Rigid and Soft Body Simulation Featuring Realistic Walk Behaviour

Oliver Urbann, Sören Kerner, and Stefan Tasse

Robotics Research Institute
Section Information Technology
TU Dortmund University
44221 Dortmund, Germany

Abstract. Using a simulation for development and research of robot motions, especially walking motions, has advantages like saving real hardware, being able to replay specific situations or logging various data. Unfortunately research in this area using a simulation depends on transferability of the results to reality, which is not given for common robotic simulators. This paper presents extensions to a basic rigid body physics simulation leading to more realism. Parametrization matching a particular real robot is done using Evolutionary Strategies. Using stable walking and kicking motions as reference for the ES the newly developed MoToFlex simulator is able to reflect typical walking issues which can be observed in reality using different walking motions.

1 Motivation

Humanoid robots are an important research area due to the higher acceptance by human beings and due to the their advantage of being able to operate in environments designed for them [7]. To grant the usability of biped robots in service robotics or the health care sector a stable walk is vital. Many researchers work on this field, not only on dynamic walking based on the Zero Moment Point (ZMP) [19], but on problems such as pushing or carrying objects [15], dealing with uneven surfaces [6] or the need for a fast modification of foot placement [14]. Currently existing biped robots of human size, e.g. the HRP-4C developed by the AIST Institute in Japan [8] or the Honda Asimo [5], are too expensive to be used widely. More favorable robots like the Nao by Aldebaran Robotics [1] reveal significant inaccuracies in joint movements resulting in an unstable walk even of theoretically correct walking motions. Authors dealing with these issues are Kim et. al [9], Meriçli et. al [12] and Gouaillier et. al [2]. Studying the occurring errors during the walk by means of a real robot has some disadvantages like wearing out gears, noisy and delayed sensor data, not replicable reactions and time-consuming experiments. Therefore working with a simulation has advantages, but because of the big gap between simulations and reality adapting a walk developed using a simulation to a real robot is a difficult task.

[1] `http://www.aldebaran-robotics.com/`

Thus the goal of this work is to downsize the difference between a simulation and reality. A simulation, called MoToFlex simulator [2] , that reflects some typical problems of a real walking Nao robot is proposed here.

The next chapter describes the fundamentals of humanoid robot simulations and related work. In section 3 the basic simulation is extended by a motor model, flexible gears with tolerance and flexible bodies. An appropriate parametrization of the simulation is crucial to reach the goal of realistic dynamical and kinematical behaviour. This is described in section 4. Finally section 5 displays that the objective has been fulfilled.

2 Related Work

The base of a simulation for humanoid robots is a rigid body physics engine, like the Open Dynamics Engine (ODE) [18]. Physics simulations implementing such an engine, like SimRobot [10], Webots [13] or SimSpark[16], allow to simulate the kinematics and dynamics of rigid bodies, represented by their center of mass and inertia tensor. This includes joints, collision detection and dealing with friction in case of contact.

To interact with the simulated model torques and forces can be applied. For example, it is possible to set the torque T_j which is applied by a joint to the connected bodies. Another common way to modify the joint angles is to directly set the desired joint speed ω_j along with a maximum torque T_{max} which can be applied to reach this speed. The torque T_j and the desired joint speed ω_j can be set simultaneously. This is useful for simulating friction when a torque T_j is applied by the joints. To do so, the desired speed of the joint is set to 0 ($\omega_j = 0$). The chosen maximum torque T_{max} is the friction torque added by the ODE to T_j.

A common way to control the joints of a robot is to set the desired joint angles. To set ω_j the joint angle positions can be differentiated. The resulting angle speeds are executed as given, resulting in a walk close to the expected one. From this it follows that a simulation based on a rigid body physics engine has to be extended by elements that imitate the real pipeline from the desired joint angles to the reached positions of the bodies. Hein et al. extended the ODE model of a robot by elements simulating the gear tolerance [4]. This extension leads to an unstable simulation and is therefore not useful. Lima et al. [11] implemented a simulation of the motors including PID controller. This is also one important element in the MoToFlex simulator but more elements will be needed to simulate flexibility and tolerance. The following section illustrates the extensions of the ODE based MoToFlex simulator and starts with an explanation how this motor simulation is adapted.

3 Development / Elements of Simulation

The previous chapters point out that the development of a walking algorithm starts using a simulation, but its usefulness is limited since common walking

[2] `http://www.irf.tu-dortmund.de/nao-devils/download/2011/MoToFlex.zip`

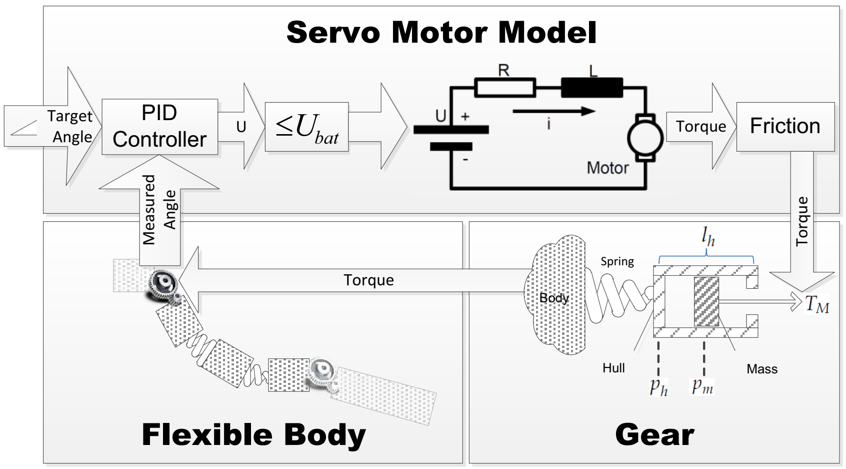


Fig. 1. Pipeline of the simulation

issues are not reflected by ordinary robotic simulations. In this chapter new extensions, arranged as a pipeline, are presented that are able to make simulations more realistic. They are implemented in a newly developed ODE based simulation, the MoToFlex simulator. Figure 1 shows the pipeline. Input of the system is the desired angle position. The pipeline consists of three phases. The first two stages, the servo motor and gear simulation, are independent from the ODE. The third stage is realized as an ODE model.

3.1 Motors

Algorithm 1 explains the servo motor simulation which bases on the work of Lima et al., see [11] for details. A PID controller uses the desired joint angles q_T and the actual joint angles q_A to calculate the voltage U, which is limited by the voltage U_{bat} of the battery. Due to the inductivity and resistance of the motor it has to be modeled as an RL circuit, see figure 1. Using the values given by the manufacturer for the equivalent series resistance R and inductor L the voltage U_r at the resistor R is close to U after only 1 ms. For the sake of simplicity U_r is therefore calculated iteratively without significant errors.

Hereafter the torque can be determined by multiplying U_r with $\frac{S \cdot K_t}{R}$ where S is the stiffness factor of the Nao joints [3], R the value of the resistor and K_t the torque constant of the motor. The latter factors are given by the manufacturer of the motor. An unknown quantity is the back electromotive force (EMF) constant K_e, which connects the motor speed with the back EMF voltage U_e induced by the motor: $U_e = K_e \cdot w_m$. To reduce the amount of variables with unknown

[3] By setting the stiffness parameter $S \in [0, 1]$ it is possible to reduce the torque applied by the motor.

Algorithm 1. Algorithm to simulate joint motors

```
1: $U = PID\left(q_T, q_A\right)$
2: if $U > U_{bat}$ then
3:     $U = U_{bat}$
4: end if
5: $U_R\left(t + \Delta t\right) = U_R\left(t\right) + \left(U - U_R\left(t\right)\right) \cdot \left(1 - e^{-\frac{t}{L/R}}\right)$
6: $T_M = \underbrace{\frac{U_R}{R} \cdot S \cdot K_t}_{\text{Torque due to power}} \underbrace{- \left(B_v + K_e\right) \cdot \omega_M}_{\text{Torque due to friction and back EMF}}$
```

values the back EMF is treated as a speed dependent friction since both reduce the speed depending only on the joint/motor speed.

Besides the back EMF two types of friction have to be taken into account, the already mentioned speed dependent friction B_v and the deceleration by the EMF on the one hand and the friction not depending on the motor speed on the other. The former are combined to one friction parameter and multiplied with the motor speed. The later is handled by the ODE, as described in section 2.

3.2 Gears

Tolerances are a wanted property of gears to compensate production inaccuracy and expansion due to warming. Besides the tolerances there can also be a flexibility caused by the used materials. Hein et. al attempted to reproduce tolerance within their ODE based simulation which thereby got unstable. Therefore this is handled outside the ODE here. A model of gears is developed consisting of two elements: a spring to simulate the flexibility and a hull containing a mass to simulate the tolerance. This model is realized as an algorithm which calculates the outcome of flexibility and tolerance before sending the torque to the ODE. This has the advantage, that there is no need to change the ODE model of the robot to simulate it, resulting in more stable physics simulation.

Algorithm 2 depicts the proposed method. See table 1 for a description of the used variables. A linear model is used which is equivalent to a rotational model but more illustrative. The conversion between torque/angle to force/distance is done with a radius of 1. First we apply the gear transmission ratio. To achieve a stable simulation the algorithm for simulating the tolerance and flexibility has to work with a high frequency. To speed up the overall simulation the ODE works with a lower frequency. The increase of the gear simulation frequency is done by the loop in line 2 of algorithm 2.

The torque T_M exerted by the motor accelerates a small mass, which can be imagined as the mass of the gears next to the motor. In case of contact with the hull the torque exerted by the spring also accelerates the mass. For a realistic and stable simulation a speed dependent friction is also required. In position $p_h = 0$ the spring is not stretched by definition and no torque is exerted by the spring. Since the hull has no mass it has no contact to the hull if and only if the mass is located between 0 and l. Otherwise the hull and the mass are in

Algorithm 2. Algorithm for simulating gears with flexibility and tolerance

```
1: $T'_M = T_M \cdot \sigma$
2: for $i = 0$ to $\frac{\Delta t_s}{\Delta t_g}$ do
3:     $p_h = 0$
4:     if $p_m > l_h$ then {The mass is in contact with the hull.}
5:         $p_h = p_m - l_h$
6:     end if
7:     if $p_m < 0$ then {The mass is in contact with the hull on the other side.}
8:         $p_h = p_m$
9:     end if
10:    $T_s = D_g \cdot p_h$ {Calculate the torque exerted by the spring.}
11:    $p_m = p_m + \dot{p}_m \cdot \Delta t_g$ {Integrate the speed of the mass to get the position.}
12:    $\dot{p}_m = \dot{p}_m + \frac{(T'_M - T_s - B_g \cdot \dot{p}_m)}{m_g} \cdot \Delta t_g$ {Integrate the acceleration (exerted torque
       divided by the mass) to get the speed.}
13:    $T_j = T_j + T_s \cdot \frac{\Delta t_g}{\Delta t_s}$
14: end for
```

Table 1. Variables and constants

p_h	Position of upper border of the hull.
p_m	Position of mass.
$\dot{p}_m$	Speed of mass.
D_g	Spring constant.
B_g	Coefficient of viscous friction.
Δt_s	Length of a time step of the ODE physic simulation.
Δt_g	Length of a time step with $\Delta t_s \geq \Delta t_g$.
l_h	Length of the hull.
T'_M	Torque excerted by the motor after gear ratio is applied.
T_j	Torque output to be applied by the joint on the bodies.
σ	Gear ratio.
T_s	Torque exerted by the spring on the body.
T_f	Dry friction torque handled by the ODE.
m_g	Weight of the mass.

contact and have the same speed. If, for example, the mass starts between 0 and l and moves towards the springs it gets in contact with the hull at position 0. It then compresses the spring and even in case of an abrupt change in the moving direction of the mass the hull remains in contact with the mass. Once the position of the hull is determined the torque T_s exerted by the spring can be calculated and is then added up to the output torque of the gears T_j (see section 2).

3.3 Flexible Bodies

This torque T_j can not only accelerate the bodies connected by the joint, it can also cause a deformation of the bodies, depending on their rigidity. Some robots can measure the angle errors of the gears since they have sensors measuring the angle between the two bodies, not only the angle of the motor. The Nao is an example for such a robot. Therefore errors due to flexible bodies have to be handled separately. As figure 1 indicates, a flexible body is realized here as smaller bodies connected by springs. This is known as the mass-spring model to realize soft body dynamics. In fact these springs are ODE double joints where a torque T_B is set:

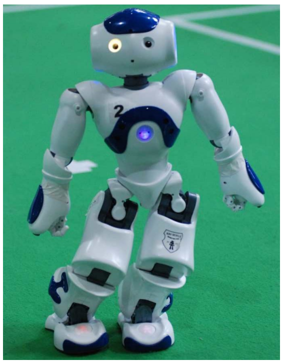
$$T_B = -D_B \cdot \alpha \tag{1}$$

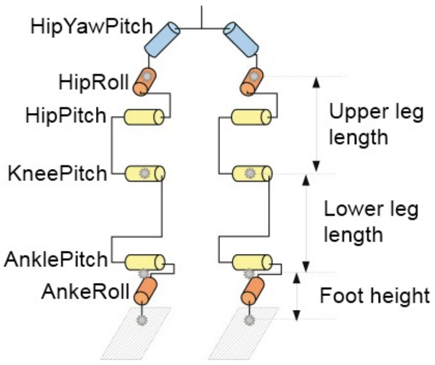
where D_B is the spring constant and α the actual angle of the axis. To dampen the spring a Coulomb friction is incorporated which is realized by the ODE itself by using the method mentioned above.

3.4 Nao Setup

The most considerable goal of the simulation, besides a realistic simulation, is computational stability and efficiency. Therefore not all bodies are flexible, but only those whose flexibility has the greatest effect on walking motions, i.e. upper and lower legs (see figure 2). Positions of the centers of mass, sizes and masses of the boxes are chosen according to the official Nao documentation (Version 1.0). Missing dimensions were measured by hand.

The Nao has two joints with 3 axes, the hip joints, consisting of two perpendicular axes and a third axis crossing the intersection of the other, see figure 2. The ODE lacks joints with more than two axes. Implementations by connecting a single axis joint to the double axis hip using a body of size and mass 0 would result in unstable simulations. Therefore the HipYawPitch axis is not implemented so far which results in the inability to walk curves but ensures higher computational stability.


(a) Nao during soccer game.



(b) Schematic view.

Fig. 2. Nao by Aldebaran Robotics

As the Nao documentation clarifies two gear types are used in the legs. Hence two different sets of gear parameters are used. Most of the parameters are unknown, only gear ratios are given by the manufacturer. Therefore an optimization method is needed to find parameters such that the simulated robot walks like a real one.

4 Parametrization

Besides the unknown ODE parameters like friction the extensions presented in the previous chapter also contain some variables with unknown values. To find reasonable values a black box optimization method is applied. Evolutionary Algorithms, in particular the Evolution Strategies proposed by Schwefel et al. [17] evinced good results in terms of optimization in the robotic domain. Hebbel et al. utilizes Evolution Strategies to find simulation parameters that minimizes the difference between the way walked by a four legged robot in the simulation and the way walked in reality [3]. Here they are used to minimize the difference between the measured joint angles in reality and the measured joint angles in the simulation to achieve an overall realistic kinematic and dynamic behaviour of the simulated Nao. To get more generalized simulation parameters there are several robots in an evaluation of one parameter set, namely four walking robots using different walking parameters and one kicking robot. The walking motions are generated using the walking control developed by Czarnetzki et al. [1]. This leads to the following fitness function, which has to be minimized:

$$F\left(q_r, q_s\right) = \frac{\sum_{t=1}^{T} \sum_{i=1}^{n} \left(q_{r,i}\left(t\right) - q_{s,i}\left(t\right)\right)^2}{T^m} \tag{2}$$

where:

– $q_{r,i}\left(t\right)$ are the measured angles of the real robot i
– $q_{s,i}\left(t\right)$ are the measured angles of the simulated robot i
– T is the duration of the simulation in frames (the simulation ends when a maximum frame number is reached or the robot has fallen down)
– n is the number of robots in the simulation.

The exponent m has to be discussed. Assuming the fitness function would comprise the sum of the quadratic angle difference only ($m = 0$). This would lead to an immediate fall right at the beginning of the simulation since this minimizes equation (2) with $m = 0$. Dividing the sum by T ($m = 1$) gives the mean angle error which leads to a falling robot when it starts walking since then the average angle error would increase. It turns out that choosing $m > 1$ leads to a stable walk where after the quadratic angle difference is further minimized.

For the parameter optimization different evolution strategies are compared. Using a cluster of 200 nodes for the evaluation a large number of children can be used. A $(10 + 800)$-ES with a maximum lifetime of 3 generations performed best and makes it possible to find good parameters after only ca. 100 generations. Using a simulation time step length of $0.001s$ and a maximum simulated walk duration of $10s$ this can be done within one day. The found set of parameters is optimized for a particular robot. Even though the experiments in the next section compares the simulation with this robot to show that it leads to a more realistic simulation, other Naos show very similar walking problems.

5 Experiments

In the context of walk development a useful simulation is able to reflect common walk issues. In this chapter the usefulness of the simulation is shown by presenting typical walk issues using real Naos compared to occurring walk issues in the MoToFlex simulation. Here again the walking control by Czarnetzki et al. generates the walking motions without sensor feedback. Two different setups are used.

In the first setup the walking motions are generated at a target speed of $200\frac{mm}{s}$ and a step duration of $1s$. In reality the orientation of the body is measured using the gyroscope and accelerometer of the robot. In the simulation the orientation of the body is directly given by the ODE. The speed is measured by walking a predefined distance. Using this setup we encounter an oscillation during the walk along the x axis[4]. This results not only in an unstable walk but also causes the swing leg to touch the ground while the body is rotated to the back. This results in a higher step length as desired and thus a higher walking speed is measurable. Figure 3 shows the pitch of the body measured on the real robot using the gyroscope and accelerometer. Compared to the simulation the amplitude is larger but frequency and phase are the same. Apart from that it is also noticeable that every second oscillation is further forward and backward respectively which occurs in reality as well as in the simulation. For comparison the orientation of the body in the simulation without any extensions (ODE only) is also shown. Besides the orientation the increased walking speed appears in reality as well as in the MoToFlex simulator. While the real robot walks at $250\frac{mm}{s}$ when a target speed of $200\frac{mm}{s}$ is set, the simulated one reaches $209\frac{mm}{s}$. Turning off the extensions leads to a walking speed of $200\frac{mm}{s}$.

In the second setup the target speed is set to $50\frac{mm}{s}$ at a step duration of $2s$. Disturbances along the y axis appear mainly at this step duration. This is caused by the joints of the standing leg failing under the strain in the single support phase causing the body to rotate towards the swinging leg. The manufacturer of the Nao, Aldebaran Robotics, also describes this problem and explains it with a sudden change of the desired torque during a single support phase [2]. In consequence the swing leg erroneously touches the ground whereby the robot is pushed to the wrong side. This leads to an overlarge lateral oscillation and a falling robot after the first step. Figure 4 depicts the roll angle of the real robot compared to the simulated one. Here again, the orientation differs quantitatively, but the body inclines to the left at the same time, and after the erroneous contact to the right. Moreover the robot falls down at a similar point of time. The problem is resolvable by changing the reference ZMP. It is worth mentioning

[4] The x axis is defined here as the frontal axis and the y axis as the lateral axis.

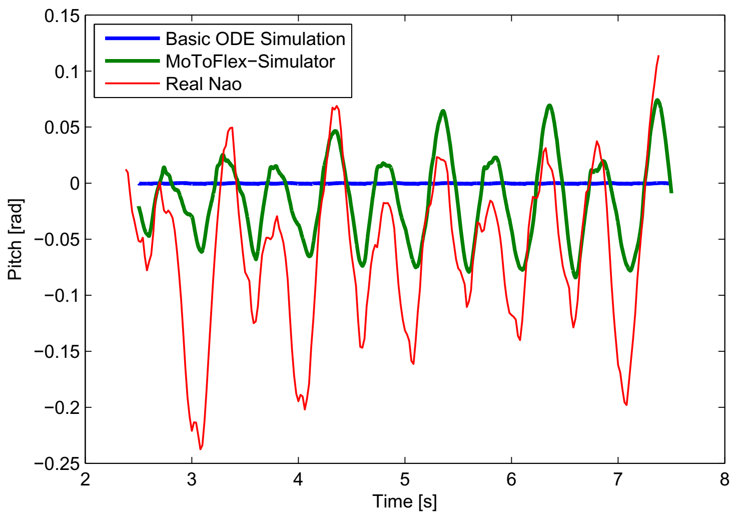


Fig. 3. Pitch of the Nao during a walk at a target speed of $200\frac{mm}{s}$ and step duration of $1s$

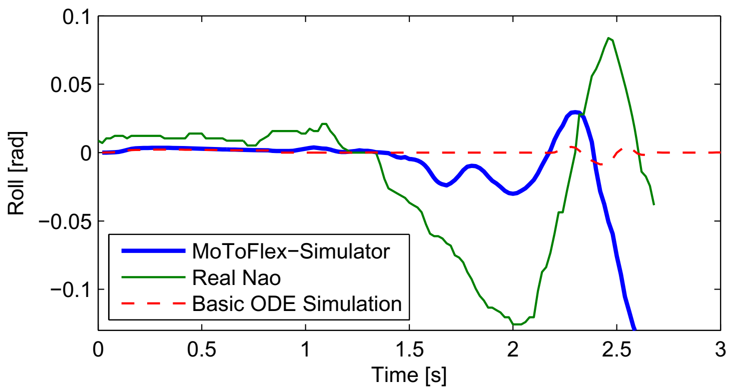


Fig. 4. Roll angle of the Nao during a walk at a target speed of $50\frac{mm}{s}$ and step duration of $2s$

that the tested modified reference ZMPs lead to a stable walk in the simulation if and only if they lead to a stable walk in reality.

6 Conclusion

In this paper, three extensions for a simulation based on the Open Dynamics Engine are proposed: a model of a servo motor including a PID controller, flexible gears with tolerance and flexible bodies used for the legs. Using Evolution

Strategies appropriate parameters for the simulation can be found. Using this parameter set the MoToFlex simulator reflects some common walk issues of a real robot enabling to develop walking motions in a more realistic simulation. Nevertheless the simulation accuracy might be improved to reduce the difference between the simulated and real posture. To do so the parameter optimization could be done by evaluating with more than 5 simulated robots. Beside four walking and one kicking robots other motions like standing up or simple leg movements could be useful.

An open question is the reason for the walking issues. Some of the simulated elements has the same effect than other. E.g. the simulation of a PID controller simulates with its proportional and differential part a spring with a damper, which is very similar to the simulation of the flexible gears. In fact the parameters vary from optimization to optimization. E.g. some have a higher proportional value with lower spring constant. Another interesting fact is, that the simulation of the flexible boxes only (ODE without motor and gear simulation) can lead to an unstable walk using the setup of the second experiment. Therefore it is difficult to use the simulation to find the parts of the real robot that are responsible for the walking issues. Despite this, the simulation can be used to develop algorithms dealing with the inaccuracies. For developing closed looped walking algorithms simulated sensors need to be integrated into the simulation. Then also typical difficulties of real measurements can be simulated, like sensor noise and different measurement delays.

References

1. Czarnetzki, S., Kerner, S., Urbann, O.: Applying Dynamic Walking Control for Biped Robots. In: Baltes, J., Lagoudakis, M.G., Naruse, T., Ghidary, S.S. (eds.) RoboCup 2009. LNCS (LNAI), vol. 5949, pp. 69–80. Springer, Heidelberg (2010)
2. Gouaillier, D., Collette, C., Kilner, C.: Omni-directional closed-loop walk for Nao. In: 2010 10th IEEE-RAS International Conference on Humanoid Robots (Humanoids), pp. 448–454 (December 2010)
3. Hebbel, M., Nistico, W., Fisseler, D.: Learning in a High Dimensional Space: Fast Omnidirectional Quadrupedal Locomotion. In: Lakemeyer, G., Sklar, E., Sorrenti, D.G., Takahashi, T. (eds.) RoboCup 2006: Robot Soccer World Cup X. LNCS (LNAI), vol. 4434, pp. 314–321. Springer, Heidelberg (2007)
4. Hein, D., Hild, M.: Simloid - research on biped robots controller, using physical simulation and machine learning algorithms. In: Concurrency, Specification and Programming, CSP 2006. Informatik-Berichte, vol. 1, pp. 143–151. (2006)
5. Hirai, K., Hirose, M., Haikawa, Y., Takenaka, T.: The development of Honda humanoid robot. In: ICRA, pp. 1321–1326 (1998)
6. Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Yokoi, K., Hirukawa, H.: Biped walking pattern generator allowing auxiliary ZMP control. In: IROS, pp. 2993–2999. IEEE (2006)
7. Kanda, T., Ishiguro, H., Imai, M., Ono, T.: Development and evaluation of interactive humanoid robots. Proceedings of the IEEE 92(11), 1839–1850 (2004)
8. Kaneko, K., Kanehiro, F., Morisawa, M., Miura, K., Nakaoka, S., Kajita, S.: Cybernetic human HRP-4C. In: 9th IEEE-RAS International Conference on Humanoid Robots, 2009, Humanoids, pp. 7–14 (December 2009)

9. Kim, J.-Y., Park, I.-W., Oh, J.-H.: Experimental realization of dynamic walking of biped humanoid robot KHR-2 using ZMP feedback and inertial measurement. Advanced Robotics 20(6), 707–736 (2006)
10. Laue, T., Spiess, K., Röfer, T.: SimRobot – A General Physical Robot Simulator and Its Application in RoboCup. In: Bredenfeld, A., Jacoff, A., Noda, I., Takahashi, Y. (eds.) RoboCup 2005. LNCS (LNAI), vol. 4020, pp. 173–183. Springer, Heidelberg (2006)
11. Lima, J.L., Gonçalves, J.A., Costa, P.G., Moreira, A.P.: Humanoid Realistic Simulator - The servomotor joint modeling. In: ICINCO-RA, pp. 396–400 (2009)
12. Meriçli, Ç., Veloso, M.M.: Biped walk learning through playback and corrective demonstration. In: AAAI 2010: Twenty-Fourth Conference on Artificial Intelligence (2010)
13. Michel, O.: Webots: Professional mobile robot simulation. Journal of Advanced Robotics Systems 1(1), 39–42 (2004)
14. Morisawa, M., Harada, K., Kajita, S., Nakaoka, S., Fujiwara, K., Kanehiro, F., Kaneko, K., Hirukawa, H.: Experimentation of humanoid walking allowing immediate modification of foot place based on analytical solution. In: Proceedings 2007 IEEE International Conference on Robotics and Automation, pp. 3989–3994 (April 2007), http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4209709
15. Nishiwaki, K., Kagami, S.: Online walking control system for humanoids with short cycle pattern generation. The International Journal of Robotics Research 28(6), 729–742 (2009), http://ijr.sagepub.com/cgi/doi/10.1177/0278364908097883
16. Obst, O., Rollmann, M.: SPARK – A generic simulator for physical multiagent simulations. Computer Systems Science and Engineering 20(5), 347–356 (2005)
17. Schwefel, H.P.: Evolution and Optimum Seeking. Sixth-Generation Computer Technology. Wiley Interscience, New York (1995)
18. Smith, R.: Open dynamics engine (2002), http://www.ode.org
19. Vukobratovic, M., Borovac, B.: Zero-moment point – Thirty five years of its life. International Journal of Humanoid Robotics 1(1), 157–173 (2004)