Proceedings of the 2002 IEEE
International Conference on Robotics & Automation
Washington, DC • May 2002

Open Architecture Humanoid Robotics Platform

Fumio KANEHIRO[1] Kiyoshi FUJIWARA[1] Shuuji KAJITA[1]
Kazuhito YOKOI[1] Kenji KANEKO[1] Hirohisa HIRUKAWA[1]
Yoshihiko NAKAMURA[2] Katsu YAMANE[2]
[1]National Institute of Advanced Industrial Science and Technology(AIST),
Tsukuba Central 2, 1-1-1 Umezono, Tukuba, Ibaraki, 305-8568 JAPAN
{f-kanehiro,k-fujiwara,s.kajita,kazuhito.yokoi,
k.kaneko,hiro.hirukawa}@aist.go.jp
[2]The Univ. of Tokyo, 7-3-1 Hongo Bunkyo-ku, Tokyo, 113-8656 JAPAN
{nakamura,katz}@ynl.t.u-tokyo.ac.jp

Abstract

This paper introduces an open architecture humanoid robotics platform (OpenHRP for short) on which various building blocks of humanoid robotics can be investigated. OpenHRP is a virtual humanoid robot platform with a compatible humanoid robot, and consists of a simulator of humanoid robots and motion control library for them which can also be applied to a compatible humanoid robot as it is. OpenHRP is expected to initiate the exploration of humanoid robotics on an open architecture software and hardware, thanks to the unification of the controllers and the examined consistency between the simulator and a real humanoid robot.

1 Introduction

The Ministry of Economy, Trade and Industries of Japan has run Humanoid Robotics Project (HRP for short) since 1998 for five years [1, 2]. Four copies of a humanoid robot (called HRP-1), teleoperations cockpit for them and a virtual humanoid robot platform (V-HRP for short)[3, 4] had been developed in phase one of HRP as the research platform, and various applications of humanoid robots are under development in phase two on the platform. We call the organization of the project *platform-based approach*. It is the antithesis of usual robotics projects in which elementary technologies are developed at first and they are integrated into a system at the final stage.

The architecture of the platform is being made open, step by step. The software of HRP-1 have been developed by Honda R&D as well as its hardware, and it is provided as a black box. We have replaced the controller for biped locomotion by our own one which has been developed on V-HRP. Besides, V-HRP has also been replaced by a new simulator on which we can develop the controllers portable to the hardware without any modification. Finally, a new humanoid robot HRP-2 is to be developed, and the controllers examined on HRP-1 will be applied to HRP-2.

The simulator is build on CORBA[5] which is a standard software architecture, and the realtime controller of the robot is run on ART-Linux[6] which is a realtime extension of Linux. Besides, the simulator and the controllers are now white boxes. Therefore, we call the package of the simulator and controllers with the compatible humanoid robot *OpenHRP*[1] which stands for Open Architecture Humanoid Robotics Platform.

Because the unification of the controllers and the consistency between the simulated and real robots are realized, OpenHRP can be a useful virtual platform for humanoid robotics on which various fundamental technologies can be developed. The virtualization of the platform is very important to inherit software library from one hardware to another efficiently.

This paper is organized as follows. Section 2 overviews the configuration of OpenHRP. Section 3 explains objects each of which constitutes OpenHRP in detail. Section 4 introduces several applications developed on OpenHRP. Section 5 concludes the paper.

[1]You can download OpenHRP after user registration at http://www.is.aist.go.jp/humanoid/openhrp/

0-7803-7272-7/02/$17.00 © 2002 IEEE

2 Overview of OpenHRP

2.1 Functional Features

OpenHRP has the following functional features.
[Dynamics computation] Thanks to an automatic computation of structure change and constraints, and to the employment of fewer coordinates, it can efficiently compute the dynamics of structure-varying kinematic chains between open chains and closed ones like humanoid robots[7].
[Contacts and collision computation] It can handle contacts and collision between arbitrary polyhedral objects. It implements two kinds of the algorithms to handle contact constraints. One is based on a conservation law of momentum which is numerically stable. And the other is based on a spring-damper model. Using these features, it is possible to simulate the situation that a humanoid which has compliant feet walks on the uneven terrain.
[Unification of controllers] This is realized by hardware abstraction and synchronization mechanism and employing ART-Linux in which realtime processing is available at the user level[8].

2.2 Implementation Features

OpenHRP is implemented as a distributed object system on CORBA(Common Object Request Broker Architecture)[5]. A user can implement a controller using an arbitrary language on an arbitrary operating system if it has a CORBA binding. A lot of implementation of ORB exist, ORBacus[9] is adopted in OpenHRP. Because ORBacus has C++ and Java binding and supports several operating systems.

OpenHRP consists of several CORBA objects and they can be distributed on the Internet and executed in parallel. Each server can be replaced with another implementation if it has the same interface defined by IDL (Interface Definition Language). Using the language independence feature of CORBA, some of objects are implemented using Java and Java3D, the others are implemented using C and C++. Currently, OpenHRP supports Windows NT4.0/2000/98/Me and Linux.

The activation of each server is done using an IMR(Implementation Repository) function automatically on demand. Each server consists of a CORBA interface part, a native language interface part and an core logic part which are shown in Fig.1. A realtime routine uses native interface and a non realtime routine does CORBA interface. As a result, lots of codes are shared between the simulator and the controller, and this sharing let the development more efficient.

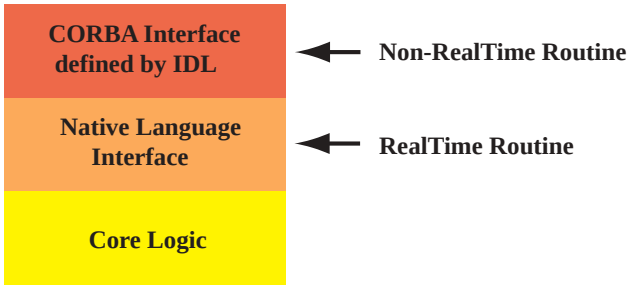

Figure 1: Internal Structure of OpenHRP servers

2.3 Configuration of OpenHRP

The configuration of OpenHRP is shown in Fig.2.

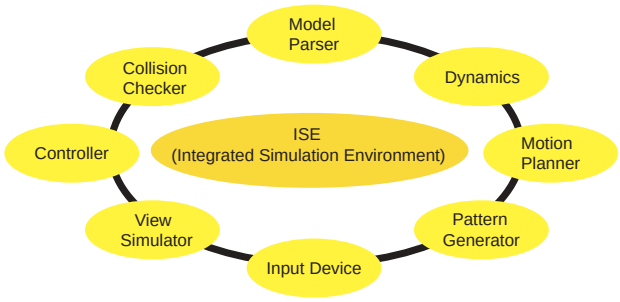

Figure 2: CORBA Objects of OpenHRP

A simulation is controlled by a CORBA client which has a graphical interface shown in Fig.3 which is called ISE(Integrated Simulation Environment). ISE uses services provided by four CORBA servers, i.e. collision checker, model parser, dynamics and view simulator. A user can implement a realtime control algorithm which is to be embedded in a CORBA skeleton prepared beforehand.

The functions of each server are as follows.

ModelParser This server loads a VRML file describing the geometric models and dynamics parameters of robots and their working environment, and provides these data to other servers.

CollisionChecker The interference between two sets of triangles is inspected, and the position, normal vector and the depth of each intersecting point are found. RAPID[10] is enhanced to this end.

Dynamics The forward dynamics of the robots are computed.

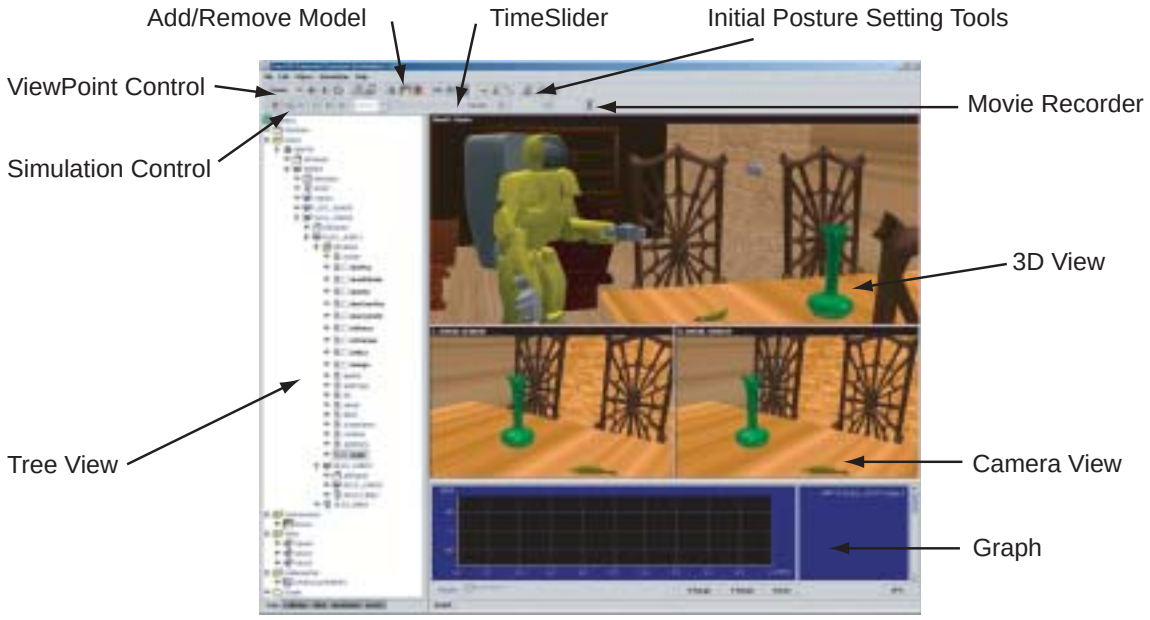

Figure 3: Integrated Simulation Environment(ISE)

Controller This server is the controller of a robot, which is usually developed by the users of OpenHRP.

View Simulator A field of view from cameras on a humanoid is generated.

Pattern Generator A dynamically stable walking motion is calculated and trajectories of joint angles and ZMP(Zero Moment Point) are generated.

Motion Planner A motion path which is collision free and dynamically stable is computed.

Input Device A status of an input device such as a joystick is provided.

3 Building blocks of OpenHRP

3.1 Dynamics simulator

Using the servers, the forward dynamics of the robots are computed in the following procedure.

Setting up of the simulation environment
(1) `ModelParser` reads a VRML file via HTTP protocol. The kinematics and dynamics parameters are sent to `DynamicsServer` and the geometric model is to `CollisionChecker`.

Execution of the dynamics simulation
(2) `Controller` reads the outputs of the simulated sensors while communicating with `DynamicsServer`.
(3) `Controller` and `DynamicsServer` execute the computations. Note that these computations can be run in parallel. The outputs of `Controller` are the torques of the actuators, and those of `DynamicsServer` are the updated states of the robot.
(4) While the forward dynamics is computed, `CollisionChecker` is called to find the position, normal vector, and the depth of each intersecting point.
(5) After these computations, `Controller` sends the control outputs to `DynamicsServer`.

Visualization and recording
(6) ISE acquires the current states of the world from `DynamicsServer`, visualizes the simulated world and records it.

In order to evaluate the performance of OpenHRP, biped locomotion of a humanoid robot is simulated. The sample humanoid robot has 6DOF arms, 6DOF legs, 3DOF waist, 2DOF neck and 29DOF in total. The interference between every links of the humanoid and the ground is checked and the interference between a foot link and the ground always occurs during the simulation. The specifications of

the platform computer include CPU:Intel PentiumIII 933MHz, Memory:512MB, and OS:Linux-2.2.17. The computation time except the visualization is 25[ms] per the unit integration time, which is usually set around 1 [ms].

3.2 View Simulator

3.2.1 Modeling for view image synthesis

View image synthesis consists of three parts, that is, modelings of illumination, shapes and materials of objects in a scene, and cameras.

Illumination Among them, illumination is relatively easier to model than the rest, since IES format data[11] can provide color, initial strength and ray distribution of many kinds of artificial light source. These data for natural light are also available. We employ IES data to model illumination.

Shapes and materials The shapes of artificial objects can also be obtained from CAD data, but it is hard to obtain the material model of objects' surface. Sato, Wheeler and Ikeuchi have been studying how to get reflectance data from observation[12].

Cameras The modeling of cameras is neither straightforward. It is desirable to calibrate images according to the zoom, focus and iris of cameras. Asada, Baba and Amano have been investigating these calibration problem [13].

Though the exact modeling of reflectance and the camera calibration are important to synthesize realistic images, we have not considered these problems so far. Because our goal of a view simulator of a humanoid robot is not having realistic images for humans, but equivalent images to real ones for image processing included in object recognition, objects tracking and/or navigation.

3.2.2 Rendering

Recalling that the viewpoints of a humanoid robot is changing frequently, it is easy to notice that usual ray tracing algorithm must take too much time for generating a view image. This is because usual ray tracing process is invoked from the scratch when the viewpoint is changed. The next option is employing simple graphics software capable of hidden surface removal, shading etc. But then it is not straightforward to model illumination, because no standard model is available for illumination in the case and the number of lights is limited to a small number for the real-time computation of the lighting equation.

The third option is radiosity rendering. IES format data mentioned above can be used for modeling illumination in several kinds of commercially available software based on this rendering algorithm. Besides, a resulting solution of radiosity rendering computation is a 3D model of a scene and it is possible to generate images at the frame rate when a viewpoint is given, because radiosity rendering computes Lambertian reflection only which does not depend on a viewpoint. A bad news of radiosity rendering is that the synthesized images lack to have the effect from specular reflection. When the surfaces of objects in a scene is smooth like metal surface, the effect of specular reflection has become more dominant and the synthesized images look significantly different from the corresponding real ones.

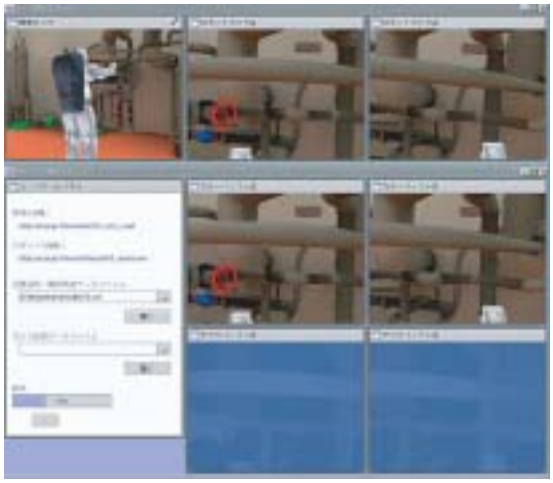

Figure 4: Example of the simulated view

Figure 4 is a snapshot of a screen of the view simulator. The middle and the right figures in the first row show the field of view from the left and right cameras respectively on the server, and those in the second row are the copies at the client. The figures in the third row are the corresponding depth images.

3.3 Graphical user interface

ISE provides functions to set up a simulation environment and analyze the simulation result.

At the simulation preparation stage, a user can load/unload models constituting the simulation world, and set initial position/attitude and joint angles using GUI interactively. The integration method and an integration timestep of the forward dynamics computation and the assignment of a controller can also be set using GUI, and these settings can be saved in a project file and reused. While executing the simulation, a progress is displayed in a 3D screen and it is possible to stop in the middle at need, change setting

and execute it again. When the simulation finishes, the result can be examined using graphs and converted into a movie.

3.4 Model editor

A robot and an environment model are described in an enhanced VRML97 format. It is based on a format which is decided by h-anim(Humanoid Animation) WG[14] and enhanced partly by ourselves. Using this format, all information which is required by the dynamics simulation such as a shape, kinematics parameters, dynamics parameters and so on is embedded in a single file. H-anim format defines three prototype nodes, Humanoid, Joint and Segment. They expresses one whole humanoid, a joint and a link. The following is a part of a model definition of a humanoid.

```
DEF HRP1 Humanoid {
 humanoidBody [
  DEF WAIST Joint {
   jointType "free"
   translation 0 0 0
   rotation 0 0 1 0
   children [
    DEF BODY Segment {
     mass 0.5
     momentsOfInertia [1 0 0 0 1 0 0 0 1]
      children [
       Inline {url "shape.wrl"}
      ]
    }
    DEF LEG_JOINT0 Joint {
     jointType "rotate"
           ....
```

There are many tools which can edit shapes and output it in VRML format. But there is few tools comprising the function that can read VRML and edit a scene graph. In addition, there is no tool which supports a prototype node perfectly. Because it is difficult that a user edit a format as the above with a normal text editor, the model editor shown in Fig.5 is provided as a part of OpenHRP. This editor is used to assemble shapes that is made with a commercial tool and input the parameter that is necessary for a simulation.

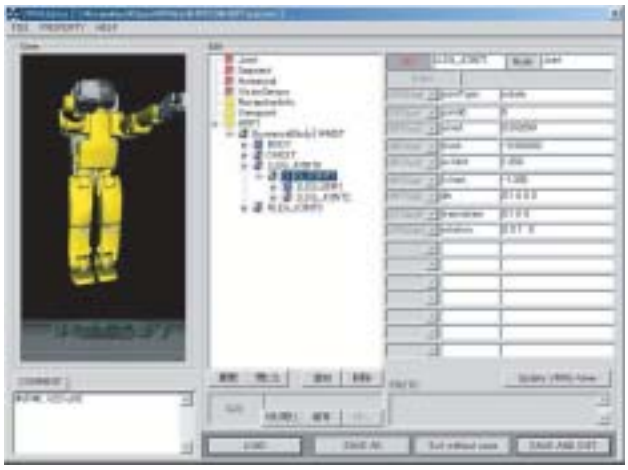

Figure 5: OpenHRP Model Editor

4 Applications of OpenHRP

4.1 Development of motion pattern generator

The walking pattern generator which based on the 3d linear inverted pendulum mode is developed on OpenHRP. A distinguished feature of this generation technique compared with others[15] is that it is able to generate a pattern at very high speed since it doesn't use an iterative calculation. Using this feature, it is not only able to generate a pattern with offline but also with online when necessary. Fig.6 shows an example of a generated pattern. The robot walks forward(rightward in the figure) at first, turn left, go backward, turn right and go forward again in this pattern. Small circles, a solid line and a broken line indicate foot places, a trajectory of ZMP and a trajectory of a waist of the robot respectively.

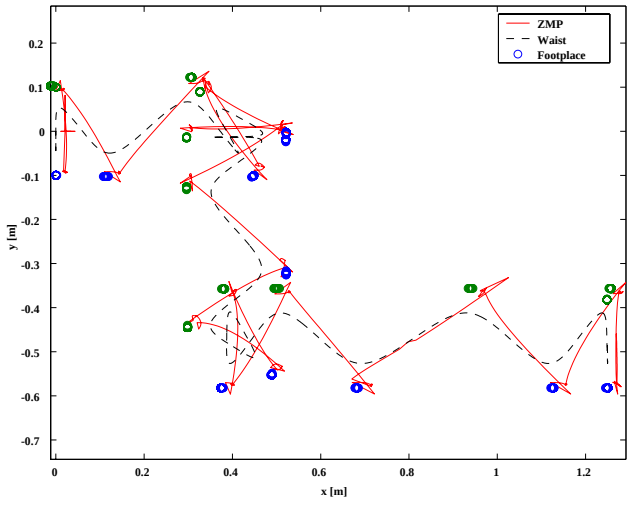

Figure 6: Example of a Generated Pattern

This walking pattern generator is also implemented by the structure that shown in Fig.1, and the identical code can be used in a CORBA server used in an offline fashion as well as in a realtime controller.

4.2 Development of feedback controller

A humanoid can't walk stably when a generated pattern is playbacked simply, because of modeling errors and an influence of compliant elements built in the feet of the robot. So a walking stabilization controller is developed on OpenHRP. This controller consists of two parts. Each foot's desired position and orientation are adjusted in order to compensate the body inclinations assuming the robot body as an inverted pendulum. The horizontal position of the torso is modified in order to reduce the error between the desired ZMP and the actual ZMP.

The result of a walk simulation using the stabilization controller is shown in Fig.7. Upper two graphs show angles of inclination of the body around roll/pitch axis while walking. These angles are estimated by the kalman filter using outputs of a gyrometer and an accelerometer. The lower one shows a vertical element of the ground reaction force which is measured by a force/torque sensor which is embedded in a foot.

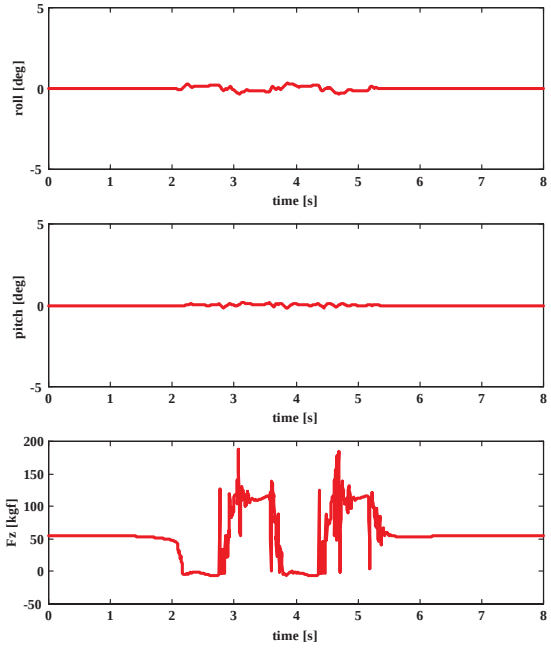

Figure 7: Simulation Result of Walking Pattern

The developed controller is applied to the real robot platform hardware HRP-1S which is shown in Fig.8[16]. HRP-1S has 1600[mm] height, 600[mm] width and 99[kg] weight excluding batteries. It is controlled by a Intel PentiumIII based VME CPU board which is in its backpack. ART-Linux is running on it.

This application doesn't need porting of the controller thanks to the controller unification mechanism of OpenHRP. This mechanism is realized by introducing an adaptor which generalizes interfaces of the hardware body and the software body in the simulation world. Therefore, this application is done by simply replacing an adaptor for the simulated body with one for the real one.

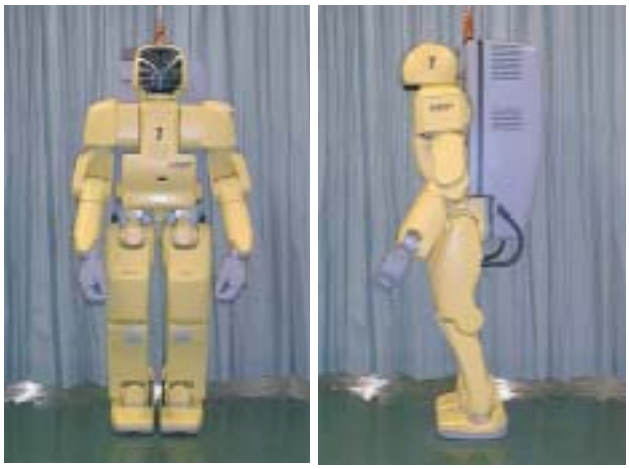

Figure 8: Humanoid Robot Platform HRP-1S

Figure 9 shows its result. Both of body inclination angles and a ground reaction force show the behavior that resembled very well with the simulation.

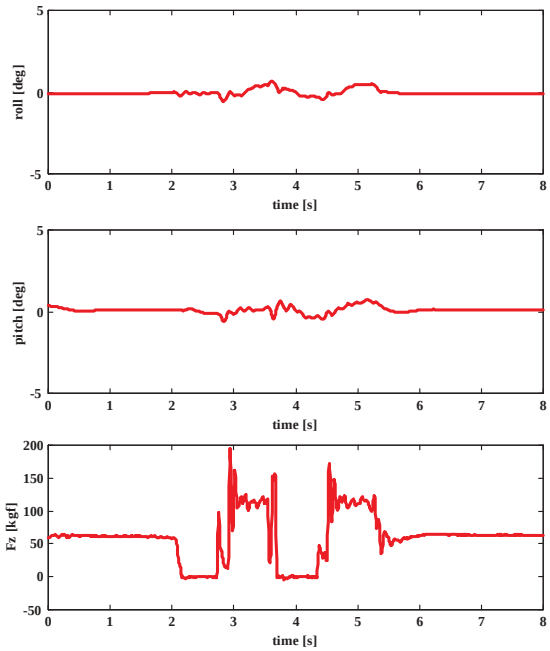

Figure 9: Experimental Result of Walking Pattern

5 Conclusions

This paper presented an open architecture humanoid robotics platform OpenHRP. The results can be summarized as follows.

- Thanks to the architecture based on CORBA, research and development becomes possible without dependency on programming languages and operating systems.
- It is possible to simulate humanoid behaviors fast and precisely by the introduction of an efficient dynamics computation, an efficient contacts/collision computation, which can handle an arbitrary shape, and a contact model which can treat a compliant element.
- Thanks to the unification, the controllers can share softwares with the dynamics simulator including the parameter parser, kinematics and dynamics computations and the collision detector. This feature can make the development of the controllers more efficient and the developed controllers more reliable.
- Consistency with an actual robot and a simulation is confirmed by a simulation and an experiment of a walking motion.

We claim that a humanoid robot platform deserves to be called a platform if the identical software can be used either on the simulator or on the robot and if the consistency between them is satisfactory kept. We believe that OpenHRP is expected to be the first humanoid robot platform in this sense and that it can initiate the exploration of humanoid robotics on an open architecture hardware and software.

Acknowledgements

This research was supported by the Humanoid Robotics Project of the Ministry of Economy, Trade and Industry.

References

[1] H. Inoue, S. Tachi, K. Tanie, K. Yokoi, S. Hirai, H. Hirukawa, K. Hirai, S. Nakayama, K. Sawada, T. Nishiyama, O. Miki, T. Itoko, H. Inaba, and M. Sudo. HRP:Humanoid Robotics Project of MITI. In *Proc. of the First IEEE-RAS International Conference on Humanoid Robots*, 2000.

[2] Hirochika Inoue, Susumu Tachi, Yoshihiko Nakamura, N.Ohyu, Shigeoki Hirai, Kazuo Tanie, Kazuhito Yokoi, and Hirohisa Hirukawa. Overview of Humanoid Robotics Project of METI. In *Proc. of the 32nd ISR*, 2001.

[3] Y. Nakamura, H. Hirukawa, K. Yamane, S. Kajita, K.Yokoi, K. Tanie, M.G. Fujie, A. Takanishi, K. Fujiwara, F. Kanehiro, T. Suehiro, N. Kita, Y. Kita, S. Hirai, F. Nagashima, Y. Murase, M. Inaba, and H. Inoue. V-HRP:Virtual Humanoid Robot Platform. In *Proc. of the First IEEE-RAS International Conference on Humanoid Robots*, 2000.

[4] Y. Nakamura, H. Hirukawa, K. Yamane, S. Kajita, K. Fujiwara, F. Kanehiro, F. Nagashima, Y. Murase, and M. Inaba. Humanoid Robot Simulator for the METI HRP Project. In *Proc. of the 32nd ISR*, 2001.

[5] http://www.omg.org. Object Management Group.

[6] Youichi Ishiwata and Toshihiro Matsui. Development of Linux which has Advanced Real-Time Processing Function. In *Proc. 16th Annual Conference of Robotics Society of Japan*, pp. 355–356, 1998.

[7] Katsu Yamane and Yoshihiko Nakamura. Dynamics Computation of Structure-Varying Kinematic Chains for Motion Synthesis of Humanoid. In *Proc. of the 1999 IEEE International Conference on Robotics & Automation*, pp. 714–721, 1999.

[8] Fumio Kanehiro, Masayuki Inaba, Hirochika Inoue, Hirohisa Hirukawa, and Shigeoki Hirai. Developmental Software Environment that is applicable to Small-size Humanoids and Life-size Humanoids. In *Proc. of the 2001 IEEE International Conference on Robotics & Automation*, pp. 4084–4089, 2001.

[9] http://www.ooc.com. Object Oriented Concepts, Inc.

[10] S. Gottschalk, M. C. Lin, and D. Manocha. OBBTree:A Hierarchical Structure for Rapid Interference Detection. In *Proc. of ACM Siggraph '96*, 1996.

[11] *IES Lighting Handbook*. 1987.

[12] Y. Sato, M.D. Wheeler, and K. Ikeuchi. Object shape and reflectance modeling from observation. In *SIGGRAPH-97*, pp. 379–387, 1997.

[13] N. Asada, M. Baba, and A. Amano. Calibrated computer graphics: a new approach to realistic image synthesis based on camera calibration. In *Proc. Int. Conf. on Pattern Recognition*, pp. 705–707, 1998.

[14] http://www.h-anim.org/. HUMANOID ANIMATION WORKING GROUP.

[15] Ken'ichirou NAGASAKA, Masayuki INABA, and Hirochika INOUE. Dynamic Walking Pattern Generation for a Humanoid Robot Based on Optimal Gradient Method. In *Proc. of the 1999 IEEE International Conference on Systems, Man, and Cybernetics*, pp. FA22–3, 1999.

[16] K. Yokoi, F. Kanehiro, K. Kaneko, K. Fujiwara S. Kajita, and H. Hirukawa. A Honda Humanoid Robot Controlled by AIST Software. In *Proc. of the IEEE-RAS International Conference on Humanoid Robots*, 2001.