Robotics, virtual reality, visualization, simulation, walking robot

Tomáš MICHULEK[*]

USING VIRTUAL REALITY TO DEVELOP SIX LEGGED WALKING ROBOT CONTROL SYSTEM

Abstract

This contribution presents the first results of our work related to design and development of a six legged walking robot. Real prototype was created along with its virtual counterpart that is simulated in virtual environment. A control system was created and tested on virtual model. This control system was then directly used to control real robotic construction.

INTRODUCTION

Autonomous mobile robots are becoming more and more important during last years. They are able to solve problems too dangerous for human beings. A special form of mobile robots are walking robots. This includes robots that locomote without wheels, or use walking wheels [9]. Their ability to move through uneven terrains makes them best choice for many applications were standard wheeled vehicles are useless. Many different constructions were created. Most popular are biped [12], quadruped [2] and six legged constructions. Robots are often inspired by various life forms [11].

Two main approaches can be used to control waking robot. First one is based on precise joint angle control. This approach is based on direct control of every DOF of the robot. The second is based on passive dynamics of the robot construction. In this case, part of the walk control problem is solved by robot construction. A precise joint control approach was used to control our robot.

Many software solutions were created to simulate walking robots. Examples can by found in [3] [6] [8] [10]. Simulation is being used to speed up the development of robot construction and to optimize its control system. The evolution of computing hardware has enabled more precise simulation of complicated robotic construction during last few years.

In this paper, we present the results of our development of a six legged walking robot, along with its virtual counterpart. A control system developed to control these robots is also described.

2. SIMULATION SOFTWARE

Since we had problems to find software package capable of meeting all our current and future requirements, our own software environment was created. It was written in C++ GLSL

[*] Ing. Tomáš Michulek, UKaI, University of Žilina, michulek@fel.uniza.sk

and Lua programming languages. It is based on OpenGL, SDL and Boost libraries. It iscompatible with Linux and Windows operating systems. Application consists of multiple different interface modules. Their responsibility is to interact with the user.

- **Viewport manager** - manages other modules and user interaction with the software
- **Model editor** – gives user the ability to create, modify and test virtual robots.
- **Control system editor** – gives user the ability to create and modify control systems.
- **Scene manager** – is responsible for simulation of robots in virtual environment.

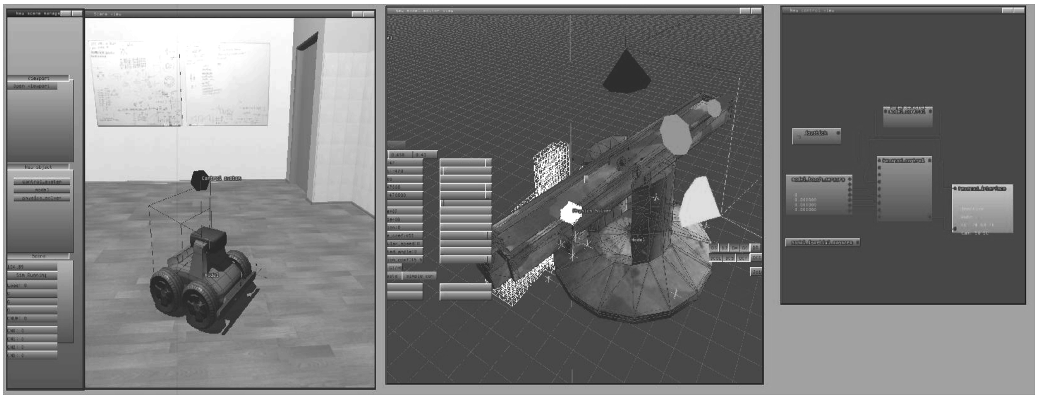

Fig. 1. Application front end: scene manager, model editor and control system editor

Each of these modules is independent dynamically loaded library. The simulation system itself is divided into these parts:

- **Robot body simulation**
- **Collision system**
- **Virtual sensors**
- **Communication interface**

2.1 Robot body simulation

Simulation of a robot body is based on rigid body dynamics. The model consists of rigid segments connected together with flexible joints. Segments are defined by their geometric shape, mass and inertia matrix. Every rigid part is affected by various forces. These forces are generated in joints in order to preserve the model geometric configuration. Another group of forces is generated as a result of a robot versus environment interaction. All these forces are then applied to a corresponding segment.

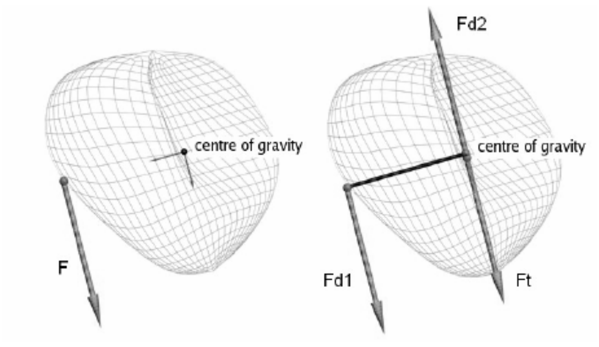

Fig. 2. Force transformation

After transformation displayed in Fig. 2. every force $\vec{F}_i$ is transformed to $\vec{F}t_i$, $\vec{F}d1_i$ and $\vec{F}d2_i$ parts. $\vec{F}t_i$ affects movement of the segment.
Resulting force is given by an expression:

$$\vec{F}_t = \sum_{i=0}^{n} \vec{F}_{ti} \tag{1}$$

Acceleration $\vec{a}$, speed $\vec{v}$ and position $\vec{c}$ of segments centre of gravity are then given by:

$$\vec{a} = \frac{\vec{F}}{m} \qquad \vec{v} = \vec{v}_0 + \int_0^T \vec{a}dt \qquad \vec{c} = \vec{c}_0 + \int_0^T \vec{v}dt \tag{2}$$

where T is actual time of simulation, $\vec{c}_0$ is position and $\vec{v}_0$ speed of the segment at the moment of a simulation start.
$\vec{F}d1_i$ with $\vec{F}d2_i$ forces affect rotation of the segment (Fig. 3.).
The resulting torque $\vec{M}_i$ is given by an expression:

$$\vec{M}_i = \vec{F}d1_i \times (\vec{p}_1 - \vec{p}_2) \tag{3}$$

The torque $\vec{M}$ affecting the segment is then computed by:

$$\vec{M}_t = \sum_{i=0}^{n} \vec{M}_i \tag{4}$$

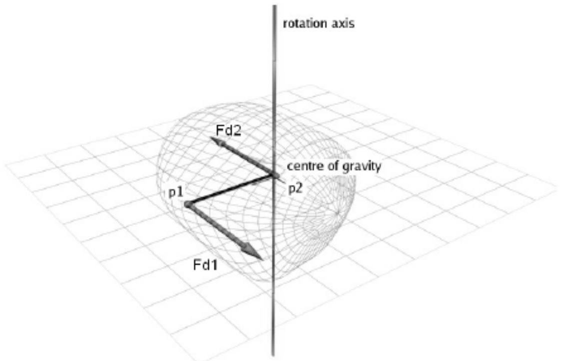

Fig. 3. Segment rotation

Angular speed of the segment can be simply computed from the torque $\vec{M}$. Control system commands are not directly affecting forces generated in joints. Various preprocessing algorithms are used to simulate real world motors. The model of six legged robot contains virtual servo motors. Only an expected angle of the joint supplied to this servos and simple regulator generates appropriate forces. This way the behavior of the simulated model can be externally affected. Simulation is being used for development and testing of control systems. Therefore, flawless behavior from the physics point of view is not required. Only a decent degree of similarity between reactions of real and virtual model is necessary.

2.2 Collision system

Every interaction between the robot and the environment results in a set of collision forces. Main task of the collision system is to compute these forces. To simplify this task, a special representation of objects in 3d space is used. Each object in the simulation is represented by a set of collision points(cpoints) and a set of collision triangles(ctriangles) (fig.4 left). Solving a ctriangle vs. cpoint collision is relatively simple. When two objects interact, the resulting force is computed as a sum of forces generated by individual ctriangle vs. cpoint collisions.

There are two separate forces resulting from the interaction between cpoint and ctriangle. First one is in the direction of triangle normal ($\vec{F}_N$). Second is parallel to the triangle ($\vec{F}_F$).

$\vec{F}_N$ is given by:

$$\vec{F}_N = \vec{N}.(C_p.d) - \vec{v}_y.C_d \tag{5}$$

Where $\vec{N}$ is triangle normal, C_p is press constant, d represents depth of collision, $\vec{v}$ is csphere speed relative to collision triangle and C_d id damping constant.

The friction force computing is more complicated, because static and dynamic friction must be taken into account.

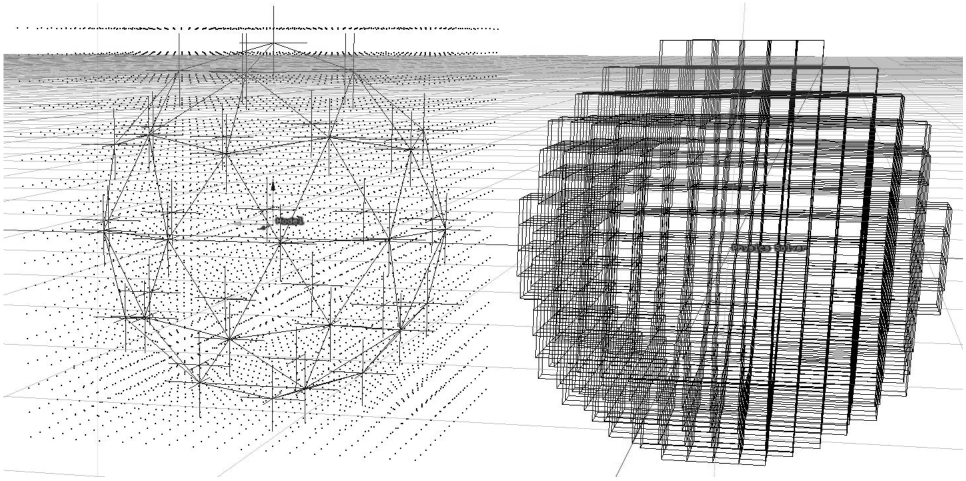

Fig. 4. Collision representation of the sphere

When two object interact, simplest solution is to compute the collision of every ctriangle of one object with every cpoint of the other. This raw force approach is very CPU demanding. When two objects with 1000 ctriangles and cpoints interact, 1000 000 collisions are computed every simulation step. More complicated 3d scene requires to much GFlops than today's CPUs or GPUs can handle. Therefore we had to find a way to lover the count of collision computed every simulation step.

We solved this problem by dividing local space of every object to certain amount of cubic subspaces (ccubes). This is displayed in fig.4 right. This cube system serves as a look-up table during collision computing. Each of these ccubes holds information about a special set of ctrinagles, than interfere with this particular cube. When a cpoint lies in this cube, there is no need to test its collision with any triangle, that does not belong to this special set. This lowers the number of collision computed per step by a considerable amount. In certain cases up to 100 times. The downside of this solution are its high memory requirements. To create a 100x100x100 ccube system, it is necessary to allocate at least 10 MB of memory. In the future we plan to further optimize the collision system, in order to lower its memory and computing requirements.

2.3 Virtual sensors

To test a control system in virtual reality, it is necessary to emulate actuators and sensors of the real robot (fig. 5). As a virtual counterpart to the real world actuators, we used the virtual servo motors. These were briefly described in other section of this article. To emulate the real world sensors, we used so called virtual sensors.

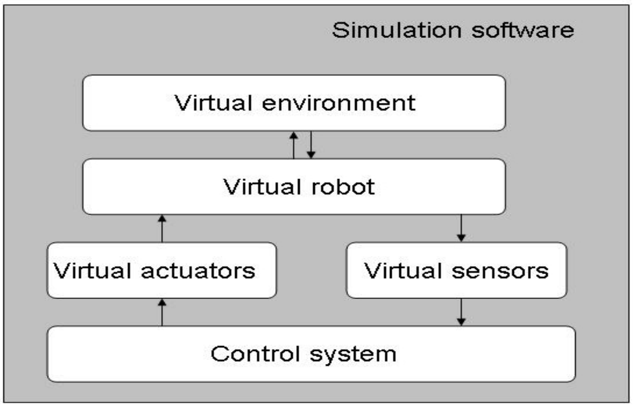

Fig. 5. Interactions inside the simulation software

Three basic types of virtual sensors were created:

- **Virtual touch sensors** - these sensors are reporting contact between the robot body and its surroundings. Touch sensor is connected to a cpoint and gets the information about any occurring collision from it. A control system receives a boolean value and a floating point value from every touch sensor. Boolean value holds an information about the collision occurrence and float value informs the control system about the size of a force vector generated during the collision.

- **Virtual distance meter** – this sensor scans distance between itself and nearest object in virtual world. It can be used as model for an ultrasound or infrared distance sensor.

- **Virtual camera** – grabs images of virtual world. This data an be used to develop, optimize and test image recognition algorithms. Robot with distance meter and virtual camera is displayed in (fig.6). The quality of virtual sensors output is very high. It is often necessary to add some noise to the signal, in order to get its quality closer to real world sensors output.

2.4 Communication interface

Peripheral devices communication interface is a fundamental element of the application. It allows image acquiring from an external USB of FireWire video camera. Real robot models are be controlled via RS232 protocol. When a lower bit-rate is acceptable, the wireless data-transfer modules are used. If necessary, the software can communicate via Internet using TCP/IP protocol too

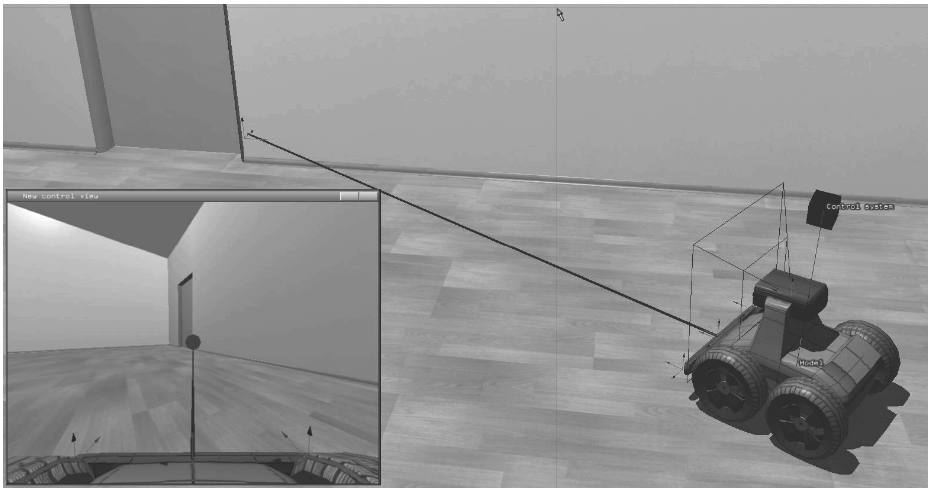

Fig. 6. Wheeled robot with a distance meter and a virtual camera

3. CREATED ROBOTS

For the purpose of practical tests we have constructed a real 6-legged walking robot. Its body is built from komatex plastic. Each leg is being moved by 3 servo motors. Whole construction has 18 DOF. A micro-controller AT Mega 16, which is placed inside the robot body, receives information about a desired position of each servo via RS232 interface. Then it encodes information into PWM for the servos.
Every servo motor contains integrated analog regulator, which rotates joint to angle encoded in the PWM signal. This regulator is acting similarly to its virtual counterpart integrated in virtual servo motors.

Current supply of the robot is provided by PC-ATX power supply or a set of Li-Pol batteries. Servos and micro-controller are operating with voltage +5V. The current consumption is up to 20A. The real robot is shown in Fig. 7.

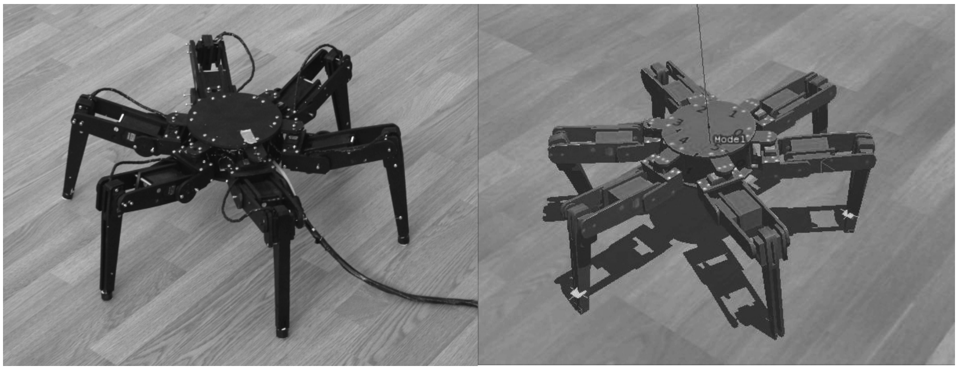

Fig. 7 Walking robot prototype and its virtual counterpart

When the construction of the real robot had been finished, its virtual copy was created. Geometric parameters of real and virtual robot are nearly identical. Parameters of virtual servo motors can be changed in order to improve its performance and speed up testing in virtual reality. The virtual model is shown in Fig. 7.

Virtual robot is equipped with six touch sensors. One on a tip of each leg. These sensors are used as a source of information about the terrain the robot is walking over uneven terrain. Each of these sensors reports contact between the leg and the surface. These sensors are We plan to mount directional touch sensors on our real prototype in near future.

4. CONTROL SYSTEM

In order to test the behavior of robots, a simple control system was developed (Fig. 7.). It allows the robot to walk in any direction, rotate and change distance between its body and a surface. The control system is divided into three cooperating blocks:

- **Central control**
- **6 individual leg controllers**
- **Inverse and forward kinematics solver**

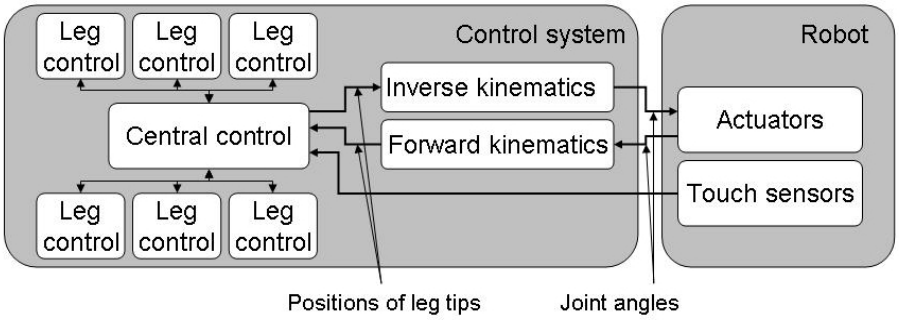

Fig. 8. Control system

The central control block coordinates steps of legs and its primary duty is to keep the robot balanced. At least three legs must be on the ground to ensure stability during walk. The individual leg controller must request permission from the central control before it can lift the leg from the ground. If the request of the leg controller is rejected, movement of the robot body is stopped until sufficient number of other legs is placed on the ground.

The individual leg controller generates vector $\vec{w}_i$, that defines direction in which the leg sole is supposed to move, relatively to the robot body. For example, if an expected movement of the robot body is $\vec{m}$ and the leg is on the ground, then $\vec{w}_i$ is given by an expression:

$$\vec{w}_i = -\vec{m}_i \tag{6}$$

The actual position of the sole $\vec{s}_i$ is then given by:

$$\vec{s}_i(T) = \vec{s}_i(0) + \int_0^T \vec{w}_i(t)dt \tag{7}$$

Leg controller acts as a simple state machine. Its states are UP, LAYING, DOWN and LIFTING When the leg controller is in LAYING state and corresponding touch sensor informs it about contact with the ground, it changes its state to DOWN.
Each leg has its own boundaries were its tip can be moved. These boundaries depend on geometric configuration of the leg. When the leg lying on the ground is about to leave its boundaries, it is lifted and moved in the opposite direction. Initially, the cylindrical boundaries were chosen in order to simplify a control system creation. More complex shape will increase the future algorithm performance.

The inverse kinematics block transforms expected positions of the leg tips to expected angles of the joints. A simple algebraic solution is used for this purpose. This prevents any unwanted oscillations. Expected angles are used to control the motors directly. This approach allows us to change geometric configuration of the robot without the need of changing any part of the control system except inverse kinematics solver.. The state DOWN is the only one of the four that grants support for the robot body. Leg controller must request permission from central control block, to leave the DOWN state.

The size of the leg movement vector $\vec{w}_i$ varies, depending on the current state of the leg . Lifting of the leg is faster than laying and the leg on the ground moves slower than lifted one.

5. EXPERIMENTAL RESULTS

The initial tests and optimizations of the control system were made exclusively on the virtual model. Then, without further changes, the control system was connected to the real robot. During this test, the response of the real robot was very similar to the response previously observed in virtual reality. Only minor differences in specific situations were observed. This fact made doing changes of the control unit very easy, because no intensive testing on the real model was needed.

When both robots were simultaneously connected to the same control system, difference between positions of their bodies became visually recognizable after 20-25 seconds. Simultaneous control was used to find hardware and software errors. When power supply of one servo, for example, was partially unplugged, the behavior of the real robot changed, but the virtual model behavior remained the same. This directed us to a possible hardware problem, rather than a control system inconsistency. When the behavior of both robots changed in a wrong way, the control system was checked first.

Resulting speed of the robot walking is given by expression:

$$v = \frac{l_s}{t_u + t_d + \frac{l_s}{v_f} + \frac{l_s}{v_b} + t_s} \tag{8}$$

Where v is speed of the robot body movement, l_s is length of each step, t_u is time needed to lift leg from the ground, t_d is time needed to lay leg on the ground, v_f is speed of body supporting leg tip, v_b is speed of the lifted leg tip, t_s is additional time depending on current leg synchronization state. When the robot is walking straight in one direction, synchronized walking is possible and three legs move simultaneously. But when robot is moving and rotating at the same time, synchronization is lost and resulting movement is slower. This issue will be hopefully fixed during further control system optimization. Maximum walk speed achieved during our experiments with real prototype was approximately 1 m/s. Besides horizontal movement we also observed vertical movement of the virtual robot body during walking (fig.9). This movement is a result of low servo-motor torque and touch sensor signal delay. We hope it will be improved soon.

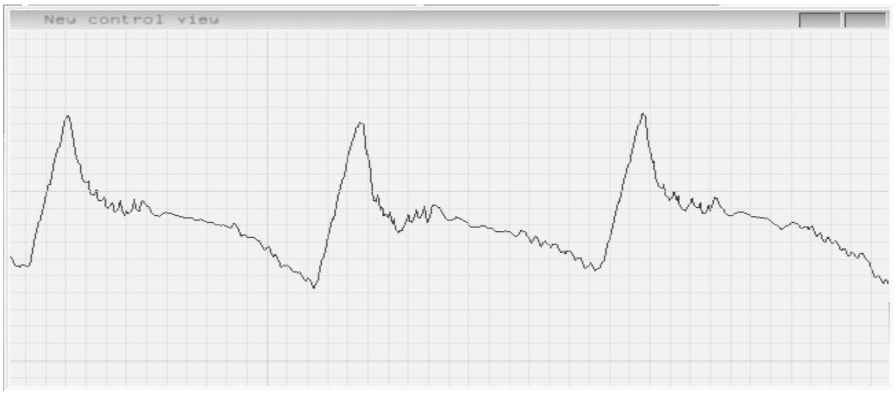

Fig. 9. Vertical body movement

6. CONCLUSIONS

In this paper we demonstrated possibilities of using virtual reality for walking robot control system development and a walk control solution. Our results can be used for further research in the field of walking robot control system development. In our further work we plan to combine our recently achieved results with artificial intelligence methods.

References

Literature

[1] LIPSON H., BONGARD J., ZYKOV V., MALONE E. : *Evolutionary Robotics for Legged Machines: From Simulation to Physical Reality*, Computational Synthesis Lab Cornell University, Ithaca NY 14853, USA

[2] DE LASA M. and BUEHLER M.: *Dynamic Compliant Walking of a Quadruped Robot:Preliminary Experiments*, Dept. of Electrical Engineering & Dept. of Mechanical Engineering,Center for Intelligent Machines, McGill University Montreal, Canada

[3] HOPLER R., STELCER M. and VON STRYK O.: *Object-oriented dynamics modeling for simulation, optimization and control of walking robots*. In Proc. 18th Symposium on Simulation Technique, ASIM, Erlangen, September 12-15, 2005

[4] BUSCH J., ZIEGLER J., AUE C., ROSS A., SAWITZKY D., BANZHAF W.:*Automatic Generation of Control Programs for Walking Robots Using Genetic Programming*, University of Dortmund, Department of Computer Science, Chair of Systems Analysis (LSXI) Dortmund, Germany

[5] BILLARD A., J. IJSPEERT A.
Biologically inspired neural controllers for motor control in a quadruped robot.
Robotics Laboratory, University of Southern California, U.S.A

[6] HAAVISTO O. , HYOTYNIEMI H.: *Simulation tool of a biped walking robot model*
Helsinki University of Technology Control Engineering Laboratory, Finland
In Espoo 2004 Report 138

[7] PORTA J.M., CELAYA E.: *Body and leg coordination for omnidirectional walking in rough terrain*, Institut de Robotika i Informatica Industrial (UPC-CSIC), Barcelona, Spain

[8] MICHEL O: *WebotsTM: Professional Mobile Robot Simulation*
Swiss Federal Institute of Technology in Lausanne, BIRG & SWIS research groups,

[9] HALME A., EPPANEN I., SALMI S., YLONEN S. , *Hybrid locomotion of a wheel-legged machine*, Automation Technology Laboratory Helsinki University of Technology, Finland

[10] AARNIO P., KOSKINEN K., SALMI S., *Simulation of the Hybtor Robot*, Information and Computer Systems in Automation, Helsinki University of Technology, Finland

[11] D. A. KINGSLEY, R. D. QUINN, R. E. RITZMANN: *A Cockroach Inspired Robot With Artificial Muscles*, Case Western Reserve University Cleveland Ohio, USA,

[12] CANDANA P. and Bongard J. C.: *The Road Less Travelled: Morphology in the Optimization of Biped Robot Locomotion* , Artificial Intelligence Laboratory, University of Zurich, Switzerland