for the realistic simulation of the physical properties of the virtual character and its environment (e.g. mass, inertia, forces, impulses, joints, gravity, friction).

The **neural network** processes the sensoric information from the body and from the environment of the virtual character (e.g. joint positions, forces, contacts) and outputs controller information for the actors (e.g. muscles, motors).

Physics engine and neural network together form the **simulation engine**.

One or more **render engine**s connect via network to the simulation engine and display the simulated data from the view of virtual cameras. The display may be a simple screen, a beamer or the monitors of a head mounted display (HMD).

The **evolution engine** is parameterizing the weights, thresholds and other values of the neural network according to fitness results of test runs. These results are calculated from the ability of a number of virtual characters to complete a special task with their neural network, e.g. walking forward, standing still. Only the best rated networks are then chosen, modified and passed on into the next generation of virtual characters. After some time (and with a well chosen rating function) the virtual characters will start to evolve towards the wanted behaviour.

4. Results

4.1. Characters

We tried to evolve the locomotion of three kinds of virtual characters (see figure 2):

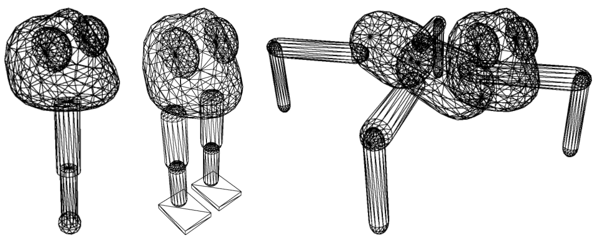

Figure 2: Character morphologies

The **monoped** consists of a relatively heavy head, a leg tilteable in two directions (2-DOF, **d**egrees **o**f **f**reedom) and a powerful extendable piston with a frictious foot for jumping.

The **biped** is built with the same head as the monoped, two humanlike legs with 1-DOF hip, knee and ankle joints and plates as feet.

The **quadruped** consists of a two segmented body with a head, and four legs with 2-DOF hip joints and 1-DOF knee joints.

The structure of the neural network used in the virtual characters (see figure 3) is based upon results of the previously presented related works.

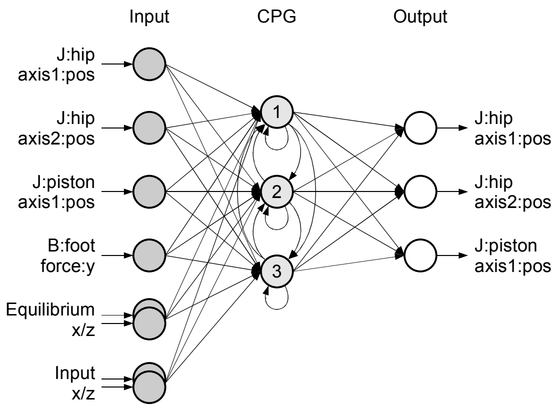

Figure 3: Neural network of the monoped

The hidden CPG layer consists of three interconnected neurons (compare [6; 7; 8]) forming a Wilson-Cowan oscillator (see [4, ch. 3]). This oscillator is capable of generating waveforms whose frequency, offset and phase is controllable by the input neurons.

The oscillator is connected to the output layer neurons driving the joint motors of the character. The body sensors are connected to the input layer. The networks of the three character types differ only by the number of input and output layer neurons.

With respect to [3] the CPG neurons were initialised with values that allowed them to oscillate from the beginning of the evolution process on.

4.2. Evolution

An important question is the choice of evolutionary operators and parameters (see e.g. [1] for details). Due to lack of time we were not able to test various combinations on their influence on the results.

We chose a **population size** proportional to the dimension of the neural network to be optimised. For the monoped and biped (62 respectively 93 parameters) we used a population size of 100, whereas for the quadruped (206 parameters) we chose 250.

Variations in the mutation rate and mutation probability, the number of parents per child and the number of children per parents had only minor influence on the results. The final operators and parameters used for the evolution processes are listed in table 1.

Algorithm	evolutionary strategies
Genotype	real value
Population size	100 (monoped, biped), 250 (quadruped)
Reproduction	2 children per parents
Fitness rating	proportional
Initialization	non-random with oscillating CPGs
Selection	tournament selection, tournament size 5
Recombination	none because of “competing convention” (see [5, S. 287])
Mutation	randomly with adaptive stepwidth by 1/5 success rule and 10% mutation probability
Reinsertion	elitest reinsertion
Migration	none
Termination	running mean

Table 1: Operators and parameters for the evolution process

The most important aspect is the **fitness rating**. On one hand it is the only factor that the evolution engine can rely on to determine the fittest individuals. On the other hand we found out that the exact and precise construction of the fitness rating is essential for a useful result of the evolution process.

As long as the interpretion of the fitness rating leaves any gaps, the evolution process will find them. For the first experiments we only tried to rate a character by its distance from the starting point after a specific amount of time or when it had fallen down. The evolution algorithm found a way for all three character forms to "cheat". Design faults in the physical simulation like erroneous implementations of the joint motor control or in the collision detection were found which resulted in "explosive" effects that thrusted the character forwards. Although obviously not very naturally looking this way of locomotion was the best result for a fitness rating when only concerning the travelled distance.

In the end we came up with a fitness formula that takes several factors into account:

The value d is the **distance** of the center of the head $\vec{x}_h(t)$ **to the target position** $\vec{x}_t(t)$ it "should" have at time t. r is a maximum allowed radius around the target position and Δt is the time step of the Simulation. The further away the head is from the radius, the smaller d gets.

$$d(t) = \frac{\Delta t}{\max(0, |\vec{x}_h(t) - \vec{x}_t(t)| - r)} \qquad (1)$$

The **energy factor** e is calculated from the squares of all accelerations a of the joints J_i, $i \in (1 \dots N_J)$. This formula is derived from the "minimum torque change model", described in [2, ch. 1.3.1]. The logarithm prevents the term from growing too fast.

$$e(t) = \log\left(1 + \sum_{i=1}^{N_J} a(t)_{J_i}^2\right) \qquad (2)$$

Certain movements, positions or behaviours are regarded as **penalty states**. They prevent the addition of the above mentioned factors to the total fitness rating.

$$p(t) = \begin{cases} 1 & \text{if not in penalty state} \\ 0 & \text{if in penalty state} \end{cases} \qquad (3)$$

The **number of steps** over time can be determined by the change of the number of feet that touch the ground. The number of steps s is limited to s_{max} to prevent a character doing a "tap dance" to get an overproportional fitness rating.

The factors are combined to a fitness rating term

$$Z(t) = \frac{d(t) \cdot p(t)}{1 + e(t) \cdot \eta} \qquad (4)$$

(with η as a weighing factor for the influence of the energy term) that is accumulated over time and finally multiplied with the number of steps to get the overall fitness rating for the virtual character

$$Z = \min(s, s_{max}) \cdot \sum_{t=t_{\text{begin}}}^{t_{\text{end}}} Z(t). \qquad (5)$$

4.3. Evolution runs

The **monoped** was the first virtual character we experimented with. Several attempts were necessary to remove errors in the physical simulation, the neural network or in the fitness formula.

The first successful evolution run 9 resulted in a character that was able to perform up to 6 jumps before falling down (see figure 4a).

Further tests with randomly initialised neural networks or 2- respective 4-neuron

CPGs resulted only in characters unable to perform any locomotion.

For the **biped** too we needed several runs in the beginning for getting rid of all conceptual errors and flaws in the physical simulation.

Evolution run 20 produced a reflex based walking gait (see figure 4b). The biped walks in a rather stiff way until gravity is turned off and the feet lose contact with the ground. Then the movements stops until contact is reestablished by turning gravity back on.

For evolution run 24 the CPG was replaced by a 2-neuron version (see figure 4c). The resulting biped walked more naturally because it was driven by CPG-patterns.

A control run with identical parameters (see figure 4d) surprisingly resulted in a biped that was able to walk reflex-based only by its ankles.

For the evolution runs of the **quadruped** we were able to use our gathered experience with the previous two character forms.

The finess rating for evolution run 1 was again purely based upon the distance from the starting point. The virtual character "abused" a flaw in the collision detection of the physical simulation to gain a high locomotion speed.

Evolution run 2 was characterised by too strong influence of the energy term which led to a total motionless character.

The reduction of that factor in run 3 finally produced a walking gait. But because of asymmetric movements the character was only able to walk in a circle (see figure 4e).

Like with the other character types we tested the influence of randomly initialised networks in run 4. The result was another character unable to move.

Control runs with identical parameters repeatedly lead to characters with different but nevertheless successful kinds of locomotion (see figure 4f).

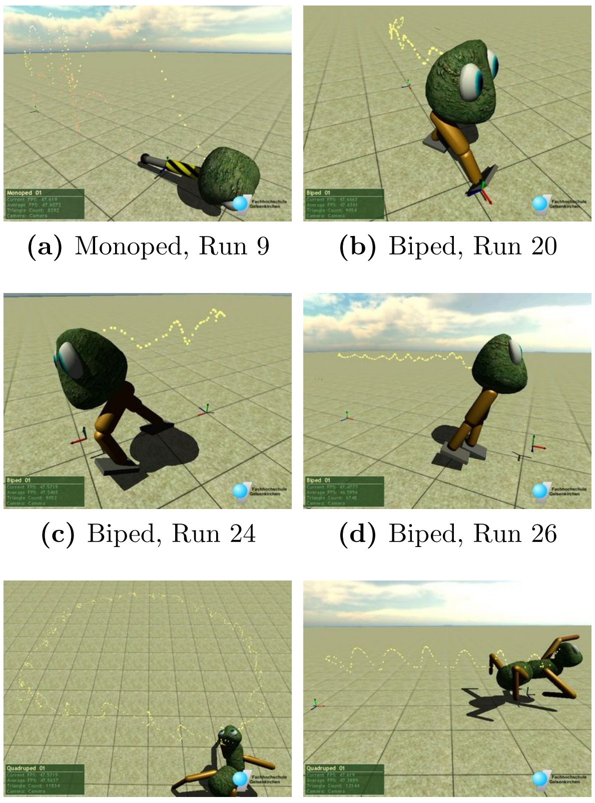
(a) Monoped, Run 9 **(b)** Biped, Run 20
(c) Biped, Run 24 **(d)** Biped, Run 26
(e) Quadruped, Run 3 **(f)** Quadruped, Run 6

Figure 4: Results of the evolution runs

5. Conclusions

Our experiments prove that it is possible to evolve autonomous locomotion with evolutionary algorithms[2].

[2]Videos of the results are available at `http://cgr.informatik.fh-gelsenkirchen.de/cerebellum/videos`.

Nevertheless the amount of work to produce a naturally looking gait is not reduced but shifted away from the animation to the fine tuning of the evolutionary parameters. Although many constellations of fitness rating formulae were tested, the resulting virtual characters were not moving as naturally-looking as we had expected at the beginning. With respect to the originally planned application of a software taking primitive commands from the animator, there is some work left for further investigation. In any case the confrontation of the animator with the fine-tuning of the fitness formula should be avoided and instead be replaced by e.g. some kind of predefined presets of parameters.

Other necessary improvements can be one or more of the following:

- The **fitness rating** should be designed more carefully and with less possibilities for misinterpretations. Several resulting characters did not show any rhythmic movements but only moved as little as necessary to prevent a low rating. The rating is the most vulnerable part of the whole principle. Small changes can decide between success and failure.
- The **design of the the virtual characters** could be improved by more details like anatomically correct limbs or biomedically modelled simulations of nerves, muscles, tendons etc.
- A more complex **neural network** could lead to refinement of the generated motion patterns but would exponentially increase the search space for the evolutionary algorithm. One may think about a combination of online-learning of short term cause-and-effect aspects during a simulation run and the long term progress by evolutionary means.
- The **search space** of the evolutionary algorithm may be reduced e.g. by eligibility tests of the resulting neural networks. Those networks that are unable to generate repetitive patterns do not have to be tested in the simulation run.
- **Interaction** with the virtual character during its simulation phase may speed up the evolution process. Like holding a child that learns to walk at its hands, it may be possible to support the character in a similar way.

All of these factors should contribute to better looking results and finally lead to animation software where autonomous virtual characters are told “what to do” instead of moving and animating them tediously in every detail.

References

[1] Lawrence David Davis, Kenneth de Jong, Michael D. Vose, and L. Darrell Whitley, editors. *Evolutionary Algorithms.* Springer Verlag, 1999.

[2] Stefan Künzell. *Motorik und Konnektionismus: Neuronal Netze als Modell interner Bewegungsrepäsentationen.* bps-Verlag, 1996.

[3] Boonyanit Mathayomchan and Randall D. Beer. Center-crossing recurrent neural networks for the evolution of rhythmic behavior. In *Neural Computation*, volume 14, pages 2043–2051, 2002.

[4] Kazuki Nakada, Tetsuya Asai, and Yoshihito Amemiya. Design of an ar-

tificial central pattern generator with feedback controller. In *Intelligent Automation and Soft Computing*, volume 10, pages 185–192. TSI Press, 2004.

[5] Volker Nissen. *Einführung in Evolutionäre Algorithmen.* Vieweg Verlag, 1997.

[6] Chandana Paul. Bilateral decoupling in the neural control of biped locomotion. In *2nd International Symposium on Adaptive Motion of Animals and Machines*, 2003.

[7] Chandana Paul. Sensorimotor control of biped locomotion based on contact information. In *International Symposium on Intelligent Signal Processing and Robotics*, 2004.

[8] Chandana Paul and J.C. Bongard. The road less travelled: Morphology in the optimization of biped robot locomotion. In *Proceedings of The IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2001.

[9] Torsten Reil and Phil Husbands. Evolution of central pattern generators for bipedal walking in a real-time physics environment. In *IEEE Transactions on Evolutionary Computation*, volume 6, pages 159–168. Urban & Fischer, April 2002.

[10] Torsten Reil and Colm Massey. Biologically inspired control of physically simulated bipeds. In *Theory in Biosciences*, volume 120, pages 327–339. Urban & Fischer, December 2001.

[11] Karl Sims. Evolving virtual creatures. In *SIGGRAPH '94 Proceedings*, pages 15–22, 1994.

[12] Russell L. Smith. *Intelligent Motion Control with an Artificial Cerebellum.* PhD thesis, University of Auckland, New Zealand, 1998.

[13] Steffen Wischmann. Entwicklung der Morphologie und Steuerung eines zweibeinigen Laufmodells, 2003.

[14] Krister Wolff and Peter Nordin. An evolutionary based approach for control programming of humanoids. In *Proceedings of the Third IEEE-RAS International Conference on Humanoid Robots*, 2003.

[15] Krister Wolff and Peter Nordin. Learning biped locomotion from first principles on a simulated humanoid robot using linear genetic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 2003.