V-HRP: Virtual Humanoid Robot Platform

Yoshihiko Nakamura[1], Hirohisa Hirukawa[2], Katsu Yamane[1], Shuuji Kajita[3], Kazuhito Yokoi[3], Kazuo Tanie[3], Masakatsu G. Fujie[4], Atsuo Takanishi[5], Kiyoshi Fujiwara[2], Fumio Kanehiro[2], Takashi Suehiro[2], Nobuyuki Kita[2], Yasuyo Kita[2], Shigeoki Hirai[2], Fumio Nagashima[6], Yuichi Murase[6], Masayuki Inaba[1], and Hirochika Inoue[1]

[1] The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan
{nakamura,katz}@ynl.t.u-tokyo.ac.jp {inaba,inoue}@jsk.t.u-tokyo.ac.jp
[2] Electrotechnical Laboratory, MITI, 1-1-4 Umezono, Tsukuba 305-8568, Japan
{hirukawa,kfujiwar,kanehiro,suehiro,nkita,terashi,hirai}@etl.go.jp
[3] Mechanical Engineering Laboratory, MITI, 1-2 Namiki, Tsukuba 305-8564, Japan
{kajita,yokoi,tanie}@mel.go.jp
[4] Mechanical Engineering Research Laboratory, Hitachi Ltd.,
502 Kandatsu, Tsuchiura 300-0013, Japan
mgfujie@merl.hitachi.co.jp
[5] Waseda University, 3-4-1 Okubo, Shinjuku-ku, 169-8555 Tokyo, Japan
takanisi@mn.waseda.ac.jp
[6] Fujitsu Laboratories Ltd., 4-1-1 Kamikodanaka, Nakahara, Kawasaki 211-88, Japan
{shiro,madonna}@stars.flab.fujitsu.co.jp

Abstract. We have developed a simulator of humanoid robots and a controller of their whole body motions in MITI's Humanoid Robotics Project. The simulator can emulate the dynamics of the motions of the robots whose structure may vary, and generate a sequence of the £elds of view from the eyes of the robot according to the motions. The structure-varying system is managed by introducing virtual links. The controller can handle a biped locomotion, a dynamic balance control at the standing position and collision avoidance motions for the robots. These software modules are integrated via CORBA, which enables the Internet clients to use the software. A humanoid robot testbed has also been developed to verify the accuracy of the simulation by experiments in the real world. We call the system *Virtual Humanoid Robot Platform* which we expect to be the virtual counterpart of the hardware robot platform for humanoid robotics.

1 Introduction

We have developed a simulator of humanoid robots and a controller of their whole body motions in MITI's Humanoid Robotics Project (HRP for short). The simulator can execute ef£cient dynamics and kinematics computation of structure-changing kinematic chains, that include any kinematic chains, open or close, and even such kinematic chains as to change connectivity in operation [10]. This function is essential, because the change of connectivity is often seen when a humanoid walks, touches or holds the environments, grabs an object with the both arms, and is even connected with another humanoid. The simulator can generate a sequence of the £elds of view from the eyes of the robot according to the dynamics simulation. We call this function of the simulator *a view simulator*. When the view simulator is integrated with the dynamics simulator, visual feedback motions of humanoid robots can be simulated.

The controller can handle a biped locomotion, a dynamic balance control at the standing position and collision avoidance motions for the robots. The biped locomotion consists of a walking on a ¤at ¤oor, turning, ascending/descending stairs, and interpolations between them. The dynamic balance control is necessary when the robot is doing some task at the standing position. The planning of collision avoidance motions of a humanoid robot must take the dynamic balance into account as well as obstacles. We have investigated how to assign the degrees of the freedom of the robot to the collision avoidance and the balancing.

A humanoid robot testbed has also been developed to verify the accuracy of the simulation by experiments in the real world. The testbed is a small humanoid robot with 540mm height and 8kg weight, but the con£guration of the robot is identical with the humanoid robot platform developed in HRP.

These software modules are integrated via CORBA (Common Object Request Broker Architecture), which also enables the Internet clients to use the software. In more details, the modules are implemented as CORBA servers, and a client can utilize them if the servers are reachable via IIOP (Internet InterOrb Protocol). We call the whole system *Virtual Humanoid Robot Platform* (V-HRP for short) which we expect to be the common base of humanoid robotics research focusing on software developments for the community.

This paper is organized as follows. In Section 2, the algorithms for the dynamics simulation are described in details and the design of the view simulator is presented. In Section 3, the principle and the speci£cations of the controllers are investigated. In Section 4, the architecture of the whole system is described. In Section 5, the comparison between the dynamics simulation and the experiment by the testbed is investigated. We conclude the paper in Section 6.

2 Humanoid Robot Simulator

2.1 Dynamics Simulator

We describe how the dynamics simulator executes ef£cient dynamics and kinematics computation of structure-changing kinematic chains, that include any kinematic chains, open or close, and even such kinematic chains as to change connectivity in operation [10].

Dynamics Computation of General Closed Kinematic Chains We proposed an algorithm to compute inverse/forward dynamics of general closed kinematic chains with or without actuation redundancy. The key idea was the general representation and computation of kinematic constraints that does not suffer from the singularity due to representation.

Description of Open Kinematic Chains via Pointers Three pointers for each link are used to describe open kinematic chains. The meanings of the pointers are illustrated in Fig.1. The *parent* pointer, points a link connected towards the base link. Conversely, the *child* pointer points a link connected towards the end-effector. Finally, the *brother* pointer, points a link with the same *parent*, in case the parent link has several links connected towards the end-effector.

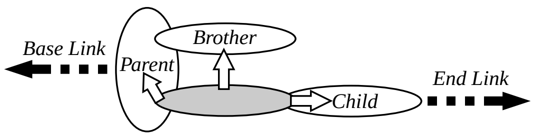

Fig. 1. Three pointers to describe open kinematic chains

The recursive computation was implemented using the three pointers and recursive call of functions. For forward path computation, the functions are called recursively for the *child* and *brother* links after £nishing the computation for itself. For backward path computation, on the other hand, recursive calls are done before the computation for itself. The recursive computations are ef£ciently done by using the pointers.

Description of Closed Kinematic Chains via Virtual Link The method in the previous section is applicable only to open chains since the parent-child relationship is ill-de£ned for closed chains.

First, as illustrated in Fig.2, we virtually cut a joint in each closed loop to avoid the ill-de£nition. Since the mechanism is no longer closed, its connectivity is described by the three pointers. Next, to maintain connection at the virtually cut joints, we add a *virtual* link at each cut joint, whose parent should be one of the two links connected by the cut joint. *Virtual* link is introduced only for describing a closed loop. It has kinematic properties such as joint values and link length, but no dynamic properties such as mass or inertia. It also has the corresponding real link. In order to indicate the real link of a virtual link, we introduce a new pointer called the *real* pointer, which points the real link from the virtual link. The *real* pointer is valid only for virtual links. Note that the description of a closed chain is not unique and depends on which in a closed loop is virtually cut for description.

To summarize our link con£guration notation, any open or closed kinematic chains are described by four kinds of pointers— *parent, child, brother* and *real*— and a *virtual* link corresponds to each closed loop. An example of description of a closed kinematic chain is shown in Fig.3.

The advantages of the representation are:

- Suitable for recursive algorithms for dynamics computation.
- Closed loops are easily identi£ed, since each closed loop has a virtual link.

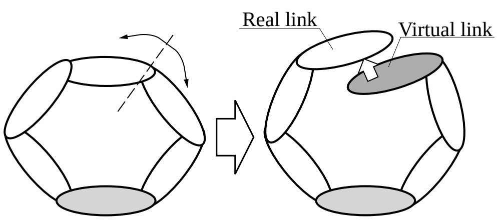

Fig. 2. Describing closed loop by virtual link

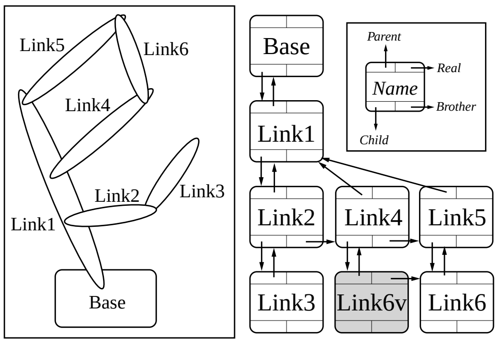

Fig. 3. Example of describing link structure

– The virtual cut joints for dynamics computation can be chosen as ones indicated by the virtual links.
– The data increases only proportional to the number of links.

An alternative method of the representation simply ignores the closing of the chain and uses external forces to to simulate the constraint. But then the method need either to compute the external forces by using the Lagrange multipliers or else, or to apply iterative computation for the purpose. The former has lacked the generality since the constraint equations had to be explicitly coded in the software, and the latter would involve heavy computation.

Structure-Varying Systems

Connecting Links First consider a case where two links are connected to create a new joint. A good example is that a hand catches a bar and allows free rotate about it. If a closed loop is generated by the connection as in the case illustrated in Fig.4, we just add a virtual link at the new joint. The procedure is quite simple:

1. Create virtual link *Link4v* whose real link is *Link4*.
2. Add *Link4v* to the data as a child of *Link3*.

which is easily programmed and processed on line. The descriptions of link connectivity before and after the connection are shown in Fig.5.

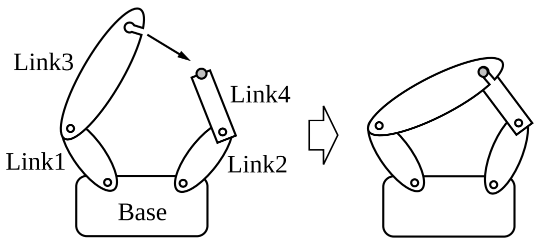

Fig. 4. Example of link connection

When a free-¤ying chain is connected to another chain, the situation becomes complex because closed chains are not always generated. See Fig.6, where a link named *Link1* of a free-¤ying chain of two links *Base* and *Link1* is connected to *Ground* and a new rotational joint is created.

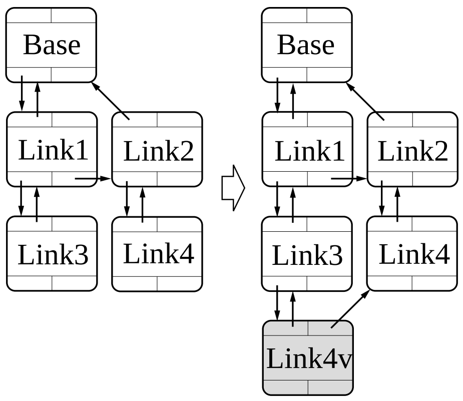

Fig. 5. Link structure description before and after connection

Considering that the structure after the connection is apparently an open chain, it seems natural to change the data as shown in Fig.7, where the remarks “*Rotate*” and “*Free*” indicate the joint types. Careful inspection on Fig.7 tells us, however, the parent-child relationship of *Base* and *Link1* is inverted, which requires modi£cation of the Denavit-Hartenberg parameters and some dynamic parameters. Although the amount of additional computation may not be very large, it will affect the total computation time especially for motions with frequent change of connection, such as walking.

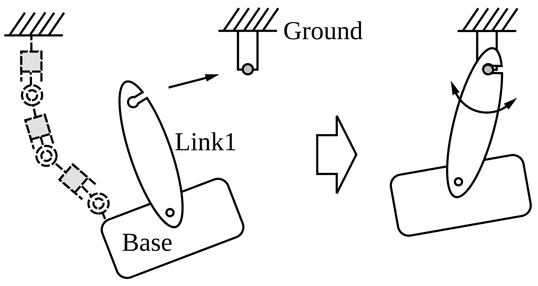

Fig. 6. Open chain generated by link connection

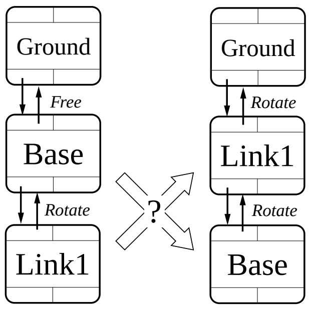

Fig. 7. Apparently possible change of link structure description

We treat this case exactly in the same way as in the previous one. In other words, we consider the new structure as a closed kinematic chain taking the *Free* joint between *Body* and *Ground* into account.

The procedure is the same as before: (1) create a virtual link of *Link1* and name it *Link1v* and (2) connect *Link1v* to *Ground* through the new rotational joint. Fig.8 shows the description of the new structure, where it is no longer necessary to reverse the relation of *Base* and *Link1*.

Handling link connection in this manner has several advantages:

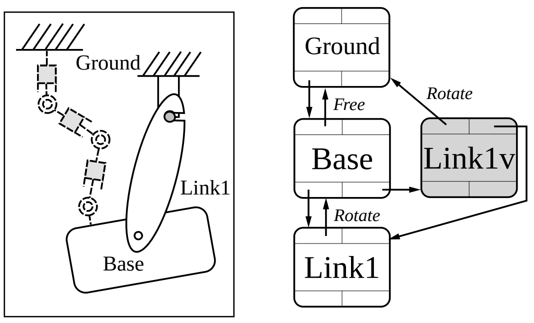

Fig. 8. Closed kinematic chain with free joint

- There is no need to reverse parent-child relationship of links, thus the overhead due to link connections is reduced.
- Arbitrary connection is treated in the same manner, whether the structure after connection is closed or open.

On the other hand, if the structure after connection is physically open as in the latter case, the amount of dynamics computation becomes larger than that when it is treated open. The increase is minimum since the number of links increases only by one.

Cutting Joints Presented below is the procedure for cutting a connection of two links at the joint between the two. Note that this means physical cutting, while the cutting in dynamics computation is virtual. If the cut joint is one indicated by a virtual link, the procedure is exactly opposite of connecting link. Suppose, in the structure after connection in Fig.4, the joint between *Link3* and *Link4* is cut, which is handled by deleting the virtual link, *Link4v*.

A humanoid is usually described as a chain with the hip link being the base link. Hands or foots may be connected to another object. A virtual link is created on every connection. As far as a humanoid is concerned, we can safely assume that cutting occurs only at the joints indicated by virtual links. In general kinematic chains, however, this is not always the case. Even if the cut joint is not indicated by a virtual link, con£guration change is handled readily by introducing a free joint. Suppose, in the structure after the connection in Fig.4, the joint between *Link1* and *Link3* is cut. The procedure in this case is: (1) cut the parent-child relation between *Link1* and *Link3*, and (2)connect *Link3* to *Base* by a free joint. The link structure and its description is shown in Fig.9. The connection between *Link3* and *Link4* is maintained by the virtual link *Link4v*.

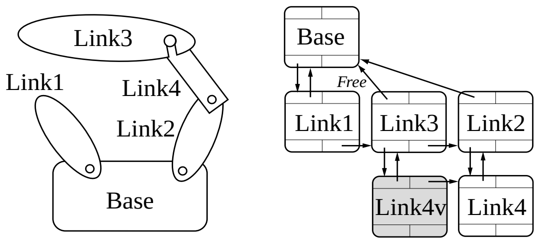

Fig. 9. Link structure and its description after cutting

Simulation examples Two simulation examples are presented here.

Walking Motion captured data of walking is used as a reference input to a control algorithm of walking. The captured and output motions are shown in Fig.10. The kinematics properties of the human £gure are set almost the same as those of real human, while the dynamics properties are quite different. It is observed that the vibration of the upper body in the result is larger than that in the original captured data. Kinematic errors of the foot motion in the original captured data are also corrected.

Jump The snapshots when our method was applied to jumping are shown in Figs.11.

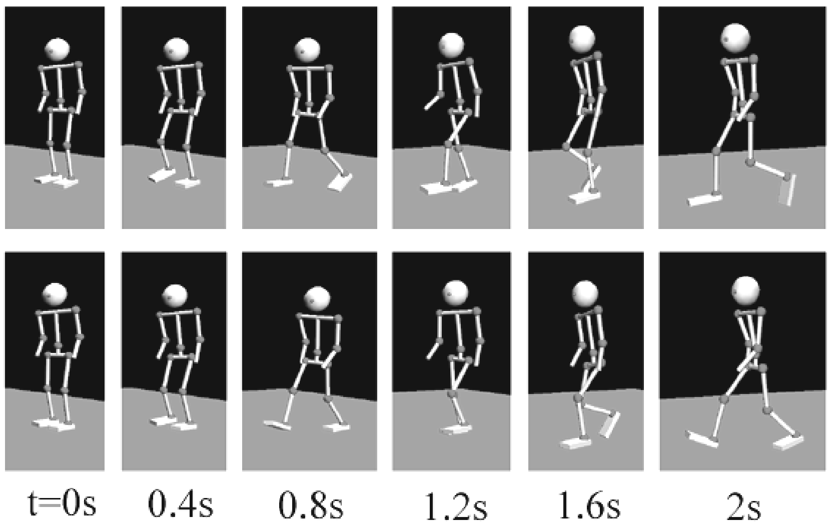

Fig. 10. Captured(above) and modi£ed(below) walking motions

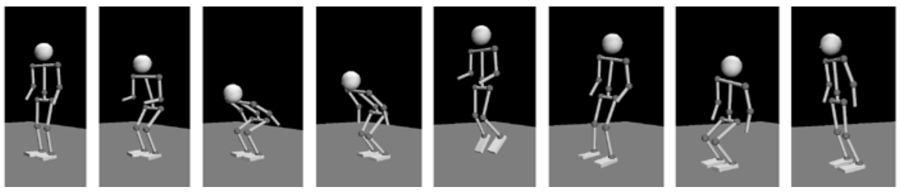

Fig. 11. Jumping

2.2 View simulator

Design of the View Simulator View image synthesis consists of three parts, that is, modelings of illumination, shapes and materials of objects in a scene, and cameras. Among them, illumination is relatively easier to model than the rest, since IES format data[5] can provide color, initial strength and ray distribution of many kinds of arti£cial light source. These data for natural light are also available. We employ IES data to model illumination.

The shapes of arti£cial 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 re¤ectance data from observation[8]. 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 [1]. Though the exact modeling of re¤ectance 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.

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 re¤ection 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 re¤ection. When the surfaces of objects in a scene is smooth like metal surface, the effect of specular re¤ection has become more dominant and the synthesized images look signi£cantly different from the corresponding real ones.

Considering these discussions, we have employed radiosity rendering algorithm for view image synthesis.

Evaluation of the Systhesized images We have applied two kinds of image processing to an example of the synthesized images and the corresponding real ones, to evaluate the view simulator. To get the real images, the parameters of the real camera, including its position, orientation and the angle of the £eld of the view, were tuned manually. Besides, the shapes of the real objects is slightly different from the corresponding geometric model. So the real image and its counterpart look different geometrically, but this slight difference does not spoil our comparison since the main point is the comparison of the illumination.

Differentiation and thinning The £rst column of Fig.12 shows gray scale images from a real camera, the radiosity rendering and a simple shading. The second column includes the corresponding images after differentiation respectively, and the third column does those after thinning. The biggest difference in the gray scale images is that the real image includes the effect of specular re¤ection that the radiosity one does not have. Besides, the brightness of some objects changes uncontinuously from their neighbors. This was caused by a coarse decomposition of the surfaces during the radiosity computation. Except for these differences, the real image and the synthesized image look similar.

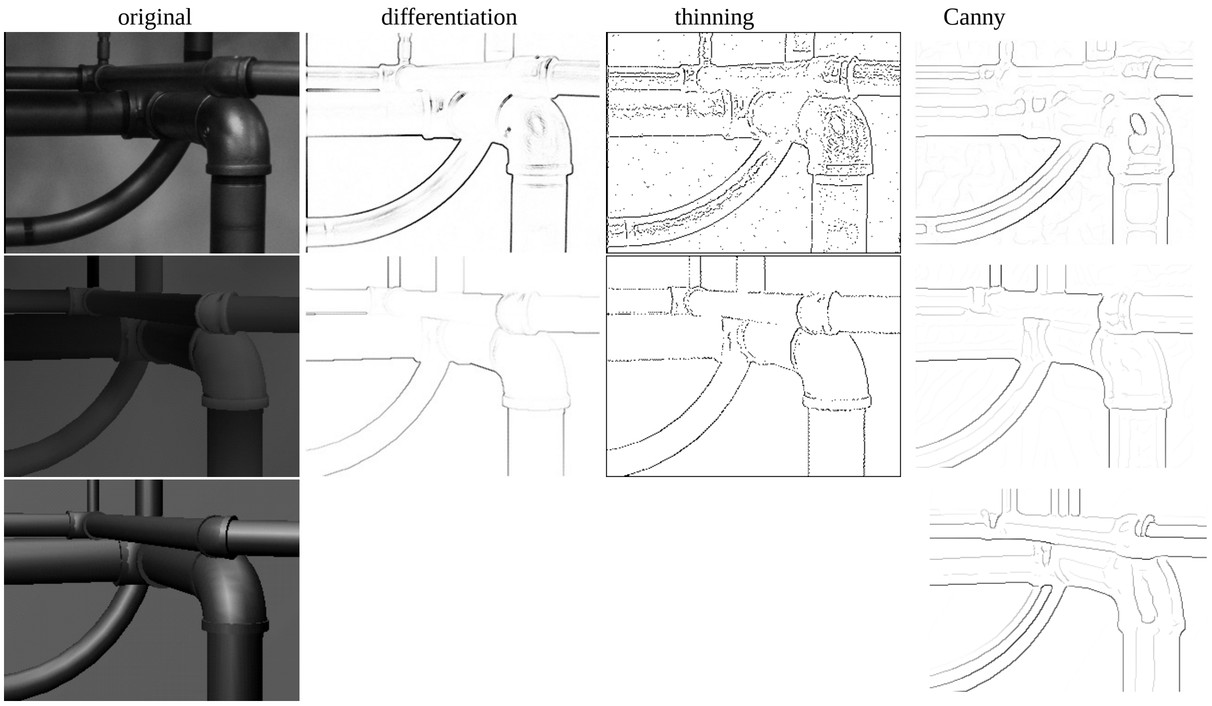

Fig. 12. Comparison between a real image and the corresponding synthesized one

We applied a simple differential operator, which £nds the sum of absolute values of horizontal and vertical differences, to the gray scale images. The results from both images are almost identical except for the specular areas. Next, thinning was applied to the differential images, which eliminates non-maximum value pixels. The result from the synthesized image has little noise, while the result from the real one includes a lot of noise. But the contours of the objects are clear in both results. For feature abstraction from these pre-processed images, we usually use robust methods, which are not affected by the noise very much. Therefore, the difference in the thinned images are not serious, and we can have the features of the synthesized images that are not signi£cantly different from those of the real ones.

Canny operator Next, we compare edges detected in the real, synthesized and shading images by the Canny operator[2]. The results are shown in the fourth column of Fig.12.

At £rst, take a look at the shading image and the detected edges in the £nal row of Fig.12. The detected edges bounding specular re¤ections correspond very well to those in the real image. The slight differences in the shape and position of the specular edges, especially the existence of specular edges on the vertical pipes at the upper part of the image, originate from the positioning of the light sources. In the real scene, lighting comes from an array of ¤uorescent lamps on the ceiling, but since it is only possible to set up a few light sources when making the shading image, we use a directional light from the ceiling towards the ¤oor and a supplementary point light next to the plant to illuminate the vertical wall. On the other hand, the edges of occluding contours look fairly different at the upper parts of the horizontal pipes. This is because shadows generated by the large pipes above are not considered in the shading image which gives more brightness to the upper half of two pipes and less one to the lower but the brightness of the upper and lower halves of the horizontal pipes is quite similar in the real image due to the shadow of the large pipes above.

In the case of the radiosity image, the edges produced by specular re¤ection are completely neglected as radiosity considers only Lambertian re¤ection. However, the edges of occluding contours, especially in the case that the wall forms the background, look very similar to those in the real image, because the effects of shadows are considered. The differences at occluding contours where the background is formed by other pipes and the edges observed at joints are caused by the coarse decomposition of the surfaces during the radiosity computation.

It would be ideal to be able to consider both the effect of shadows using the radiosity method and the effect of specular re¤ection using the shading method. If the radiosity method can be improved such that all edges except for those generated by specular re¤ection are equivalent with the ones in the real images, then evaluation of image processing algorithms could be done by using the shading image for the specular edges and the radiosity image for the other edges.

Simulation example A snapshot of the view simulation is shown in Fig.13. The upper-left picture is a bird view, and two upper-right pictures are the £elds of view from the left and right eyes respectively. Two lower-left pictures are the derivative image of the left view and thinned image of the right view respectively.

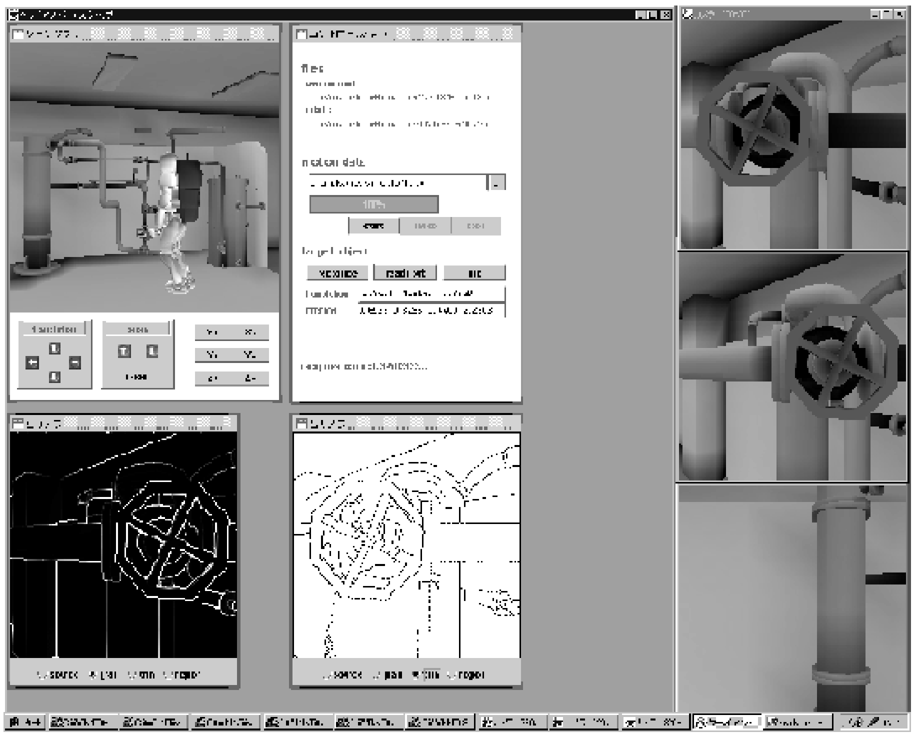

Fig. 13. Snapshot of the view simulator

3 Humanoid Motion Controller

3.1 Biped Locomotion

Biped Locomotion adaptive to Terrain We have developed a motion controller for biped locomotion adaptive to the terrain, including walking straight, turning, going up/down the stairs, and walking on the rugged ground, whose basic algorithm was given in [9]. With this programming library, complex locomotion can be realized as a sequence of these basic motion patterns. The linking motion between the basic motions of the robot is also automatically generated for the continuous motion control. The control data for the walking adaptive to terrain generated by the library has been examined and proved to be consistent with the mechanical dynamics using our dynamics simulator. The data consistency has also been proved to control the testbed hardware model, to walk using the motion controller.

We have also developed a leg-shaped mechanical test-bench with six degrees of freedom to improve the accuracy of the control algorithm. The accuracy of the simulation has been veri£ed in more details by taking the delay of the joint response into account, which was observed between the simulation results and the experiments.

The con£guration of the programming system is illustrated in Fig.14.

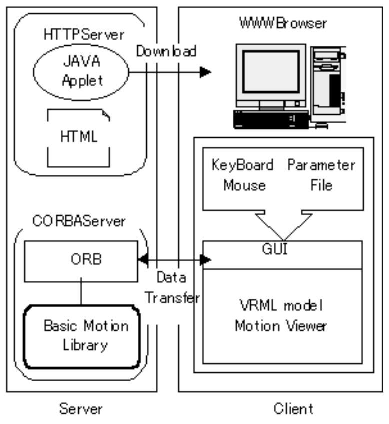

Fig. 14. Con£guration of the programming system for the locomotion

Network Programming Interface for the controller We have developed the graphical user interface for the programming system through LAN/WAN. A user can easily make the robot motion control sequence with built-in block programming functions, generate the motion control data using the CORBA-connected network modules on-line, then visualize the robot motion using the corresponding VRML robot model. The developed interface has been designed to help complex operation such as choosing the menus of the controller, specifying the control parameters, and testing the sequence of the motion through the network.

It can also generate the necessary scripts and £le, for specifying parameters such as the total simulation time and so on, to use the dynamics simulator automatically. This network interface provides a seamless system operation in generating a walking motion sequence, verifying with the mechanical dynamics simulation, and con£rming the robot motion using a VRML browser. The network programming interface is built on a Web browser, and its outlook is shown in Fig.15.

3.2 Dynamic Balance Control of a Humanoid Robot at the Standing Position

The basic idea of the balance control is a direct feedback of the total angular momentum and the position of the center of gravity as the state of the entire robot system. The total angular momentum $\mathbf{L} \equiv [L_x, L_y, L_z]$ can be calculated by

$$\mathbf{L} = \sum_i (\mathbf{r}_i \times m_i \frac{d}{dt}\mathbf{r}_i + \mathbf{R}_i \mathbf{I}_i \omega_i), \tag{1}$$

where $\mathbf{r}_i$:position vector of the center of gravity of the i-th link, m_i:the mass of the i-th link, $\mathbf{R}_i$: orientation matrix of the i-th link frame, and $\mathbf{I}_i$, ω_i: inertia tensor and angular velocity in the i-th link frame respectively. $\mathbf{L}$ can be

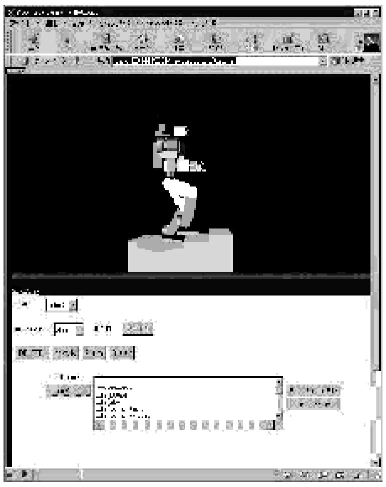

Fig. 15. Outlook of the network programming interface

calculated in real-time from the absolute posture and the angular velocity of the robot body measured by gyro sensors and from joint velocities measured by encoders.

The position of the center of gravity, $\mathbf{r}_G \equiv [r_{Gx}, r_{Gy}, r_{Gz}]$ is given by

$$\mathbf{r}_G = \sum_i m_i \mathbf{r}_i / M, \tag{2}$$

where M is the total mass of the robot. Then the dynamics of the entire robot system can be represented by the Euler's law of motion,

$$\frac{d}{dt}\mathbf{L} = M\mathbf{r}_G \times \mathbf{G} + \tau, \tag{3}$$

where $\mathbf{G}$ is the gravity acceleration vector and $\tau \equiv [\tau_x, \tau_y, \tau_z]$ is the ground contact moment. This equation shows us how the total angular velocity changes under the given ground contact moment.

Using ankle actuators of the supporting leg and the torque sensor embedded in the corresponding foot, we can control contact moment with suf£cient accuracy. So, we regard the reference contact moment as the input to the system of Eq. (3) to control the angular momentum and the position of the center of mass. The objective of the balance control is to realize $L_x = L_y = 0$ and $r_{Gx} = r_{Gy} = 0$, then one of the simplest feedback laws can be written by

$$\begin{aligned} \tau_x^d &= -k_{px} L_x - k_{vy} r_{Gy}, \\ \tau_y^d &= -k_{py} L_y - k_{vx} r_{Gx}, \end{aligned} \tag{4}$$

where k_{**}s are feedback gains. In this control, only the ankle actuators of the support leg are used for the balancing, and we can arbitrary specify the motions of all other joints. This is a great advantage of the proposed control method. A similar feedback law was introduced by Sano and Furusho for the control of dynamic biped walk[7]. We have implemented the feedback law given by Eqs.(1), (2) and (4) on the dynamics simulator. Physical parameters of the testbed hardware were used for those dynamic simulations.

In the £rst example, the robot is standing with two legs, and both of legs are controlled in the same manner. Only the balance control for pitching motion (around y-axis) was applied since the robot was stable around x-axis. Under the proposed balance control, the robot could successfully sit down, reach arms to the ground and stand up again. All joints except ankles were position-controlled to generate the desired motion. A snapshot of the motion is illustrated in Fig.16.

In the second example, to demonstrate a three-dimensional balance, a kicking motion was tested (See Fig.17). The robot made full swing of the left leg in one second while balancing with the right leg. Under the proposed control, the robot was successfully kicked and balanced. The motion of the arms and the body was added just for natural outlook, but those were not necessary to keep the balance. All compensation was done by the ankle actuators of the supporting leg.

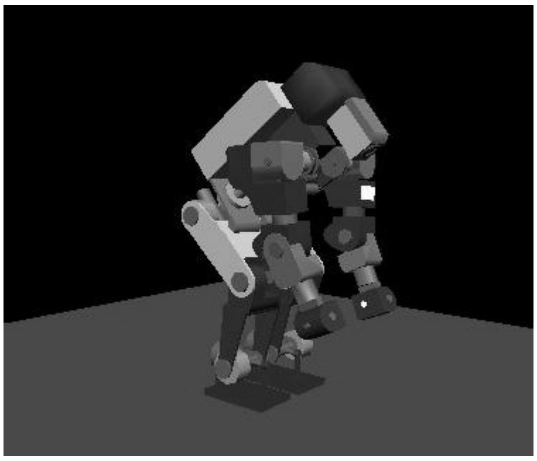

Fig. 16. Sitting down to pick up an object from the ground

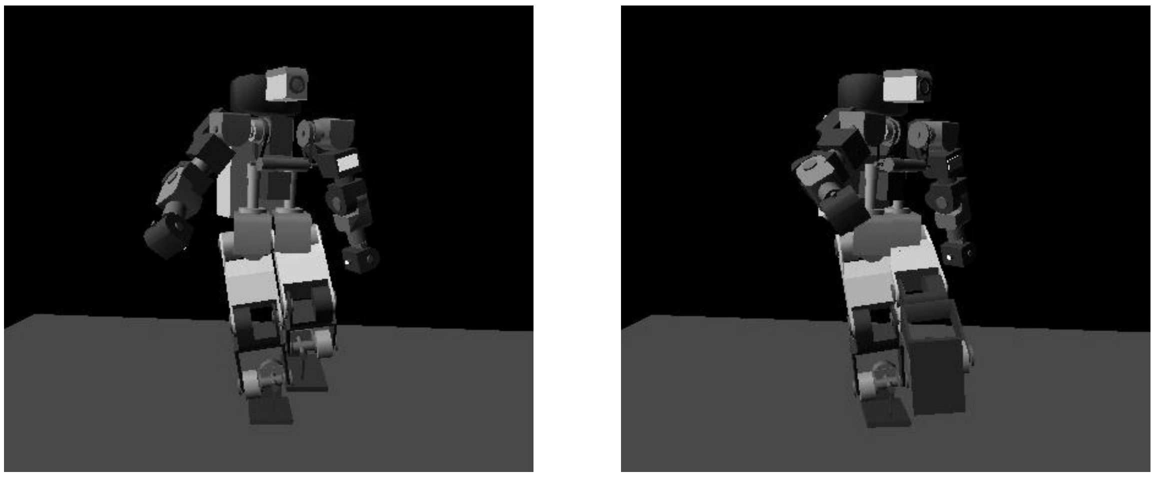

Fig. 17. Kicking motion

3.3 Collision Avoidance Motions

A humanoid robot has many degrees of freedom, but the planning of collision avoidance motions of a humanoid robot must take the dynamic balance into account as well as obstacles. We have investigated how to assign the degrees of freedom of the robot to the collision avoidance and the balancing. The basic design of the algorithm are

- Motions are planned such that one arm, supposed to execute some task, should avoid collision with the working environment of the robot.
- Motions are planned by using six joints, that is, LEG[3](hip), LEG[4](knee), ARM[1],[2],[3](shoulder) and ARM[4](elbow). See Fig.18. LEG[3] and LEG[4] are assumed to move identically for the left and right legs.
- The planned motions are represented by the trajectories of the six joints as well as the trajectories of the position and orientation of the shoulder joints.
- The balance of the body is controlled by the algorithm described above using LEG[5],[6](two ankle joints).
- The deviations of the shoulder position from the planned trajectories due to the balancing are cancelled by moving joints LEG[1]-[4].
- Those of the shoulder orientation are cancelled by modifying the planned trajectories of joints ARM[1]-[3].

LEG[3], LEG[4] and ARM[4] are selected for the motion planning, since the movable ranges of these joints are relatively wider. The sweeping volume of the arm under the planned trajectory remains identical even when the balance control is applied, since the above cancellation works. The underlaying idea exists in the observation that three axes of rotation of the shoulder joint meet at one point.

The top level of the motion plannine algorithm is the following.

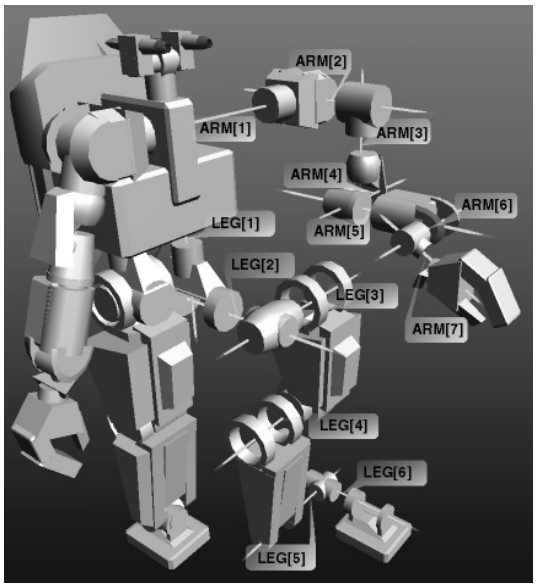

Fig. 18. Con£guration of the joints of a humanoid robot

1. Set joints ARM[5],[6],[7] to £xed values which are supposed to be appropriate for the execution of some task.
2. Set joints LEG[1],[2],[5],[6] to £xed values corresponding to the orthogonal standing position.
3. Plan collision avoidance motions of the robot by using LEG[3], LEG[4], ARM[1],[2],[3] and ARM[4]. Assume here that the relative position between the robot and the ¤oor does not chance while the planning.
4. Move ARM[4] and LEG[1]-[6] while £xing ARM[5],[6],[7] such that the position of the shoulder joint should follow the planned trajectory and the body should be balanced. At that time, the orientation of the shoulder joint may deviate from the plannce trajectory, which can be cancelled by modefying the trajectories of ARM[1],[2],[3].

The randomized roadmap algorithm[4] is applied for Step3 currently, but any motion planning algorithm must work in principle. The proposed algorithm has been implemented. We use RAPID[3] for the collision detection step in the algorithm. A simulation example is illustrated in Fig.19, where the robot is trying to avoid the pipes ahead before capturing the handle.

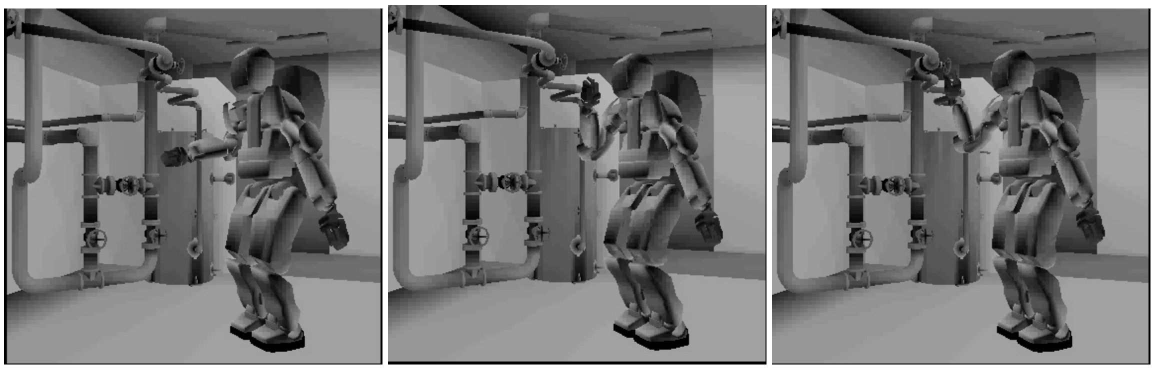

Fig. 19. Snapshots of a collision avoidance motion

4 System Architecture

The con£guration of V-HRP is illustrated in Fig.20. V-HRP consists of the dynamics simulator, the view simulator,

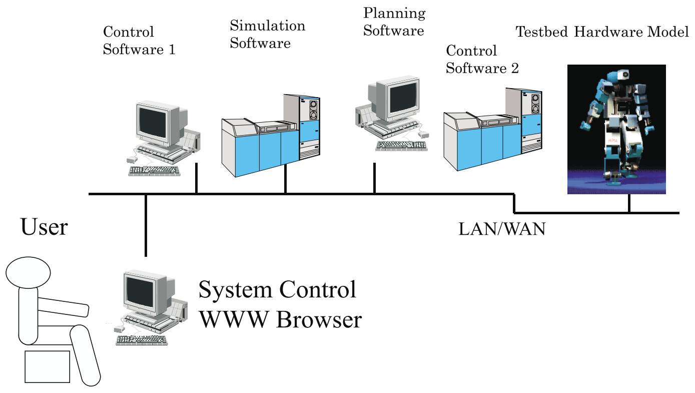

Fig. 20. Con£guration of V-HRP

and the controllers for the biped locomotion, dynamic balancing and collision avoidance, each of which is implemented as a CORBA server. The user interface is constructed on a Web browser. Thanks to the architecture of the system, these software modules can be run on on distributed platforms, possibly on different operating systems, and the users can access the system via LAN/WAN if they can reach the servers by IIOP.

The users can implement their application codes in any language, if the language is binded to an implementation of CORBA. For example, ORBacus **(http://www.ooc.com)** supports C/C++ and Java.

5 Comparison between the Simulation and Experiments

We have developed a small humanoid robot or testbed, to verify the validity of the dynamics simulator. The testbed has 6 d.o.f. for each leg, 7 d.o.f. for each arm and 26 d.o.f. in total, which is identical with the humanoid robot platform developed in HRP. The weight of the robot is about 8kg and the height is 540mm. The robot also has a posture sensor, foot sensors and CCD camera. The controllers of the actuators are distributed in the body, and USB is used as the internal network. The appearance of the testbed is shown in Fig.21.

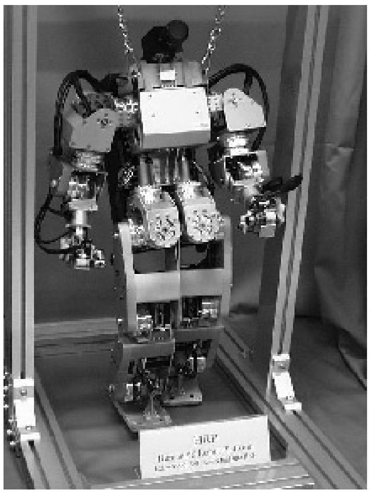

Fig. 21. Humanoid testbed

Figure22 shows an example of the comparison between the dynamics simulation and the veri£cation experiment by the testbed robot. The trajectories of the roll and pitch angle of the body are drawn. Though the basic trend of the trajectories are not so far, the simulator must be brushed up to have more accurate results.

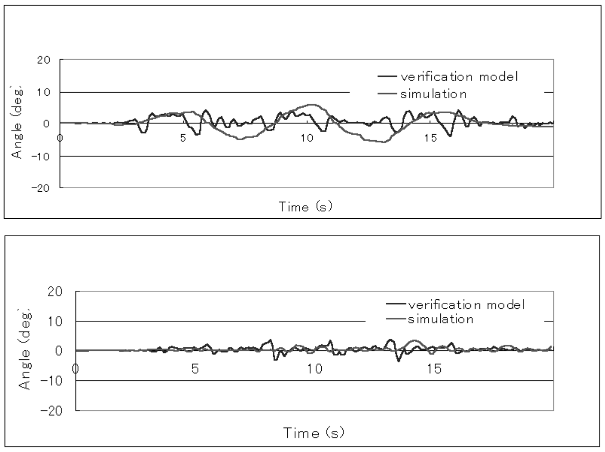

Fig. 22. Comparison between the simulation and the experiment

6 Conclusions

We have developed Virtual Humanoid Robot Platform, V-HRP, as an infrastructure for humanoid robotics research. The features of V-HRP include

- seamless dynamics simulation of humanoid robots when the structure changes between open/closed kinematic chains,
- synthesis of the £eld-of-view of humanoid robots for the simulated dynamics motion,
- motion planners and controllers as the basic library including biped locomotion, dynamics balancing at the standing position and collision avoidance motions,
- network distributed computation on LAN/WAN which allows users to start with the minimal computer resource,
- testbed humanoid robot developed for quantitative evaluation of the simulator.

The V-HRP is to be widely used under the project as the foundation for exploiting applications and accumulating developed algorithms and software of humanoid robots.

Acknoledgements

This research was supported by the Humanoid Robotics Project of the Ministry of the International Trade and Industries, through the Manufacturing Science and Technology Center.

References

1. Asada,N., Baba,M. and Amano,A., Calibrated computer graphics: a new approach to realistic image synthesis based on camera calibration, Proc. Int. Conf. on Pattern Recognition, pp.705-707, 1998.
2. Canny J., A computational approach to edge detection, IEEE Trans. on Pattern Analysis and Machine Intelligence, vol.PAMI-8, no.6, pp.679-698, 1986.
3. Gottschalk,S. et al., OBB-Tree: A hierarchical structure for rapid interference detection, proc. of ACM Siggraph, 1996.
4. Kavraki,L.E., Svestka,P., Latombe,J-C. and Overmars,M.H., Probablictic roadmap for path planning in high-dimensional con£guration space, IEEE Trans. Robotics and Automation, vol.12, no.4. pp.566-580, 1996.
5. IES Lighting Handbook, Application Volume, 1981.
6. Nakamura,Y. and Ghodoussi,M., Dynamics computation of closed-link robot mechanisms with nonredundant and redundant actuators, IEEE Trans. Robotics and Automation, vol.5, no.3, pp.294-302, 1989.
7. Sano,A. and Furusho,J., Realization of natural dynamic walking using the angular momentum information, Proc. of IEEE Int. Conf. on Robotics and Automation, pp.1476-1481, 1990.
8. Sato,Y., Wheeler,M.D. and Ikeuchi,K., Object shape and re¤ectance modeling from observation, SIGGRAPH-97, pp.379-387, 1997.
9. Jin'ich Yamaguchi, Atsuo Takanishi, et. al. Development of a bipedal humanoid robot - control method of whole body cooperative dynamic biped walking, Proc. IEEE Int. Conference on Robotics and Automation, pp. 368, 1999.
10. Yamane,K. and Nakamura,Y., Dynamics computation of structure-varying kinematic chains for motion synthesis of humanoid, Proc. of IEEE Int. Conf. on Robotics and Automation, pp.714–721, 1999.