Proceedings of the 2005 IEEE
International Conference on Robotics and Automation
Barcelona, Spain, April 2005

Real-time and Precise Self Collision Detection System for Humanoid Robots

Kei Okada Masayuki Inaba Hirochika Inoue
Graduate School of Information Science and Technology, University of Tokyo
#701, Engineering Building No. 8, 7-3-1, Hongo, Bunkyo-ku, Tokyo, Japan
k-okada@jsk.t.u-tokyo.ac.jp

***Abstract*— In this paper, we describe the real-time and precise self collision detection system that does not reduce the number of polygons and checks more than 100 collision pairs in real-time by using AABB based collision detection libraries.**

Previous researches on collision detection of humanoid robots which reduce collision pairs or simplify a shape of a robot has disadvantages such as increasing the dangerousness or decreasing range of movement. However our self collision detection system uses detailed geometric model and collision pairs as many as possible.

We have experimentally evaluated collision detection libraries on a real-time self collision detection application of a humanoid robot. This experiment suggests that AABBs based method is much faster than conventional OBBs based method. Finally, we demonstrated real-time collision detection and avoidance function that automatically stops entire motion if self collision occurs using HRP2 humanoid robot.

***Index Terms*— self collision detection, humanoid robot**

I. Introduction

Recently as a highly stable walking control technology of a humanoid robot progresses, there has been increasing interest in sensor based whole body behavior control. On the contrary to conventional behaviors of a humanoid robot such as carefully pre-programmed walking or gesturing motions, a sensor based behavior of a humanoid robot which motions of a robot changes on the fly, causes a "self collision" that two or more links of a robot collide. A self collision yields serious damage to a robot itself. Thus, a software for detecting self collisions in real-time is a fundamental and essential function for developing a sensor based behavior of a humanoid robot safely.

There has been very little work on real-time self collision detection for a humanoid robot. A challenge of real-time self collision detection is it's computation cost. For example, in the case of a humanoid robot with 31 links, collision pairs to be checked becomes 435 pairs, each link has about 1,000 polygons and the total number of polygons becomes more than 30,000. Therefore, in order to archive a real-time self collision detection system, current computation resource is not sufficient.

There are two approaches for realizing real-time self collision detection. One is to reduce collision pairs and the other is to simplify a shape of a robot. Real-Time collision detection system developed by Kanehiro et al. [1] reduce collision pairs from 350 to 36 for a 29 DOF humanoid robot using online and offline hybrid approach. Their robot has about 10,000 polygons in all. Another approach is proposed by Kuffners et al. [2]. They reduce polygon model of a robot by transforming from polygon soup(314,588) to convex hull(2,702). They also reduce collision pair to be checked from 435 to 76.

It is obvious that both approach has disadvantages. Reducing collision pairs to be checked increase the dangerousness of self collisions. Reducing the number of polygons in a robot model causes a wrong collision detection result. Especially, in the case of current humanoid robots with complex shapes, approximating complex shapes to convex hull is fatal.

In this paper, we present real-time and precise self collision detection system that does not reduce the number of polygons and checks more than 100 collision pairs in real-time by using AABB based collision detection libraries.

II. Real-time Self Collision Detection for Humanoid Robot System

In this section, we describe the design of a self collision detection software for the HRP2 Humanoid Robot System [3]. Although we employ the HRP2 robot system as a hardware and software platform in this paper, our system design is so general that the contents of this paper are able to apply to other humanoid robot system.

A. General framework for self collision detection software

Fig.1 shows the general framework of self collision detection system for a robot. Following two information are required for finding collided object pair.

3D geometric model information for objects
: Usually it requires polygon triangles of the object surfaces.

Transformation matrix of each object
: For updating current coordinates (the position and rotation) of the object.

Then we are able to perform collision query to see if there exists collided objects using the 3D geometric model information and the transformation matrix information.

A humanoid robot manufacturer usually provides the robot model data which contains the 3D geometric model of each link and the joint or kinematics information that which link pair are connected each other. Therefore the software for loading this model data is to be developed, which is illustrated as a "Model Loader" in the figure.

TABLE I

SUMMARY OF PREVIOUS WORKS OF REAL-TIME SELF COLLISION DETECTION SYSTEM FOR HUMANOID ROBOTS

	DOFs	pairs	Polygons	Library	Comp. Time	Note
Kanehiro('01)	29	35	10,000	RAPID	0.3 - 0.6 [msec]	P3 933Mhz
Kuffner('02)	31	76	2,702	V-Clip	0.429 [msec]	P3 866Mhz Dual
This Work('04)	31	109	36,053	RAPID/ ColDet/ OPCODE	1.5 (20.0) [msec] 1.0 (4.5) [msec] 0.7 (2.5) [msec]	P3 1266Mhz ():worst case

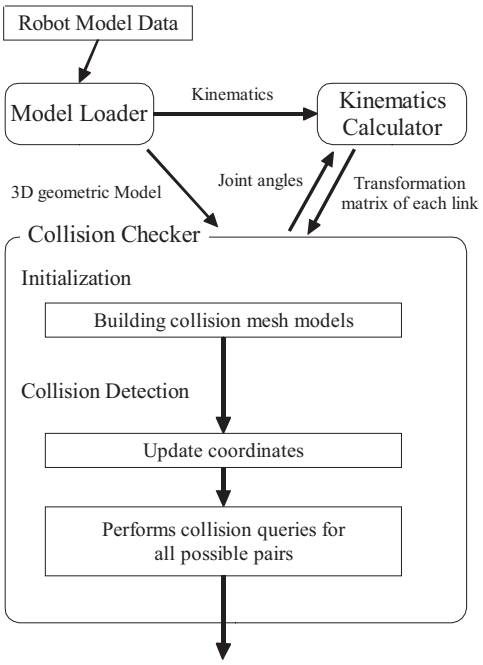

Fig. 1. General framework for self collision checking software

The another required software is "Kinematics Calculator" which calculates a current transformation matrix from the joint angle data of a humanoid robot by using kinematics information of a robot.

B. Self collision checking software on HRP2 system

In the case of HRP2 which is based on the OpenHRP control system, several functions are implemented as a CORBA server that includes the ModelLoader server for reading the model data of a humanoid robot and the Dynamics server that provides forward kinematics calculation using the loaded model data. A runtime software for the HRP2 robot must written as a plugin that is loaded into the Controller server and runs at every 5[msec] cycle. Each plugin receives current sensory data and joint angle data and output reference angle data for each joint. Since the cycle of the Controller is 5 [msec] and we assume that acceptable computation time for collision checking calculation is 1[msec] as described in the previous work [1].

C. Collision check pair reduction

One of the difficulties of self collision checking for humanoid robots is that it consumes computation time. Since there exists a large number of object pair to be checked and each object (link) has very complex shape.

Let N to be number of all links of a humanoid robot, the number of pairs of the links is ${}_NC_2$. Assuming that collision between a given link and its parent link never occurs because of joint limits, there still ${}_NC_2 - (N-1)$ object pair to be checked. In the case of the HRP2 humanoid robot which has a total of 31 links, collision pairs to be checked become 435 according to the equation ${}_NC_2 - (N-1)$.

Several researchers have proposed the method to reduce collision check pair. Kanehiro et al proposed hybrid collision check approach [2] that pre-compute the table of colliding pairs in off-line and check collision only for collision candidate pairs in on-line. By using this method, they reduce collision pairs of the HRP-1S robot from 350 to 36. Kuffner et al uses heuristics for reducing collision pairs of their H7 robot from 435 to 76.

In this paper, we also use our heuristics for reducing collision pairs from 435 to 109. Our heuristics are listed below. Fig.2 shows correspondence table between a link name and a figure.

- Links in a leg never collide with links in the same leg.
- Links in a leg never collide with links in a torso, a chest and a head.
- Links in a arm never collide with links in the same arm.
- Links in a shoulder part never collide with any other links.
- A head yaw and a waist yaw never collide with any other links.
- A crotch yaw link, a crotch roll and ankle pitch link in a leg never collide with any other links, but a crotch yaw link only collide with a crotch pitch link in the same leg.
- A shoulder pitch and yaw links only collide with lower arm links in another arm.
- A neck pitch link never collide with any links in the legs and the torso link.

D. Polygon reduction

The number of polygons in a robot model also causes difficult issue. Since the detailed 3D geometric model of the

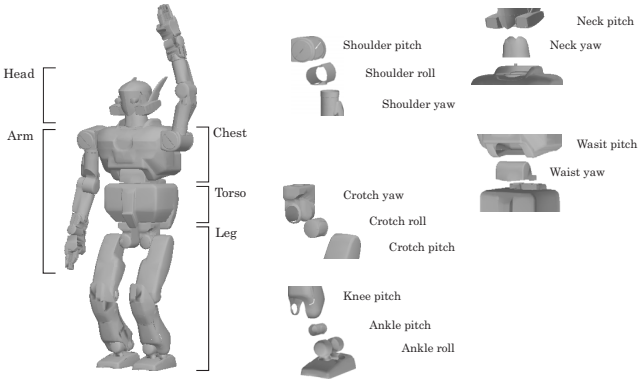

Fig. 2. Name of each links and it's correspond figure

TABLE II
NUMBER OF TRIANGLES OF HRP2 HUMANOID ROBOT

Link Name	tri.	Link Name	tri.
LLEG_JOINT0	715	LARM_JOINT0	1386
LLEG_JOINT1	80	LARM_JOINT1	537
LLEG_JOINT2	1136	LARM_JOINT2	1330
LLEG_JOINT3	2052	LARM_JOINT3	531
LLEG_JOINT4	64	LARM_JOINT4	1859
LLEG_JOINT5	1064	LARM_JOINT5	1983
RLEG_JOINT0	715	LARM_JOINT6	453
RLEG_JOINT1	80	RARM_JOINT0	1386
RLEG_JOINT2	1136	RARM_JOINT1	537
RLEG_JOINT3	2052	RARM_JOINT2	1330
RLEG_JOINT4	64	RARM_JOINT3	531
RLEG_JOINT5	1064	RARM_JOINT4	1859
CHEST_JOINT0	490	RARM_JOINT5	1983
CHEST_JOINT1	3557	RARM_JOINT6	453
HEAD_JOINT0	356	HRP2_WAIST	2203
HEAD_JOINT1	3068	Total	36054

each link requires computation time for collision checking, approximate model are used. However approximate model reduces the range of movement of the robot. For example, Kuffner et al uses convex hull of each link in their work [2]. They reduce a total number of polygons in a robot model from 314,588 to 2,702 triangles. In this work, we use detailed 3D model that has 36,054 for not loosing a range of movement. Table.II shows a number of polygons of each link of the HRP2 robot.

III. AABB BASED FIRST COLLISION DETECTION ALGORITHM

A. Collision Detection (CD) Library

Research on collision detection (CD) algorithm has a large and extensive history in the computer geometry literature [4], [5].

Existing software packages for collision detection are classified into two types. One is exact collision detector that is able to finds collisions between non-convex polygon soups that a object with hulls and another is that only detects collisions between convex hulls. In this paper we only tested exact collision library for non-convex polygon objects since approximating complex humanoid robot shape to much simpler convex hulls may reduces the range of movement of each joints.

Exact collision detector initially computes bounding volume hierarchies of the original polygon model. Then if any bounding volume pair intersects, it calculates whether the original objects are overlapped. Many researchers have been proposed representations for bounding volume as followings:

AABB(Axis-aligned Bounding Box)
: An Axis-aligned Bounding Box (AABB) is a rectangular bounding box at an axis aligned orientations. AABBs based bounding volume hierarchies implementations reduce the memory requirements. This approach generally yields faster collision queries than other bounding box volume representations. ColDet [6] and OPCODE [7] adopt this approach. ColDet supports timeout setting that is to limit detection time.

OBB(Oriented Bounding Box)
: An Oriented Bounding Box (OBB) is a rectangular bounding box at an arbitrary orientations in 3D space. OBBs generally allow geometries to be bounded more tightly with a fewer number of boxes compares to AABBs. RAPID [8] and V-Collide [9] packages uses OBBs. RAPID is widely used CD library. It finds a list of the intersecting triangle pairs. V-Collides is an "Nbody" processor built on top of the RAPID system. It only reports which object pairs collide while RAPID tells which pairs of triangles collide.

RSS(Rectangle Swept Sphere)
: A Rectangle Swept Spheres (RSS) corresponds to the set of points obtained by sweeping the center of a sphere over a 3D rectangle, that is the Minkowski Sum of and origin-centered sphere and an arbitrary oriented rectangle. PQP [10] uses this representation. The advantage of using RSS is the performance of distance and approximate distance queries between objects.

k-DOPs(Discrete Orientation Polytopes)
: A Discrete Orientation Polytopes (k-DOPs) is bounding volume whose faceset have normals from a given discrete set of k vectors to approximate input models. QuickCD [11] uses this approach.

On the other hand, collision detection library for convex hulls are classified as the feature based method on the Lin-Canny algorithm or the simplex based method on the GJK algorithm. V-Clip, I-Collide [12], SWIFT [13] uses features based method and Enhanced GJK [14] uses simplex based method.

B. Performance Evaluation of Real-Time Self Collision Detection System

Fig.3 plots the performance of self collision checker using several CD libraries in the HRP2 Humanoid Robot System. We have build real-time self collision checking

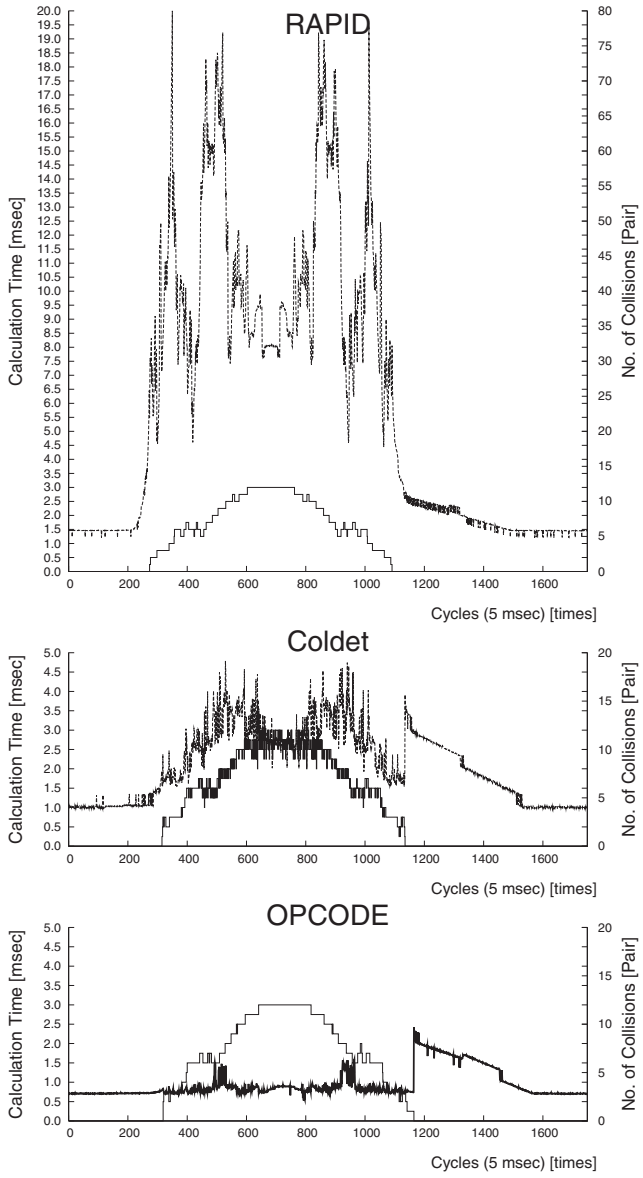

Fig. 3. Performance Comparison of Collision Checking Computation Time

plugin on 1266Mhz PentiumIII with 512KB Cache and 512MB Memory running ART Linux [15]. Collision link pairs to be checked are 109, a total number of polygons is 36,053 and a number of polygons in each link are shown in Table.II.

The input motions used for this evaluation is also used in next section. Fig.4 shows this motion.

Computation time for collision detection is shown by the left vertical axis of Fig.3, a number of collided links is shown by the the right vertical axis and elapsed time is shown by the horizontal axis.

Computational time for detecting collisions using RAPID library is constant (1.5[msec]) when there is no self collided links. However when there exits self collisions, it takes more computation time. It becomes 20[msec] in the worst case.

Computational time for detecting collisions using ColDet library is also constant when there is no self collided links. However when there exits self collisions, it takes more computation time. It takes 1.0 [msec] when there is no collision, and it becomes 4.5[msec] in the worst case. When two collided objects uncouple, it takes more computation time than when two object collide.

In the case of using OPCODE library, computation time when there exists collision links or time with no collisions are not same as other libraries. However computation time when collision occurs is not so slow. It takes 0.7 [msec] when there is no collision, and it becomes 2.5[msec] in the worst case. In the case of humanoid robot self collision detection system, the robot does not move when self collision occurs, there for 2.5[msec] computation time while colliding is allowable.

Throughout above experimental results, AABB based implementation is the fastest and have sufficient performances.

IV. Self Collision Detection and Avoidance Experiment

We build self collision detection and avoidance plugin for the HRP2 Humanoid Robot system using AABB based OPCODE library. In the control function, it checks if collision occurs among links at every control cycle using current joint angle information. This plugin send input reference angle data to the output, when no collided links found. It output stored data which is previous collision-free output data when self collision occurs. When collided links uncouple, it outputs interpolating data from the stored data to the current data. We added a safety margin around each link.

Fig.4 shows the input motions for checking developed solf collision detection software. The robot try to close it's arms as shown in figures. This motion results self collision that one hand get into another hand. Our self collision software successfully detect self collided links that have red color in the figures. Fig.5 shows the motion of the real robot. The input motion occurs collision between hands, however the motion of the real robot stops just before it collide.

V. Conclusions

In this paper, we described real-time and precise self collision detection system that does not reduce the number of polygons and checks more than 100 collision pairs in real-time by using AABB based collision detection libraries.

We have presented experimental performance comparison of collision detection library on self collision detection experiment using HRP2 Humanoid Robot System. Through our comparison experiments, AABBs based method is much faster than conventional OBBs based method.

Throughout the experience of developing and evaluating the collision detection software, we have noticed that there are two cases in self collisions. First case is a collision in a compound joint such as a hand and a lower arm or a

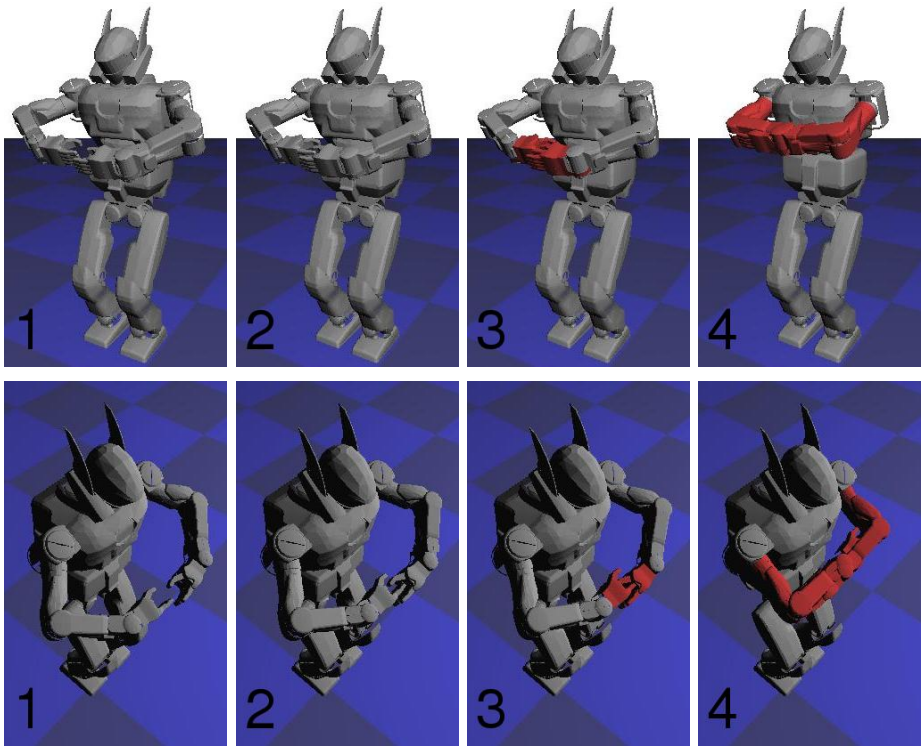

Fig. 4. Self Collision Detection Results

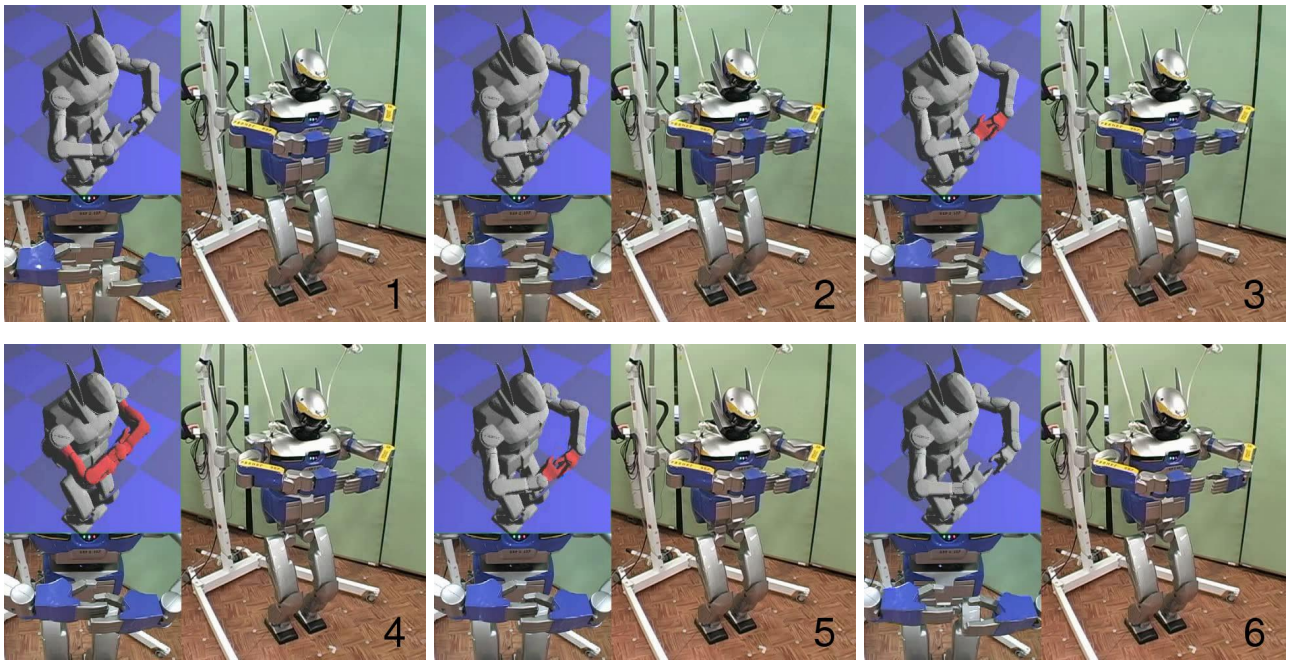

Fig. 5. Motion of HRP2 Robot with Self Collided Motion

chest and a head, another one is a collision between an non compound joint which is usually between different limbs, such as a leg and a arm or a left arm and a right arm. We have found that the first case requires precise collision detection to prevent from reducing a range of movement. Safety margin around objects are undesirable. However in the latter case, safety margin is required since the position between two different object in different limbs are not precisely determined because of mechanical deflection or joint angle error.

Therefore we argue that the hybrid self collision technique is necessary which combine highly precise collision

detection in a compound joint and collision detection with enough safety margins in an different limbs. We are now implementing the collision detection system which composed of on-line memory based collision detection and on-line geometry based collision detection.

Current implementation of the self collision detection and avoidance plugin stops whole body motion when self collision links are detected, however this looses balance when the robot is walking. Another implementations such as to stop motions only collided links or to output approximate motions are required. Using motion planner when recovering from collided posture for approximating original motion is also further research topic.

Another interesting issue is real-time computational resource control of collision detection. Since computation resource for a humanoid robot is limited. Servo loop, walking pattern generator, collision checker and sensor based motion generator need to share the resource. Especially computation time of sensor based motion generator may not be constant, computation time allowed for collision checking is not constant. Thus, setting computation time deadline for detecting collision is required. This problem includes hierarchical queries for detection self collisions and motion generator for preventing and recovering from self collisions. Pioneering research to this approach is addressed in [16].

REFERENCES

[1] F. Kanehiro and N. Miyata and S. Kajita and K. Fujiwara and H. Hirukawa and Y. Nakamura and K. Yamane,K and T. Kohara and Y. Kawamura and Y. Sankai. Virtual Humaonid Robot Platform to Develop Controllers of Real Humanoid Robots without Porting. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'01)*, pages 1093–1099, 2001.

[2] J.J. Kuffner and K. Nishiwaki and S. Kagami and Y. Kuniyoshi and M. Inaba and H. Inoue. Self-Collision Detection and Prevention for Humanoid Robots. In *In Proceedings of 2002 IEEE International Conference on Robotics and Automation (ICRA '02)*, pages 2265–2270, 2001.

[3] H. Hirukawa, F. Kanehiro, K. Kaneko, S. Kajita, K. Fujiwara, Y. Kawai, F. Tomita, S. Hirai, K. Tanie, T. Isozumi, K. Akechi, T. Kawasaki, S. Ota, K. Yokoyama, H. Honda, Y. Fukase, J. Maeda, Y. Nakamura, S. Tachi, and H. Inoue. Humanoid Robotics Platforms developed in HRP. In *Proceedings of the 2003 IEEE-RAS International Conference on Humanoid Robots (Humanoids 2003)*, 2003.

[4] M. Lin and S. Gottschalk. Collision Detection between Geometric Models: A Survey. In *In the Proceedings of IMA Conference on Mathematics of Surfaces*, 1998.

[5] M. Lin and D. Manocha and J. Cohen and S. Gottschalk. Collision Detection: Algorithms and Applications. In *In Proceedings of Algorithms for Robotics Motion and Manipulation*, pages 129–142, 1999.

[6] ColDet - Free 3D Collision Detection Library. *http://coldet.sourceforge.net/*.

[7] OPCODE – Optimized Collision Detection –. *http://www.codercorner.com/Opcode.htm*.

[8] S. Gottschalk and M. C. Lin and D. Manocha. OBB-Tree: A Hierarchical Structure for Rapid Interference Detection. In *In Proceedings of ACM Siggraph '96*, pages 171–180, 1996.

[9] T. Hudson and M. Lin and J. Cohen and S. Gottschalk and D. Manocha. V-COLLIDE: Accelerated Collision Detection for VRML. In *In Proceedings of VRML'97*, 1997.

[10] E.Larsen and S. Gottschalk and M. C. Lin and D. Manocha. Fast Proximity Queries with Swept Sphere Volumes. Technical Report Technical report TR99-018, Department of Computer Science, University of North Carolina, 1999.

[11] J.T. Klosowski and M. Held and J.S.B. Mitchell and H. Sowizral and K. Zikan. Efficient Collision Detection Using Bounding Volume Hierarchies of k-DOPs. *IEEE Transactions on Visualization and Computer Graphics*, 4(1):21–36, 1998.

[12] J. Cohen and M. Lin and D. Manocha and K. Ponamgi. I-COLLIDE: An Interactive and Exact Collision Detection System for Large-Scaled Environments. In *In Proceedings of ACM International 3D Graphics Conference*, pages 189–196, 1995.

[13] S. A. Ehmann and M. C. Lin. Accelerated Proximity Queries Between Convex Polyhedra By Multi-Level Voronoi Marching. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'00)*, pages 2101–2106, 2000.

[14] G. van den Bergen. A Fast and Robust GJK Implementation for Collision Detection of Convex Objects. *Journal of Graphics Tools*, 4(2):7–25, 1999.

[15] Y. Ishiwata, T . Matsui, and Y. Kuniyoshi. Development of Linux which has advanced real-time processing function. In *Proceedings of the 16th Annual Conference of Robotics Society of Japan*, pages 355–356, 1998 (in Japanese).

[16] H. Hirukawa and Y. Ishiwata and K. Toda. Imprecise but Realtime Algorithm for Collision Detection. In *Proceedings of the 27th Annual Conference of Robotics Society of Japan*, pages 1029–1030, 1999 (in Japanese).