# The tango of a load balancing biped

Eric D. Vaughan, Ezequiel Di Paolo, Inman R. Harvey

Centre for Computational Neuroscience and Robotics,
University of Sussex,
Brighton, BN1 9QH
{e.vaughan, ezequiel, inmanh}@sussex.ac.uk

**Summary.** One of the most popular approaches to developing bipedal walking machines has been to record the human gait and use it as a template for a walking algorithm. In this paper we demonstrate a different approach based on passive dynamics, neural networks, and genetic algorithms. A bipedal machine is evolved in simulation that when pushed walks either forward or backwards just enough to release the pressure placed on it. Just as a tango dancer uses a dance frame to control the movements of their follower, external forces are a subtle way to control the machines speed. When the machine is subjected to noise in its body's size, weight, or actuators as well as external forces it demonstrates the ability to dynamically adapt its gait through feedback loops between its actuators and sensors.

## 1 Introduction

Human walking is an elegant solution to bipedal locomotion. It is both resilient to disturbance and efficient. Recently the companies Honda, Sony, and Toyota have all developed their own androids that try to capture these qualities. To walk, these machines use an algorithm based on zero moment point (ZMP) [10] that computes their leg trajectories in order to keep the machines dynamically stable. The result are robots that are robust to disturbance but not particularly efficient. When humans walk they generally step forward on a straight leg and allow the opposing leg to swing past like an inverted pendulum. However these trajectory based machines step down on a bent leg to ensure a dynamically stable gait. The result is a walk that may be inefficient due to the extra power required to support their torso. To explore simpler more efficient models of walking, passive dynamic machines have been built that can walk like inverted pendulums un-powered down small inclines [6]. However, these machines' ability to resist disturbance and balance weight above their hips is limited due to their lack of an active control system. It would seem that humans somehow combine these two ideas to attain a gait that is both robust and efficient. Somehow their motor system keeps their gait stable while leveraging the passive dynamic properties of the body.

In this paper we design a bipedal robot that combines the concepts of dynamic stability and passive dynamics. While we are currently exploring a 3D machine with

12 degrees of freedom, to match the scope and size of this paper we have simplified it to 2 dimensions and 6 degrees of freedom *(Figure 1)*. Our simplified machine was simulated in a physics simulator. An active control system based on neural networks was used to keep the machine stable. Using a genetic algorithm the bodies' and control systems' parameters were evolved to minimise energy consumption while exploiting the passive dynamic properties of the body. The result was a machine that walked both forward and backward. It was dynamically stable and resistant to external noise as well as discrepancies in its body's construction. It took advantage of passive dynamic properties by supporting its torso on a straight leg and using a passive knee joint.

## 2 Previous work

In our previous work we evolved a three-dimensional bipedal robot in simulation that had 10 degrees of freedom but no torso above the hips. The parameters of the body and neural network were encoded in an artificial genome and evolved with a genetic algorithm. The machine started out as a passive dynamic walker on a slope and then over many generations the slope was lowered to a flat surface. The machine demonstrated resistance to disturbance while retaining passive dynamic features such as a passive swing leg. [12].

At MIT the bipedal robot simulation M2 was created with 12 degrees of freedom [7]. It had passive leg swing and used actuators that mimicked tendons and muscles. Its control system was composed of a series of hand written dynamic control algorithms. A genetic algorithm was used to carefully tune the machines parameters.

## 3 Methods

The body of our simulated machine had six-degrees of freedom: one at each hip, one at each knee, and one at each ankle *(Figure 1)*. While the machine was built in a 3D simulator only the $x$ and $y$ planes were explored. To make this possible the machine's legs were allowed to move freely though each other.

The physics of the body were simulated using the open dynamics engine (ODE) physics simulator [11]. Weights and measures were computed in meters and kilograms with gravity set to earth's constant of 9.81 $m/s^2$. The body on average was one meter tall and had 12 parameters *(Figure 1)*: $M_w$ is mass of waist, $M_t$ is mass of thigh, $M_s$ is mass of shank, $M_f$ is mass of foot, $L$ is length of a leg segment, $Y_t$ is offset of thigh mass on the y-axis, $Y_s$ is offset of shank mass on the y-axis. $X_t$ is offset of thigh mass on the x-axis, $X_s$ is offset of shank mass on x-axis, $L_f$ is length of foot, $W$ is radius of waist. The weight of the torso $M_t$ was 1.5 times the weight of $M_w$. $W$ was 1/3 of $L$. $T$ was $2 * L + \frac{W}{2}$. Parameter ranges were selected based on observations of the human body. The mass of the foot was restricted to be less than that of the shank, the mass of the shank was less than that of the thigh, etc. All parameters were encoded in a genome, for optimisation through a genetic algorithm. Feed-forward continuous time neural networks (CTNN) were used to add power to the machine. Unlike traditional neural networks, a CTNN uses time constants to

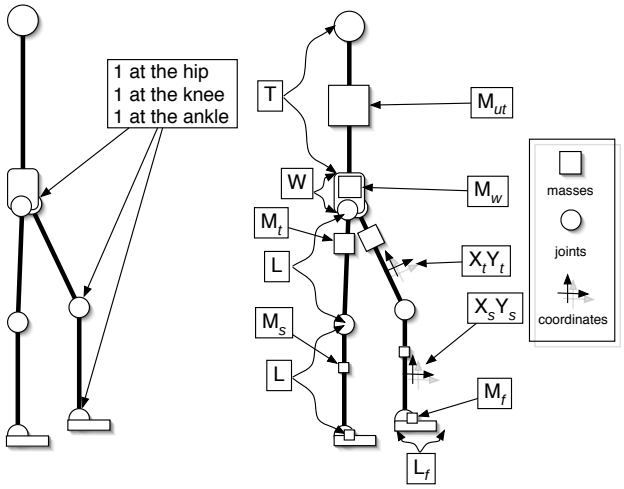


**Fig. 1.** Left: Degrees of freedom in the body. Right: body parameters evolved by the genetic algorithm.

allow neurons to activate in real time and out of phase with each other. For a detailed analysis of this kind of network refer to [1]. The state of a single neuron was computed by the following equation:

$$\tau_i \dot{y_i} = -y_i + \left[\sum_{j=1}^{N} w_{ji}\sigma(g_j(y_j))\right] + I_i + \Omega \tag{1}$$

Where $y$ is the state of each neuron, $\tau$ is a time constant, $w$ is the weight of an incoming connection, $\sigma$ is the sigmoid activation function tanh(), $g$ is the gain, $I$ is an external input, $\Omega$ is a small amount of noise in the range of [-0.0001, 0.0001]. The state of each neuron was integrated with a time step of 0.2 using the Euler method. In our model neurons were encoded in the genome with $\tau$ and $g$ while the axon's weights were encoded with real values in the range of [-5, +5]. Biases were omitted.

Two islands [13] of a geographically distributed genetic algorithm [4] were used each with a population of 50 individuals. The genotype of each individual contained real valued genes. After each mating each gene was mutated by adding a small random number in the range of $[-\frac{m}{2.0}, +\frac{m}{2.0}]$. Initially the mutation rate $m$ was set to 0.5 and then lowered slowly during evolution. Crossover was random. This kind of evolutionary algorithm was used as it has previously proved effective in this context but we do not discount other algorithms being equally effective.

## 4 Network Design

Each leg was given a copy of an identical neural network each of which had two different states either *stance* or *swing*. Upon creation they were connected to each

other by sharing two neurons called centre of mass ($COM$) and *winner (Figure 2)*. To walk one network became *stance* keeping the leg straight and supporting the torso while the other became *swing* and guided the leg either forward or backward. On each step the roles of the networks were swapped. Each network decided its state by using a winner take all circuit *(Figure 2)*. The leg with the most foot pressure became the stance leg and inhibited the other to become the swing leg. When the stance leg's foot later lost contact with the ground the other leg became the stance leg and inhibited the first. The *stance* state was implemented making a positive connection between a gyroscope that detected the orientation of the waist around the $x$ axis and the desired hip velocity (a) in *(Figure 2)*. If the torso fell forward the leg moved under it by powering the hip. An Inhibitory connection between the winner take all circuit inhibited this behaviour when the network was not in stance state (b).

When a network was in *swing* state it needed to move the leg forward just enough to catch the machine's centre of mass (COM). This is a similar problem to that of a Segway human transporter [9]. A Segway has only two wheels but must support a person standing on top by computing how fast it must turn its wheels to drive under its COM. A simple algorithm to do this is to attach a gyroscope and wheel encoder to the machine and to set up a simple feedback loop to the wheel motors. If the sum of the gyroscope angle and its derivative added to the sum of the wheel angle and its derivative is connected to the wheel motor the machine will automatically balance itself. In our machine the legs take the place of the wheels so we compute our COM by summing our gyroscope and current hip angle with the velocity of the torso. Velocity can be computed by a function of the derivative of the hip angle, the length of the leg, the derivative of the gyroscope and the length of the torso. However, for simplicity we have taken the velocity directly from the physics simulator (c). The hip torque was computed by taking the derivative of the hip angle and subtracting the desired hip velocity (e). To move the leg negative connections were made from the leg's hip angle sensor and the other leg's COM to the hip actuator's desired velocity (f). This allowed the leg to move to an equilibrium point just in time to catch its mass falling forward. Similar to the stance leg this behaviour was inhibited when not in *swing* state (g). To keep the knee straight when the foot was touching the ground a positive connection was made from the foot's pressure sensor to the knee velocity (h). When the foot was off the ground, the knee was completely un-actuated and allowed to passively swing. Knee torque was set to the magnitude of its velocity. Ankles were implemented by placing a negative connection between their angle sensors and desired velocity. Ankle torque was automatically set to the magnitude of their desired velocity making the ankles act as damped spring (i). To inject power into the stride when the COM was forward a positive connection was made from the COM to the ankle velocity (j).

### 4.1 Stepping Reflex

Two reflexes were used to automatically lift the swing leg forward or backward if the COM shifted past a threshold. Without these the machine could support the torso and guide the swing leg but had no way of initially getting a foot off the ground. Each foot was given a recharge counter and four neurons were added to each network: $forward$, *backward*, *decay*, and *strength* (k). If the foot was fully charged and the COM shifted over a threshold, the forward or backward neurons

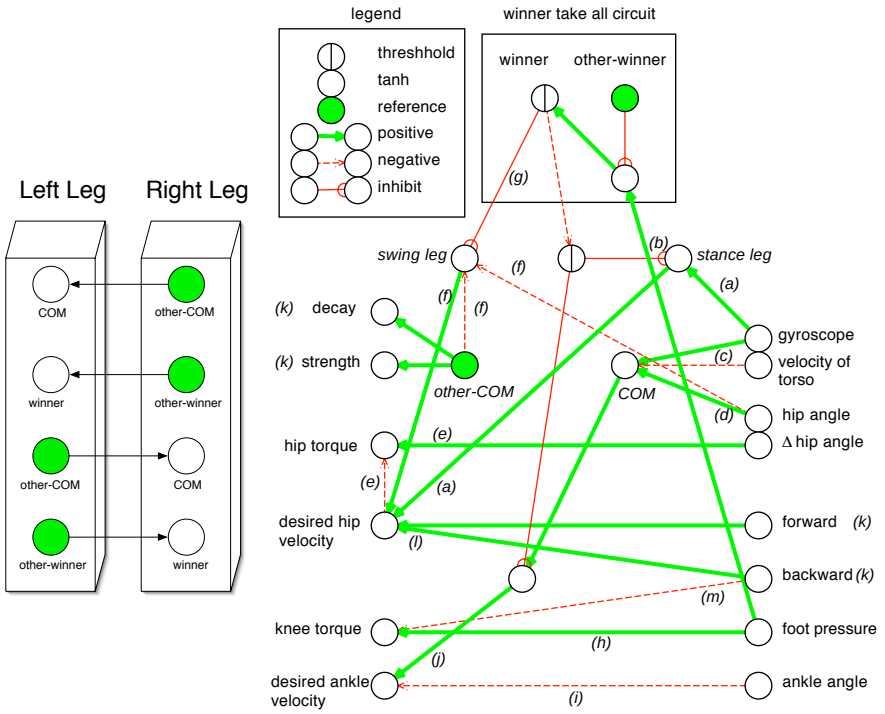


**Fig. 2.** Left: Each network is identical and communicates through shared neurons. Solid circles are actual neurons, transparent ones are references. Right: network structure. Inhibitory connections push the activation toward zero regardless of sign. Actuators had two inputs desired velocity and maximum torque available to achieve that velocity. If torque was not specified it became the magnitude of the desired velocity.

were pulsed lifting the leg and reseting the recharge counter. The rate of decay and strength of the pulse was taken from the rate and strength neurons. Each of these neurons were connected to the COM allowing the machine to take larger steps depending on how far off balance it was. The greater the magnitude of the rate the slower the decay of the pulse over time. The greater the magnitude of strength the larger the magnitude of the initial pulse. To step forward or backward a positive connection was made between the *forward* and *backward* neurons and the desired hip velocity (l). In the forward case this allowed the hip to lift and the knee to swing passively forward. In the backward case an additional negative connection was made between the *backward* neuron and the knee torque(m). This momentarily compressed the knee enough to allow it to passively swing past the stance leg.

# 5 Experiments

## 5.1 Dynamic gait adjustment

In connected ballroom dances such as tango the lead communicates intended movements to the follower through a dance frame. When the lead moves forward they gently push into the follower causing them to step backward. The more the lead pushes the faster the follower moves. In our first experiment we use this idea to teach our machine to walk at different speeds both forward and backward. A population of machines were constructed and each one was evaluated for fitness in the following way:
1. Place the machine standing upright with legs together on a flat surface.
2. Constantly push it forward with just enough force to get it to reach a desired velocity chosen at random. Compute its fitness with the the following function:

$$fitness = t\left(\frac{1}{1+p}\right)\left(\frac{1}{1+z}\right)\left(\frac{1}{1+x}\right)\left(\frac{1}{1+f}\right) \tag{2}$$

Where: $t$ is time the machine walked before falling, $p$ is the torque used, $z$ is the acceleration of the hip along the z dimension, and $x$ is the rotation of the hip around the x-axis. $f$ is the force required to get the machine to the desired velocity. $p$, $z$, $f$, and $x$ were averages taken over the entire evaluation time.
3. Put the machine back to the starting position and push it backward with just enough force to reach a desired velocity chosen at random. Compute its fitness again.
4. Take the worse fitness of the two runs.
5. Repeat step (1-4) nine more times and take the average fitness.
This type of evaluation ensures a machine walks equally well forward or backward with the minimum amount of force pushing on it. It also selects for machines that walk as long and straight as possible without explicitly specifying how they move their legs. This allows their leg trajectories to emerge from the dynamics of their bodies rather than from the observations of a human gait. The machine's gait after 800 generations is illustrated in *Figure 3*.

To determine how closely our machine was able to match a desired velocity two graphs were made. One comparing the machine's desired velocity with its actual velocity *Figure 4* and one showing its change in foot timing at several different velocities. The first graph shows a valley. As the machine is pushed either forward or backward its velocity increases or decreases respectively. If the machine is close to zero it attempts to stand still.

## 5.2 Robustness to noise

One of the great difficulties with computer simulation is that it often fails to transfer to a physical robot due to unforeseen differences found in the real world. Therefore a controller that is adaptable to unforeseen changes may have a better chance of making the transfer. To determine how adaptable our machine was we subjected it to both internal and external noise. Internal noise was defined as errors in the body itself such as incorrect body masses, leg lengths, or noisy actuators. External errors were defined as random external forces that attempt to push the machine off

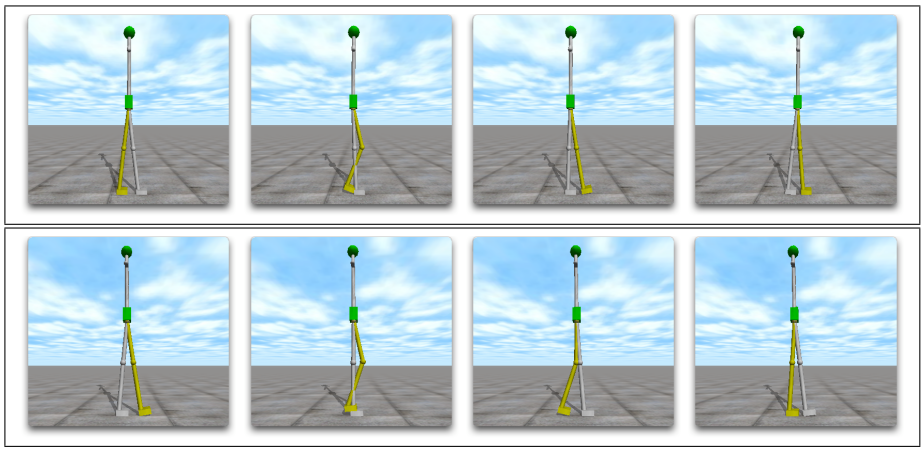

**Fig. 3.** Top: gait of machine walking forward. Bottom: gait of machine walking backward.

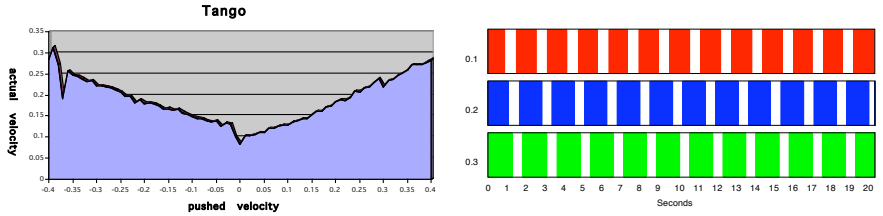


**Fig. 4.** Left: Comparison of the machine's desired velocity with its actual velocity. The $x$ axis is the desired velocity forward (positive) or backward (negative) and the $y$ axis is the absolute value of the machine's actual velocity. Right: The foot strike timing of the left leg at three different speeds 0.1, 0.2, and 0.3 m/s. Solid bars indicate the foot is touching the ground while transparent indicate the foot has left the ground.

balance. Internal noise was introduced by adding an error to each body parameter and actuator upon construction (3).

$$p = p + p * (rand() - 0.5) * 2.0 * e \qquad ae = (1 - rand()) * e \tag{3}$$

Where: $p$ is the body parameter, $rand()$ is a function that returns a random number in the range of 0 and 1, $e$ is the percentage of error. Errors were introduced to actuators by multiplying the desired velocity and torque from the network by $ae$ and adding random noise in the range of $[-\frac{e}{10}, +\frac{e}{10}]$ on each time step. $ae$ and $p$ were computed just once on the construction of the machine. External noise was added by applying a random force along the x and z axis (4) each time step.

$$x = (rand() - 0.5) * 0.5 * e \qquad z = (rand() - 0.5) * 0.5 * e \tag{4}$$

The machine was tested for the average number of forward steps it could take over 20 trials when pushed forward at 0.2 m/s for error rates between 0 and 50% *(Figure 5)*. The graph shows a graceful degradation in the number of steps taken as noise increases. With 10% noise the machine is able to take 93 steps on average and after 50% noise the machine can still take an average 10 steps.

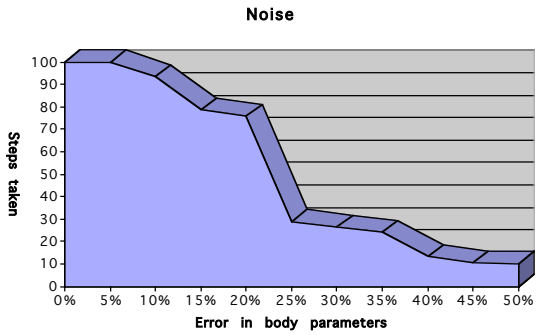


**Fig. 5.** Graph illustrating robustness to error. The $y$ axis is the average number of steps taken over 20 trials while walking forward at a velocity of 0.2 m/s. $x$ is the percentage of error $e$. The number of steps taken was capped at 100.

### 5.3 Efficiency

In studies conducted in Kenya women there were observed to carry great weights on their heads while using very little energy [2]. This was attributed to the observation that they walk like inverted pendulums supporting the weight on a straight leg as they move forward. To study the efficiency of our system the population of machines was evolved for an additional 200 generations and evaluated for their ability to walk forward while carrying varying amounts of weight. Initially the body weighed 42 kg, 32 kg of which were contributed by the torso. In the experiment 20 walks were conducted increasing the torso's weight by a percentage of the total body mass. At 0% the torso was unchanged at 32 kg, at 200% *(Figure 6)* the torso was $32 + 42 * 2 = 116kg$. To start each walk the machine was pushed to a velocity of 0.25 m/s and its energy consumption was computed as it walked 60 meters. At 0% it used 193 kg-m of torque and at 200% it required 292 kg-m. This revealed that to carry twice its body weight (200%) this machine required only 51% more energy *(Figure 6)*.

### 5.4 Construction of a physical machine

To transfer our simulation to a physical machine we are currently developing actuators based on series elastic actuators developed by MIT [8]. These devices combine

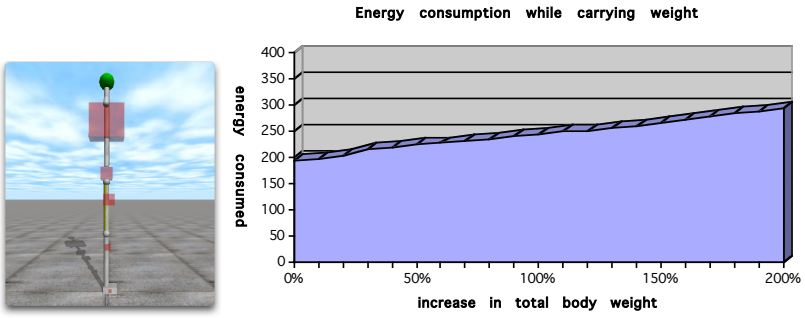


**Fig. 6.** Left: The machine carrying 200% of its original weight (Boxes indicate mass distribution). Right: Graph illustrating its ability to carry weight. The $y$ axis is the total torque required in kg-m and the $x$ axis is the weight carried in kg.

a worm gear with a spring to make a device that has shock tolerance, high fidelity and low impedance. By setting up a negative feedback loop between spring deflection and a desired deflection they can be used to apply varying amounts of force. If the desired deflection is zero they can emulate passive components. When this is combined with a proportional-derivative (PD) controller whose input is a desired velocity and maximum torque, it can implement the type of actuators used in our simulation. Our primary concern is issues involving unforeseen differences between simulation and reality. While our experiments have demonstrated resistance to errors in the body and actuators we also have other techniques that can aid in the transfer. Jakobi successfully transfered a controller for a simulated khepera robot to a physical one by carefully adding noise during evolution [5]. Plastic weight updating rules have also been used to make the transfer to physical machines. In one experiment, [3] evolved a controller that allowed a simulated Khepera robot to navigate a maze and then transfered it to a physical one. To demonstrate how adaptive plasticity could be they then transferred the same controller to a different six wheeled Koala robot. In a second experiment they evolved a four legged walking robot in simulation and successfully transfered it to a physical machine. While our physical robot is only in its early stages we are taking appropriate measures to ensure our machine will transfer successfully.

## 6 Conclusion

We have demonstrated a simulated machine that combined dynamic stability with passive dynamics. Its control system was composed of two identical neural networks that formed a dynamic system whose basin of attraction was walking. When pushed forward or backward it walked just enough to support its centre of gravity. By using passive knee swing and stepping down on a straight leg it demonstrated the ability to support large weights efficiently. The machine maintained stability even when subjected to noise such as external forces, body parameter errors, and actuator errors. This is an interesting result since the CTNNs of our model do not store

information through weight changes, as many conventional artificial neural networks do. Instead it had to rely entirely on the feedback between its sensors and actuators. This adaptability may provide a mechanism for transferring simulated control systems to physical robots.

This technique is very powerful and we are currently using it to explore more complex 3 dimensional bipedal machines. Some of these simulated machines have demonstrated the ability to dynamically run. We are now beginning to build a physical android based on this model and hope to discover further insights into how to use these methods to develop practical bipedal machines. Videos of our simulated machines can be found at (www.droidlogic.com).

## References

1. R. D. Beer. Toward the evolution of dynamical neural networks for minimally cognitive behavior. In P. Maes, M. Mataric, J. Meyer, J. Pollack, and S. Wilson, editors, *From animals to animats 4: Proc. 4th International Conf. on Simulation of Adaptive Behavior*, pages 421–429. MIT Press, 1996.
2. S. Eugenie. How to walk like a pendulum. *New Scientist*, 13, 2001.
3. D. Floreano and J. Urzelai. Evolutionary robots with on-line self-organization and behavioral fitness. *Neural Networks*, 13:431–443, 2000.
4. P. Husbands. Distributed coevolutionary genetic algorithms for multi-criteria and multi-constraint optimisation. In T. Fogarty, editor, *Evolutionary Computing, AISB Workshop Selected Papers*, volume 865 (LNCS), pages 150–165. Springer-Verlag, 1994.
5. N. Jakobi, P. Husbands, and I. Harvey. Noise and the reality gap: The use of simulation in evolutionary robotics. *ECAL*, pages 704–720, 1995.
6. T. McGeer. Passive walking with knees. In *Proceedings of the IEEE Conference on Robotics and Automation*, volume 2, pages 1640–1645, 1990.
7. J. Pratt and G. Pratt. Exploiting natural dynamics in the control of a 3d bipedal walking simulation. In *Proceedings of the International Conference on Climbing and Walking Robots (CLAWAR99), Portsmouth, UK*, 1999.
8. D. W. Robinson, J. E. Pratt, D. J. Paluska, and G. A. Pratt. Serieselastic actuator development for a biomimetic robot. *IEEE/ASME International Conference on Advance Intelligent Mechantronics.*, 1999.
9. segway. Segway models. www.segway.com, 2004.
10. C. Shin. Analysis of the dynamics of a biped robot with seven degrees of freedom. In *Proc. 1996 IEEE International Conf. of Robotics and Automation*, pages 3008–3013, 1996.
11. R. Smith. The open dynamics engine user guide. http://opende.sourceforge.net/, 2003.
12. E. Vaughan, E. Di Paolo, and I. Harvey. The evolution of control and adaptation in a 3d powered passive dynamic walker. In *Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems, ALIFE'9 Boston*. MIT Press, September 12th-15th, 2004 (http://www.droidlogic.com/sussex/papers.html).
13. D. Whitley, S. Rana, and R. Heckendorn. The island model genetic algorithm: On reparability, population size and convergence. *Journal of Computing and Information Technology*, 7:33–47, 1999.